Data Augmentation using GANs

Petros Kafkas

June 24, 2023

Table of Contents

Project Description

- The purpose of this project is to make use of generative AI in order to address the problem of class imbalance.
- To this end, we experiment using different generative AI architectures, from autoencodes to various GANs
- We extracted accuracy measurements for the various solutions tested and compared them against a baseline.

Project Pipeline

step 1: Create an imbalanced dataset (for this case, MNIST)

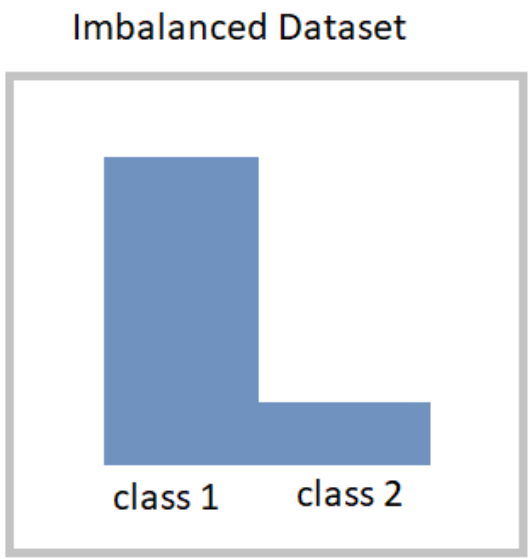


Figure: Project Pipeline

Project Pipeline

step 2: Feed the imbalanced class to a Generative Model

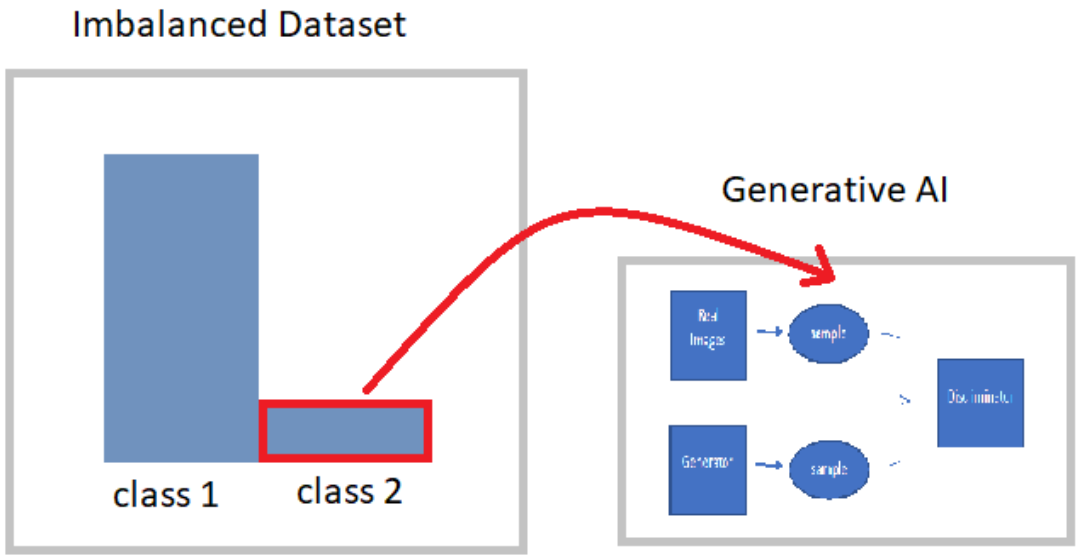


Figure: Project Pipeline

Project Pipeline

step 3: Generate *Artificial* data to correct the imbalance

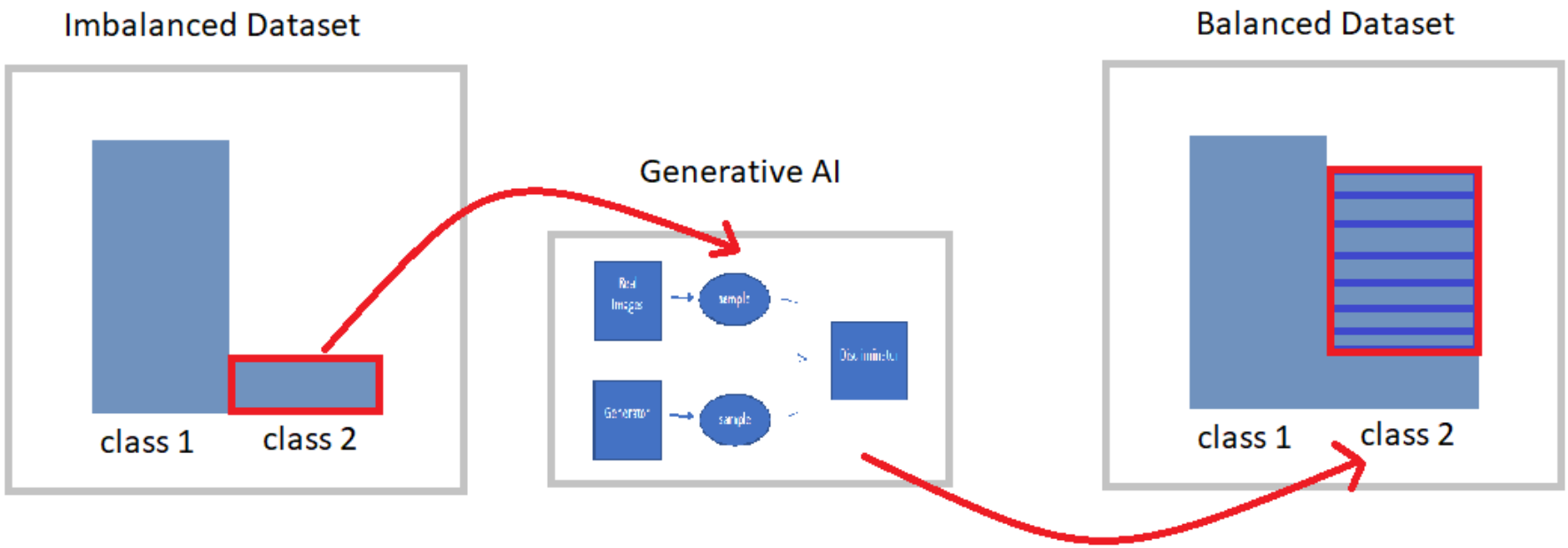


Figure: Project Pipeline

Project Pipeline

In more detail:

- We deleted $\backsim$ 99% of all the data corresponding to number 5.
- We used a standard CNN in order to measure the class accuracy (baseline measurement)
- Extra: We applied standard augmentations (crop, rotation etc.) and measured again in order to compare the Generative Methods to the classic data augmentation approach.

Data Visualization

Figure: Dataset Visualization (Balanced)

Data Visualization

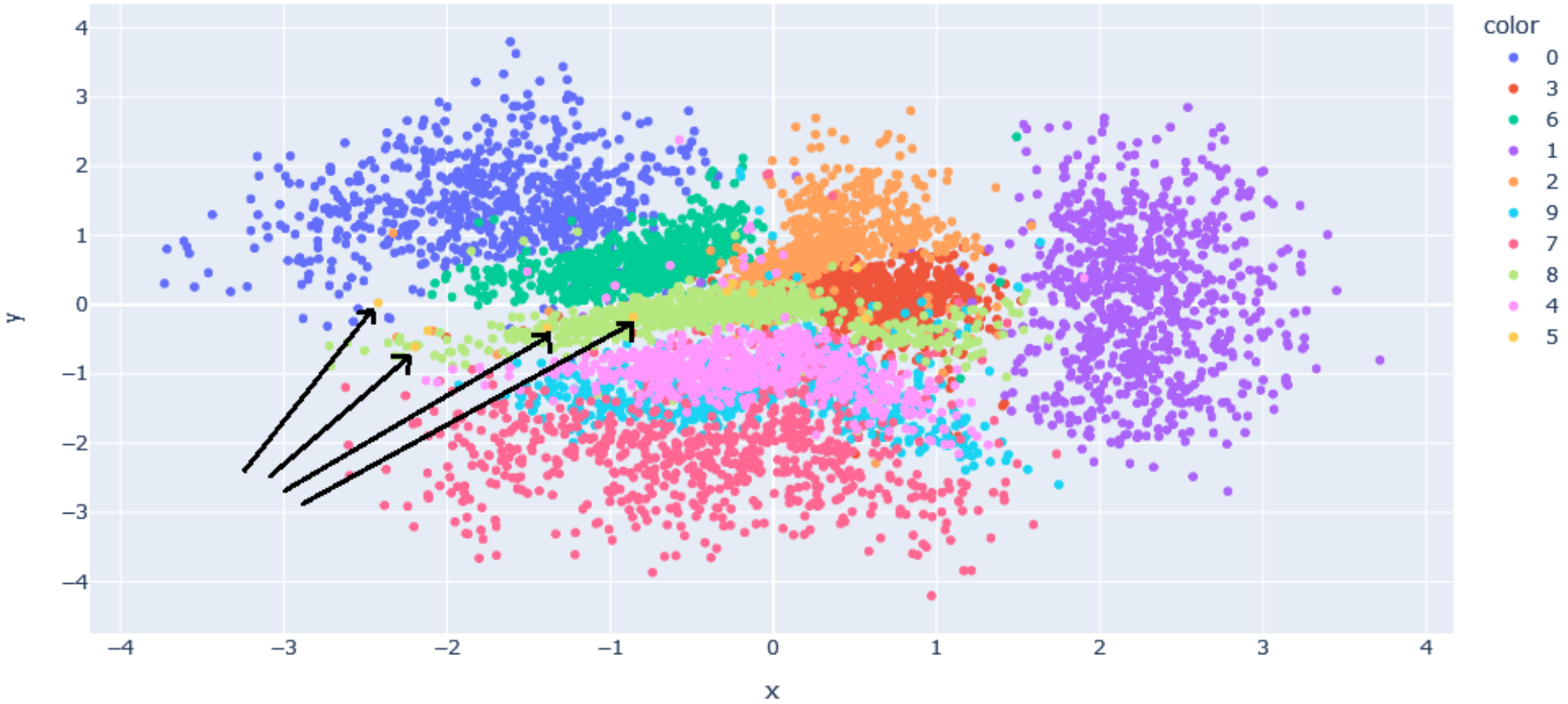


Figure: Dataset Visualization (Imbalanced)

Generative Models

- Autoencoders / VAEs
- GANs / WGANs
- ProGAN
- CGAN

Variational Autoencoders

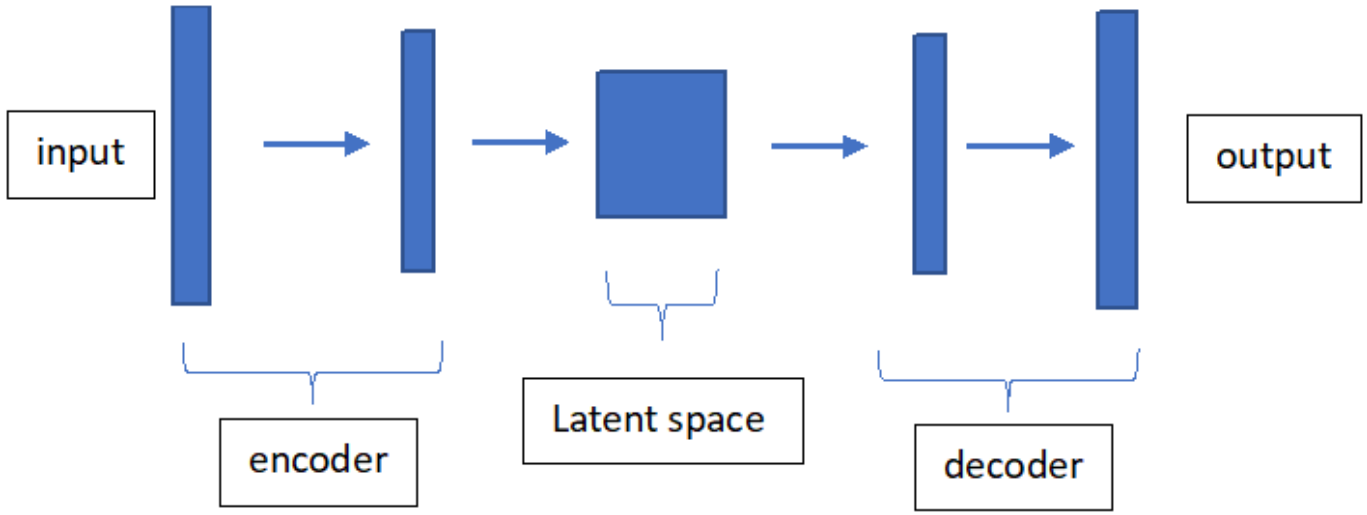


Figure: Basic Autoencoder

- An AE learns two functions, an encoding function that transforms the input data, and a decoding that recreates the input data from the encoded representation
- The latent space is rerpesented by a fixed vector

Variational Autoencoders

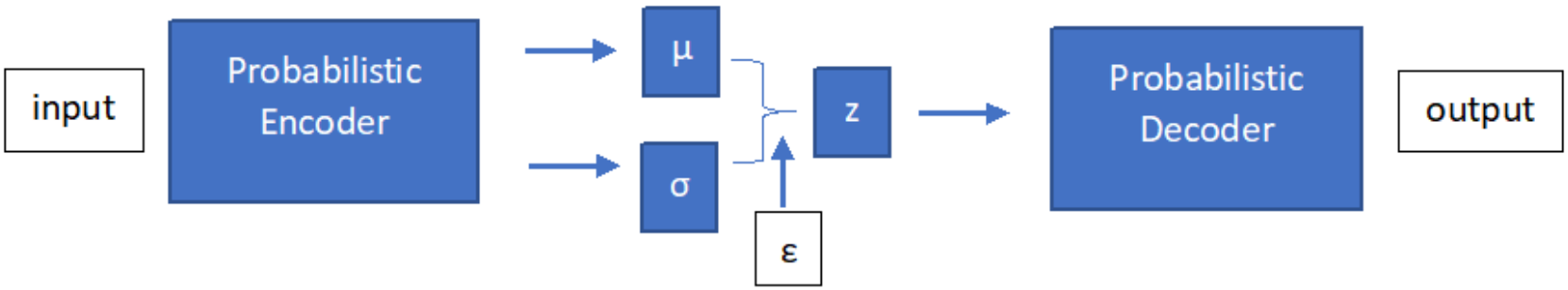


Figure: Variational Autoencoder

The *encoder* neural network maps the input variable to a latent space that corresponds to the parameters of a variational distribution. If the latent space is regular enough (well "organized" by the encoder during the training process), we could take a point randomly from that latent space and decode it to get a new content

Variational Autoencoders

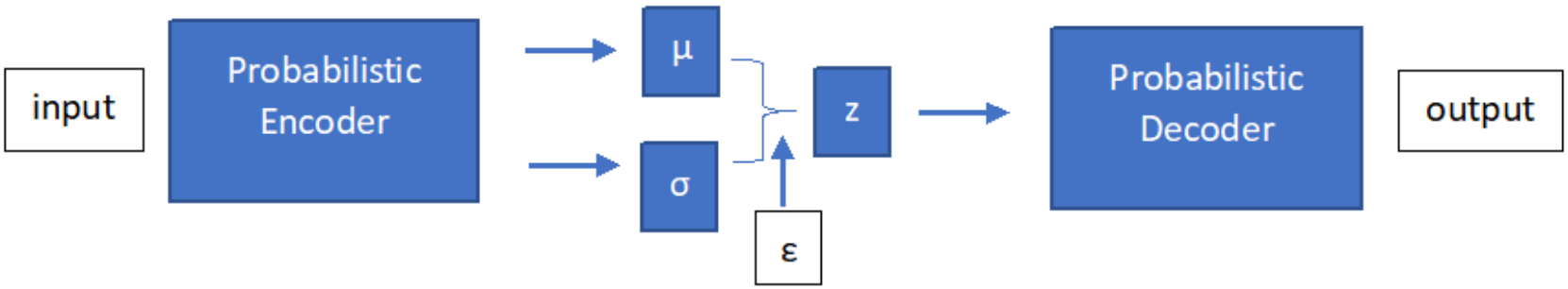


Figure: Variational Autoencoder

$$\text{Loss L} = E_{q_\phi(x|z)}[\log p_\theta(x|z)] - \mathrm{D_{KL}}(q_\phi(z|x)||p(z))$$

Figure: Variational Autoencoder

Variational Autoencoders

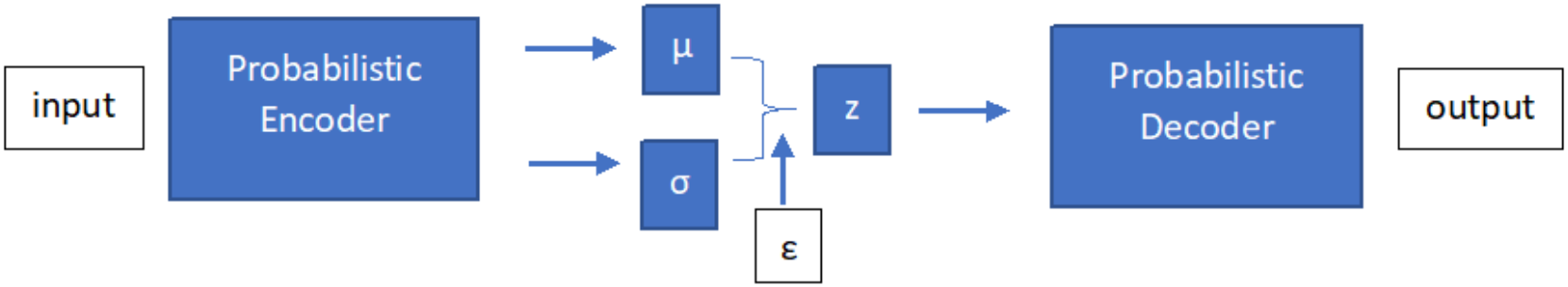


Figure: Variational Autoencoder

$$\text{Loss } L = \underbrace{E_{q_\phi(x|z)}[\log p_\theta(x|z)]}_{\text{Reconstruction Loss}} - D_{KL}(q_\phi(z|x)||p(z))$$

Figure: Variational Autoencoder

Variational Autoencoders

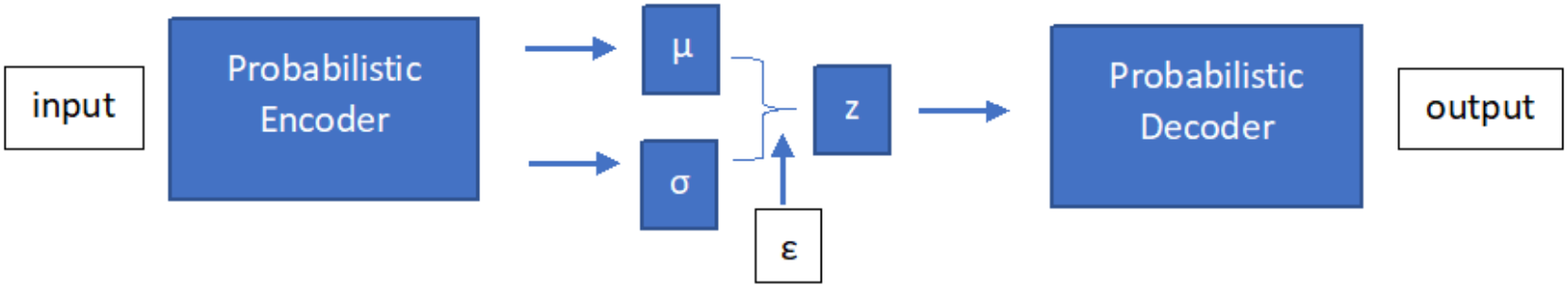


Figure: Variational Autoencoder

$$\text{Loss } L = \underbrace{E_{q_\phi(x|z)}[\log p_\theta(x|z)]}_{\text{Reconstruction Loss}} - \underbrace{D_{KL}(q_\phi(z|x)||p(z))}_{\text{Approximate Normal(0,1)}}$$

Figure: Variational Autoencoder

Variational Autoencoders

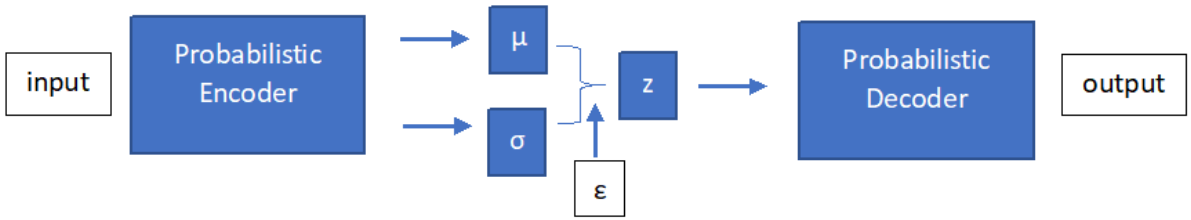


Figure: Variational Autoencoder

Cannot Back-propagate due to stochastic nature of sampling!

Variational Autoencoders

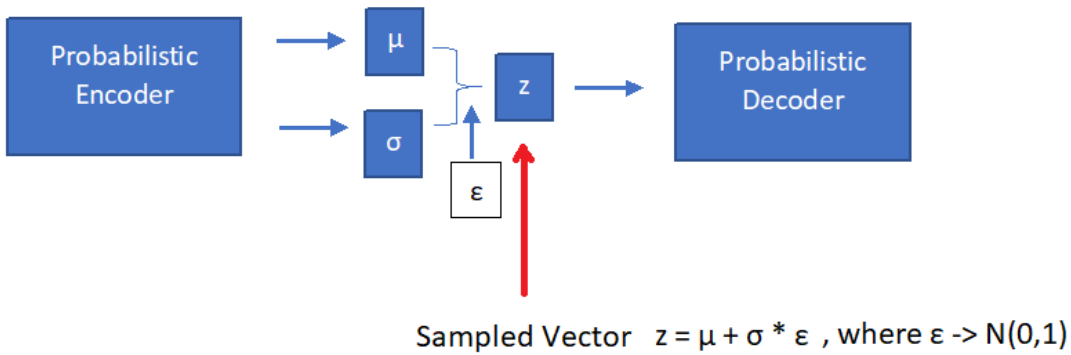
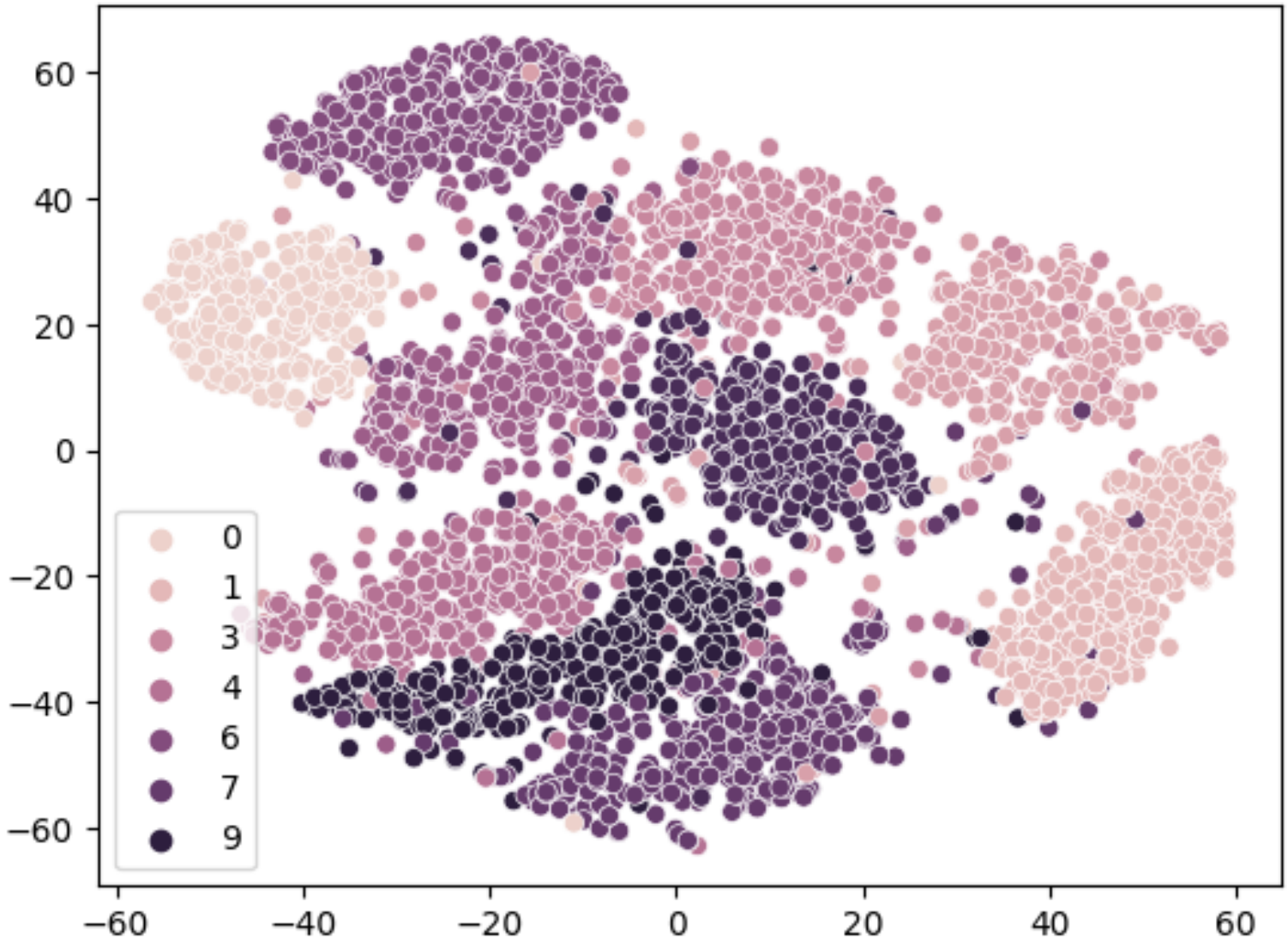


Figure: Latent Distribution Sampling

Reparametarization Trick: We sample ϵ from a normal distribution

Variational Autoencoders



Variational Autoencoders

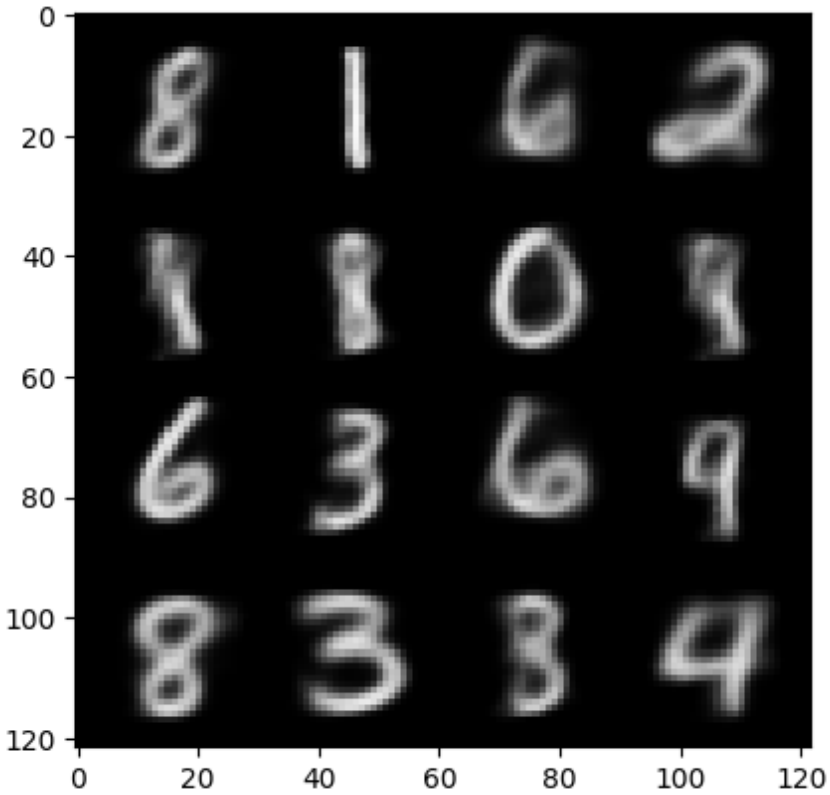


Figure: Data Generation using VAE

Basic Gan

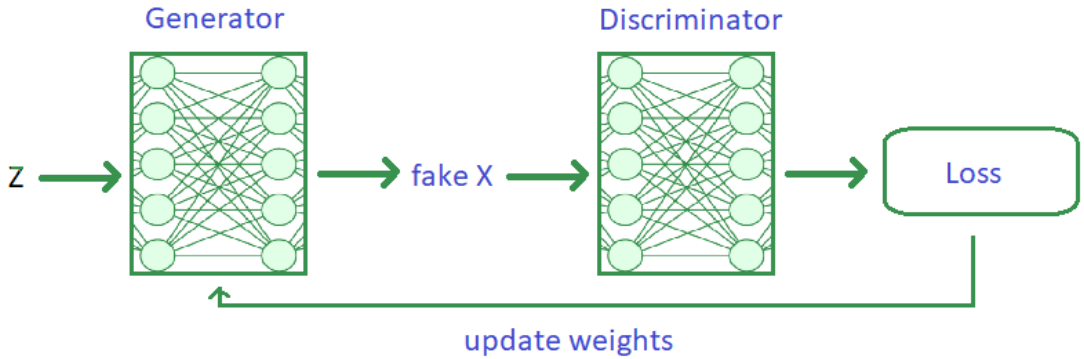


Figure: Generator Training

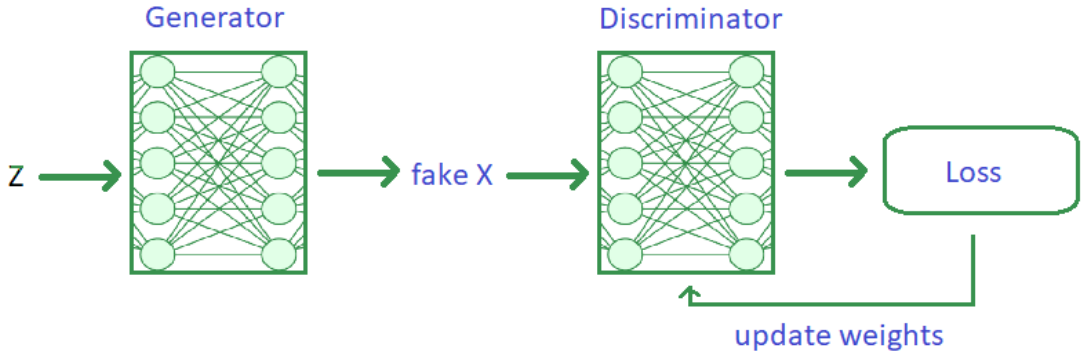


Figure: Discriminator Training

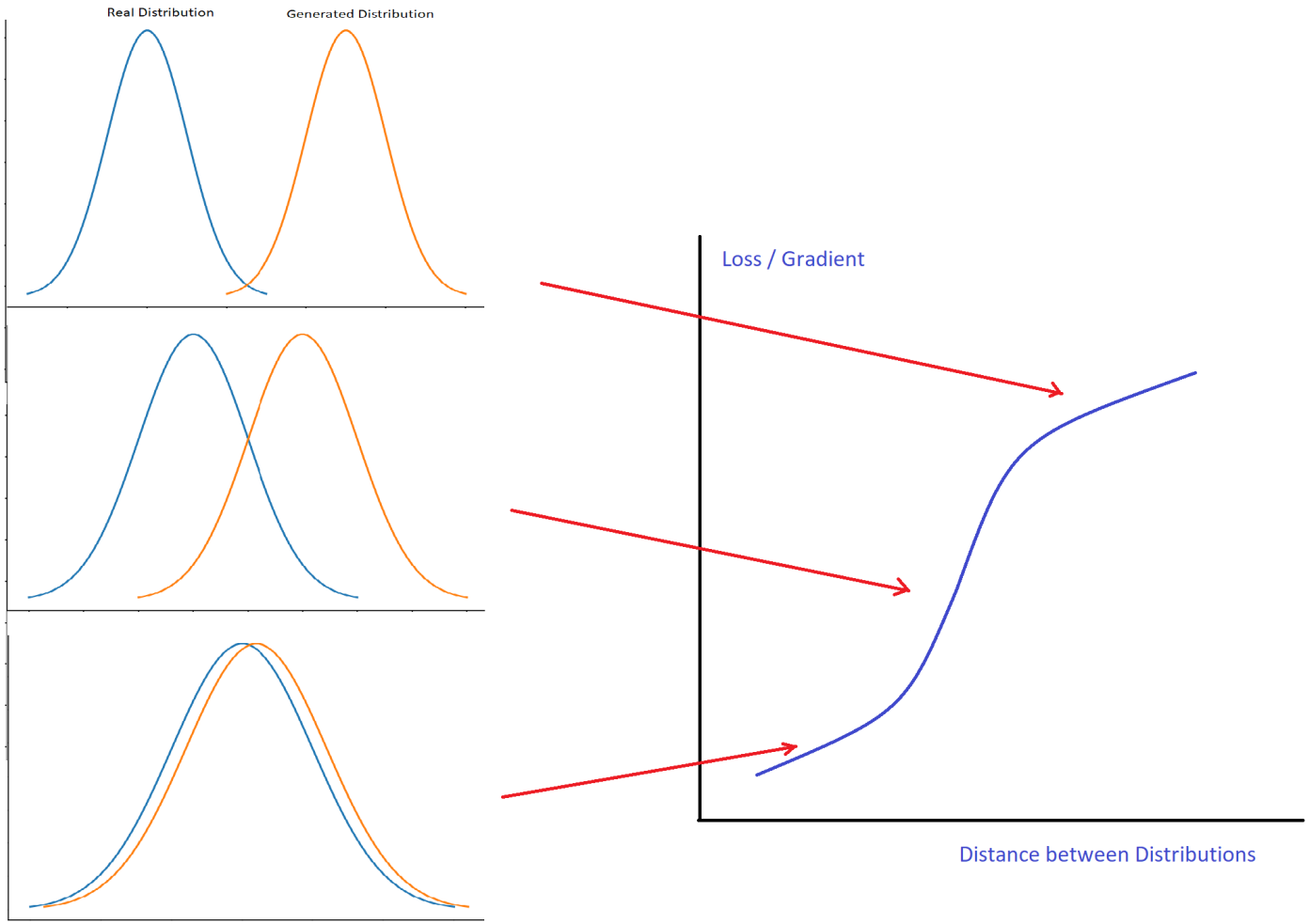
Basic Gan

Problems with the basic gan architecture:

- Mode Collapse: The generator gets stuck in a single mode
- Loss function (BCE) does not offer a good correlation between the loss value and the generated results
- Flat gradients during training:

Basic Gan

- Flat gradients during training:



Advanced Gan: WGAN

Solves the problems of the basic gan by introducing a new loss function: Wasserstein Loss.

$$\textit{Critic: } min_d - [E(D(x)) - E(D(G(x)))]$$
$$\textit{Generator: } min_g - [-E(D(G(x)))]$$
$$\textit{Sample Labels: } -\frac{1}{n}\sum_{n=1}^{\infty} y_n * pred_n$$

MinMax Game: The critic tries to maximize the difference between its real and fake predictions, while the generator tries to minimize that difference.
Important: gradient norm penalty to achieve Lipschitz continuity!

Advanced Gan: WGAN

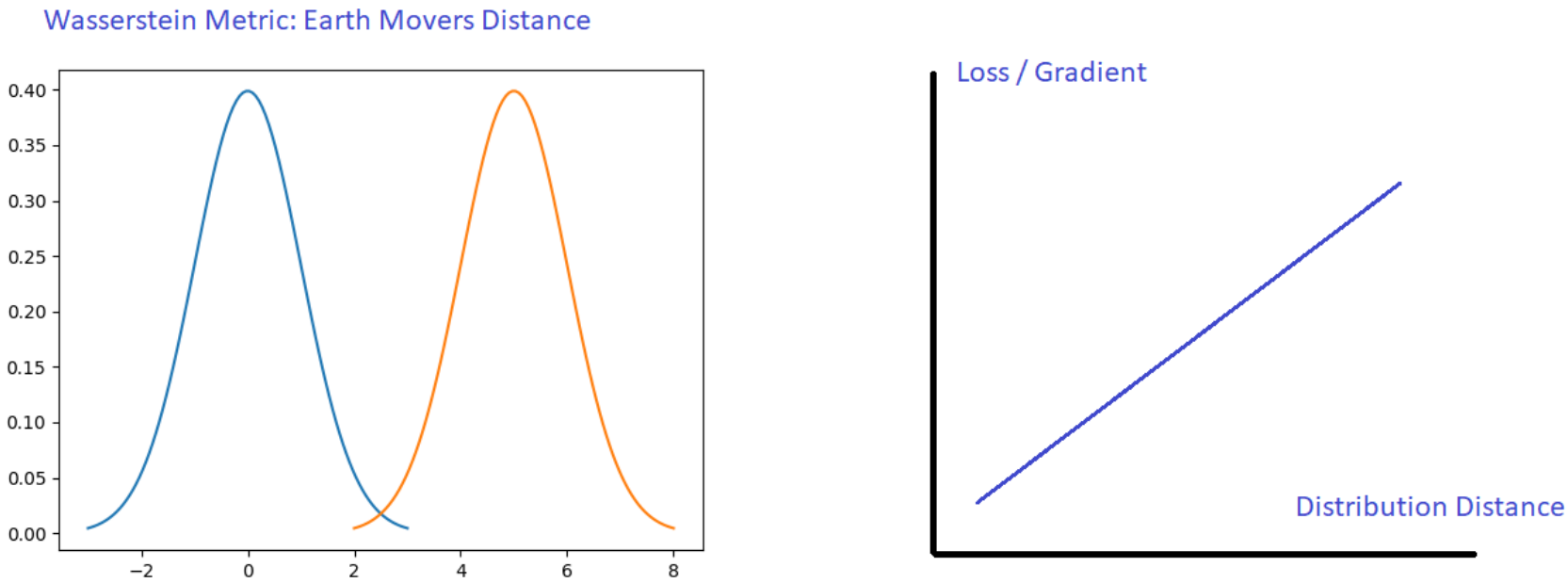


Figure: Earth's Mover Distance

Progressive Growing GAN

Key Idea: grow both the generator and discriminator progressively, starting from a low resolution, add new layers that model increasingly fine details as training progresses

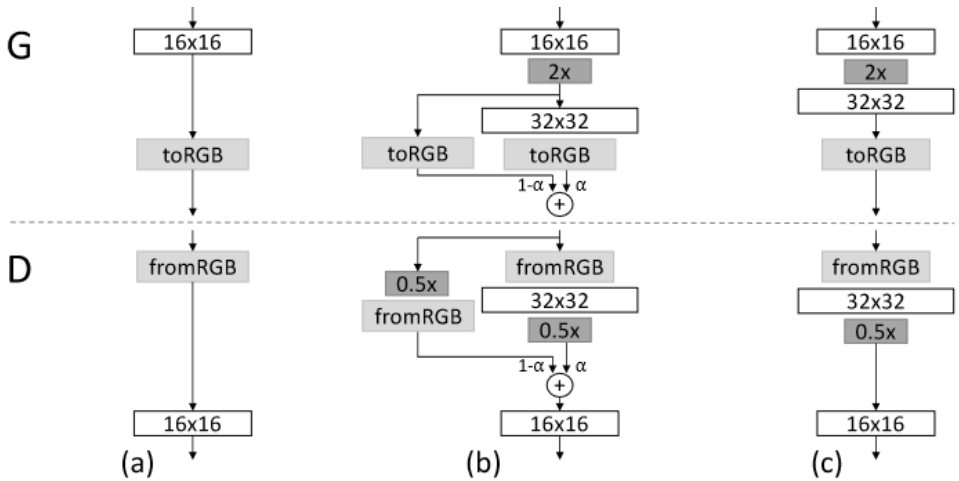


Figure: progan

Progressive Growing GAN

Key Idea: grow both the generator and discriminator progressively, starting from a low resolution, add new layers that model increasingly fine details as training progresses

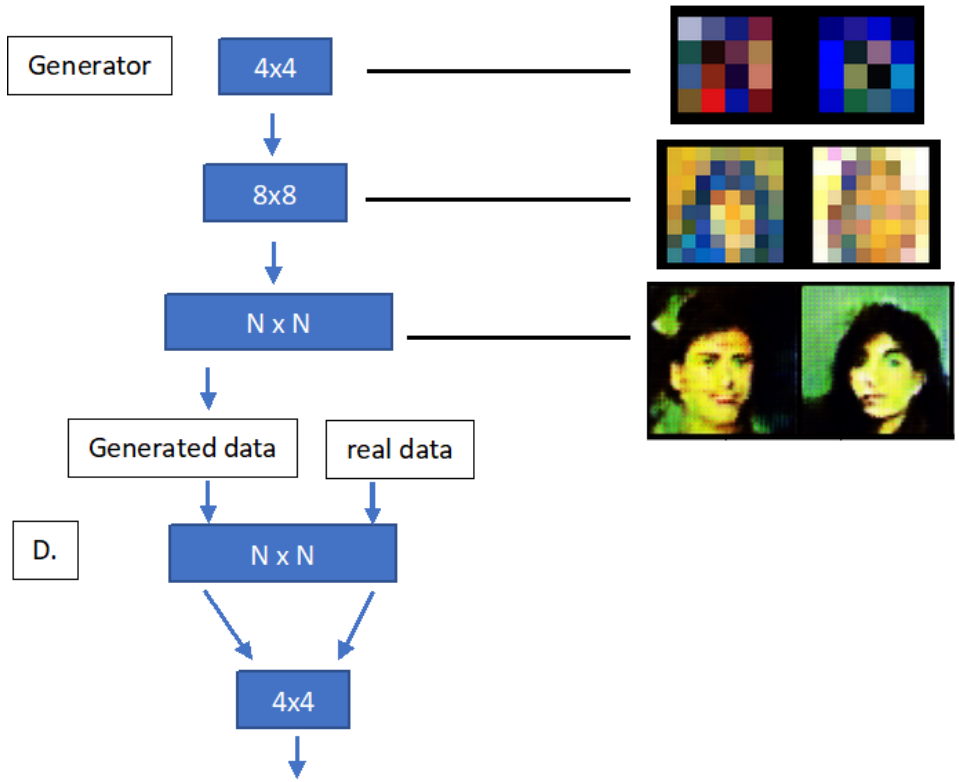
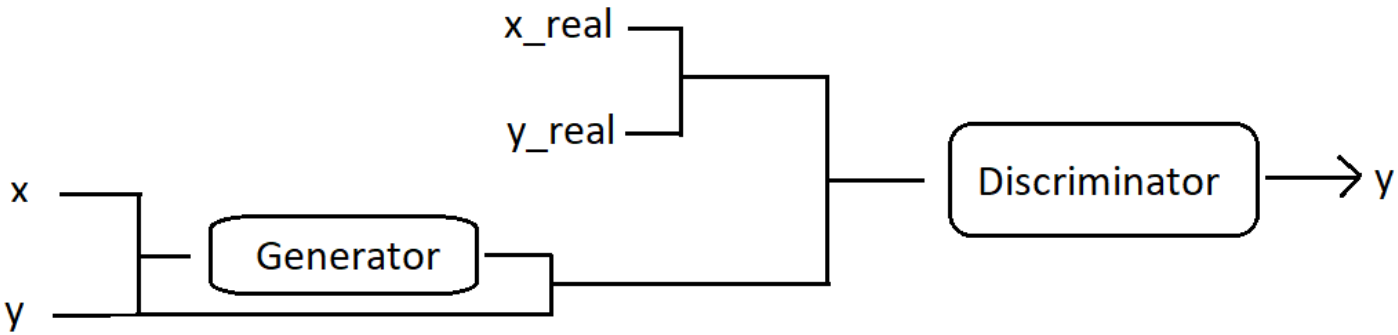


Figure: progan

Conditional WGAN



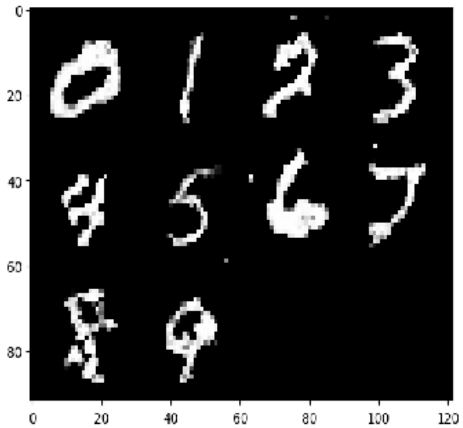


Figure: conditional gan

Experimental Results

For our experiment we used the Wasserstein GAN for data generation, as it was the most *"cost-effective"* in terms of quality of data generated and of training time.

We repeated the experiment several times for different amount of data generated: 1) class imbalance 99.9% (baseline), 96%, 90%, 75%, 45%, and finally, 10%.

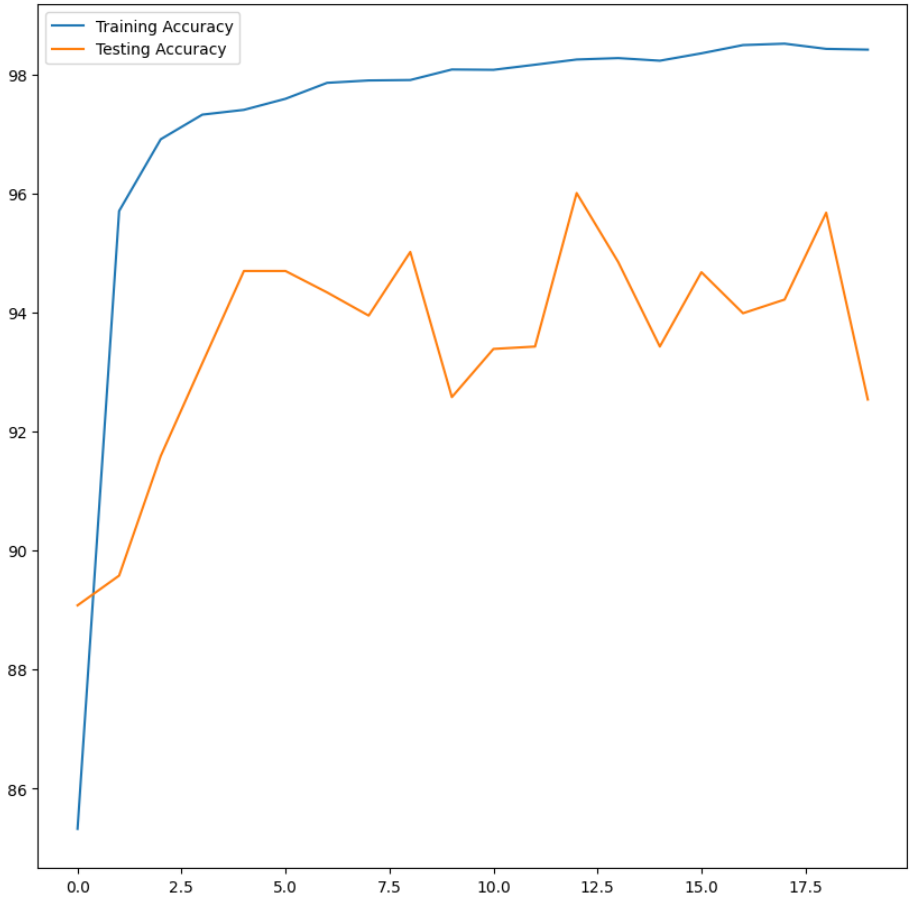
We measured the accuracy using a convolutional based classifier with two covolutional layers followed by a linear layer.

Experimental Results

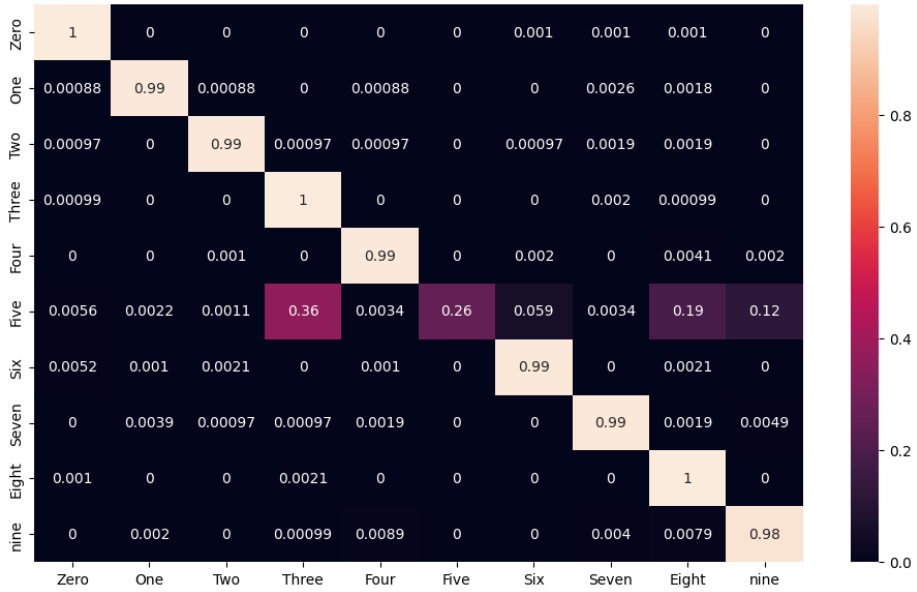
```
(cnn_model): Sequential(
  (0): Sequential(
    (0): Conv2d(1, 8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.2, inplace=False)
  )
  (1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (2): Sequential(
    (0): Conv2d(8, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.2, inplace=False)
  )
  (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)
(lin_model): Sequential(
  (0): Sequential(
    (0): Linear(in_features=1568, out_features=600, bias=True)
    (1): ReLU(inplace=True)
    (2): Dropout(p=0.5, inplace=False)
  )
  (1): Linear(in_features=600, out_features=10, bias=True)
)
```

Figure: classifier network

Experimental Results



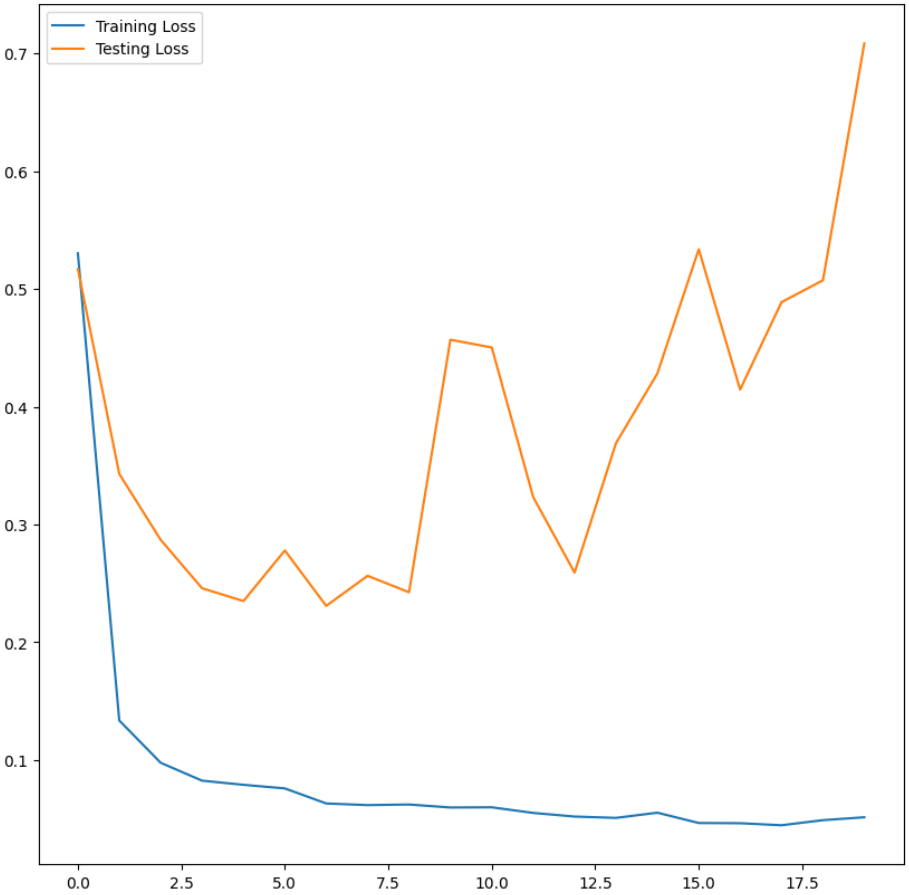
(a) accuracy



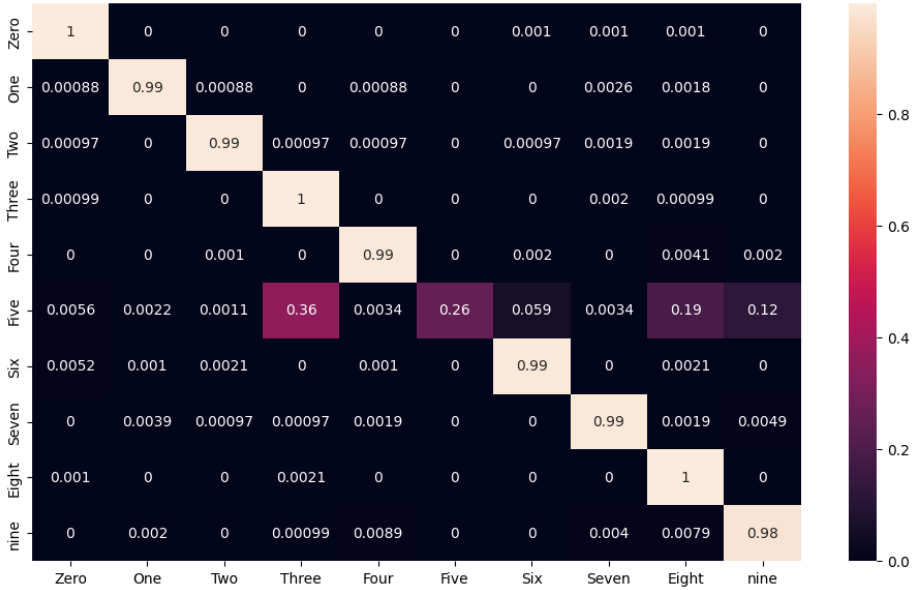
(b) confusion matrix

Figure: Baseline (99% imb.)

Experimental Results



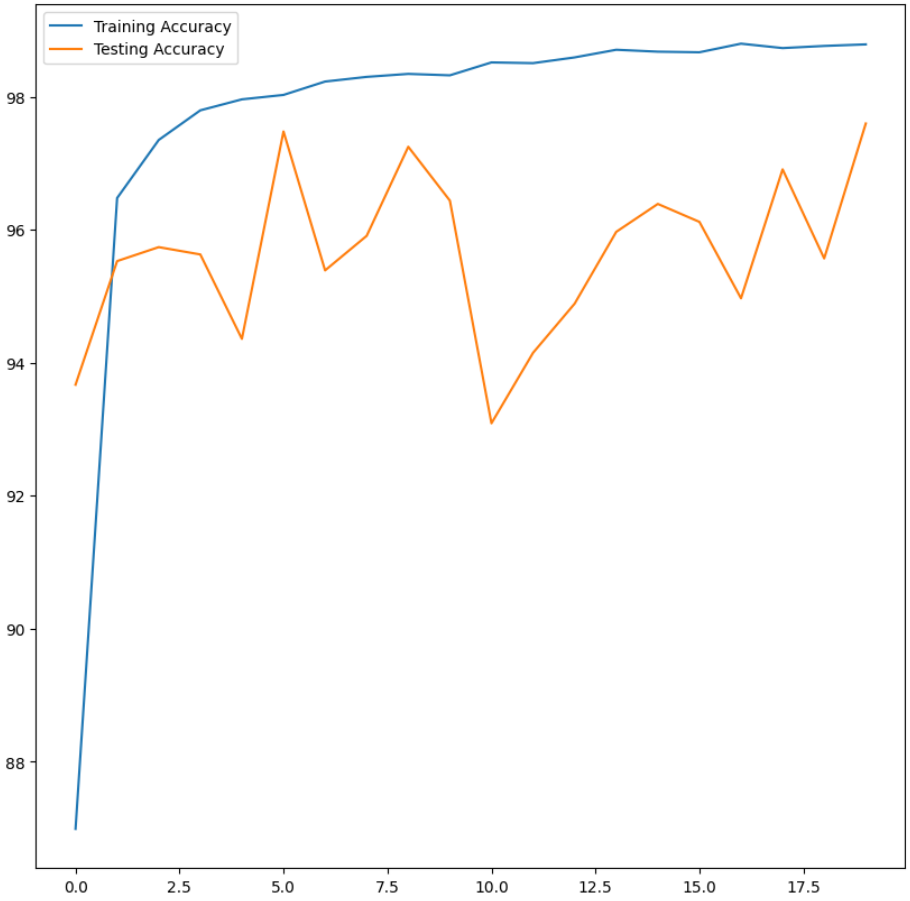
(a) accuracy



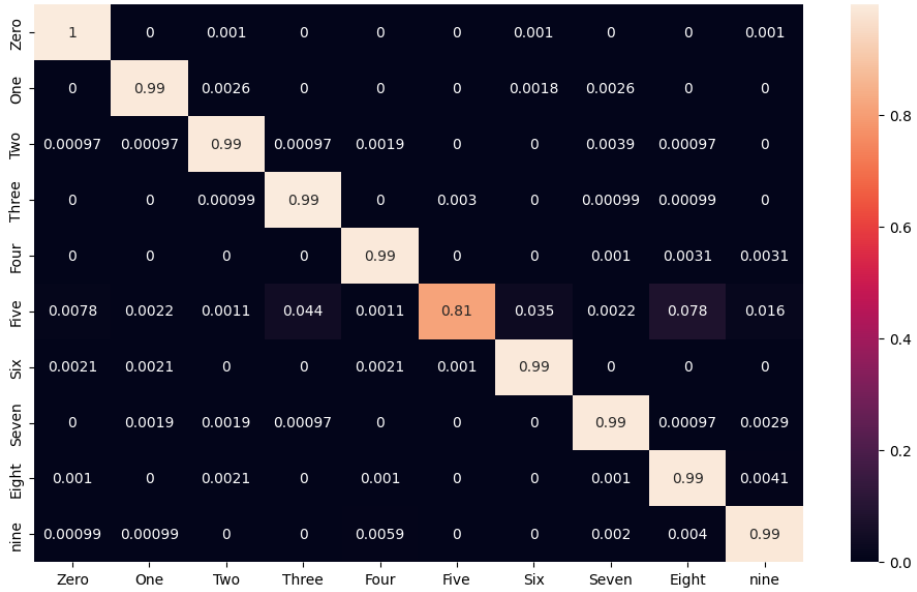
(b) confusion matrix

Figure: Baseline (99% imb.)

Experimental Results



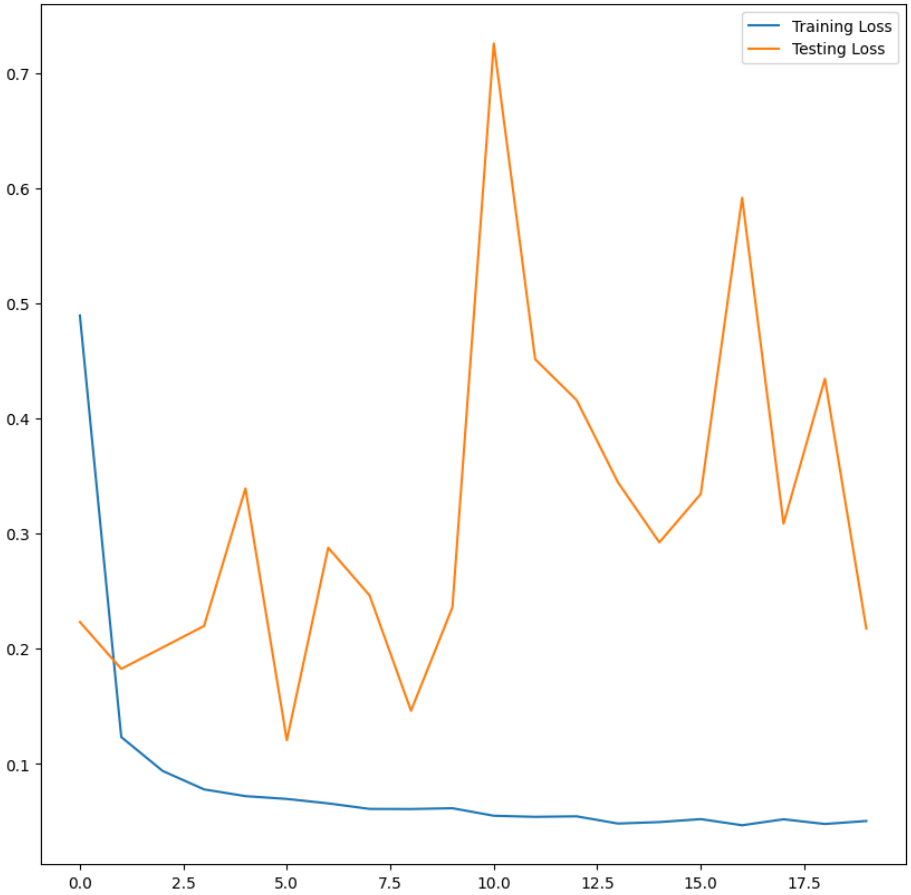
(a) accuracy



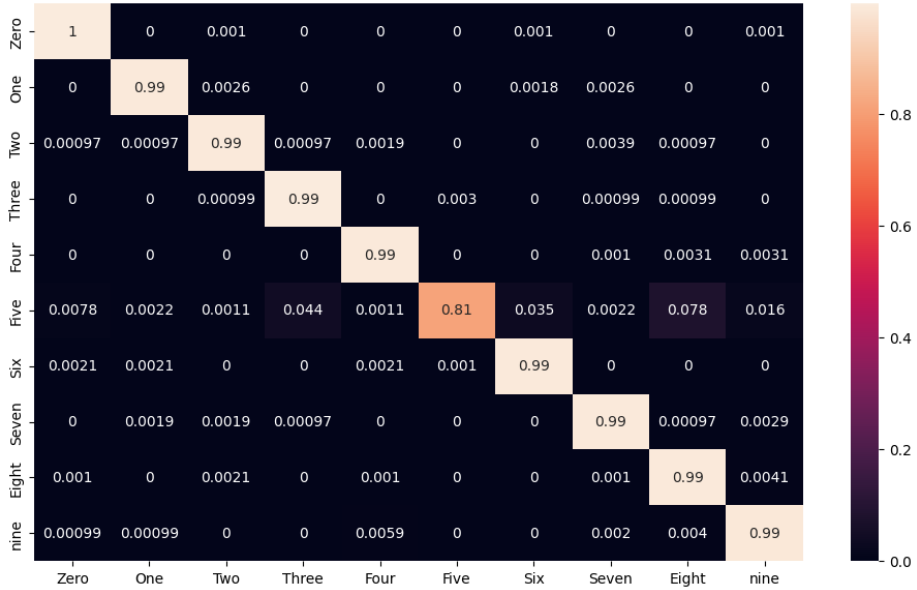
(b) confusion matrix

Figure: 75% imb.

Experimental Results



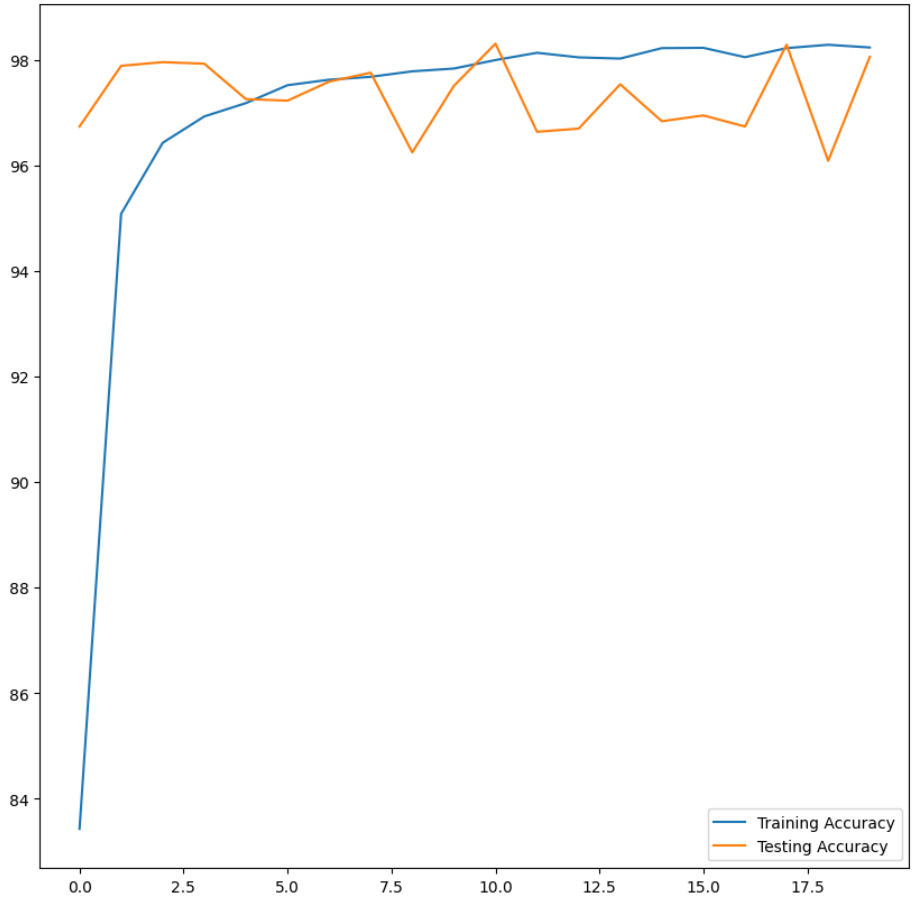
(a) accuracy



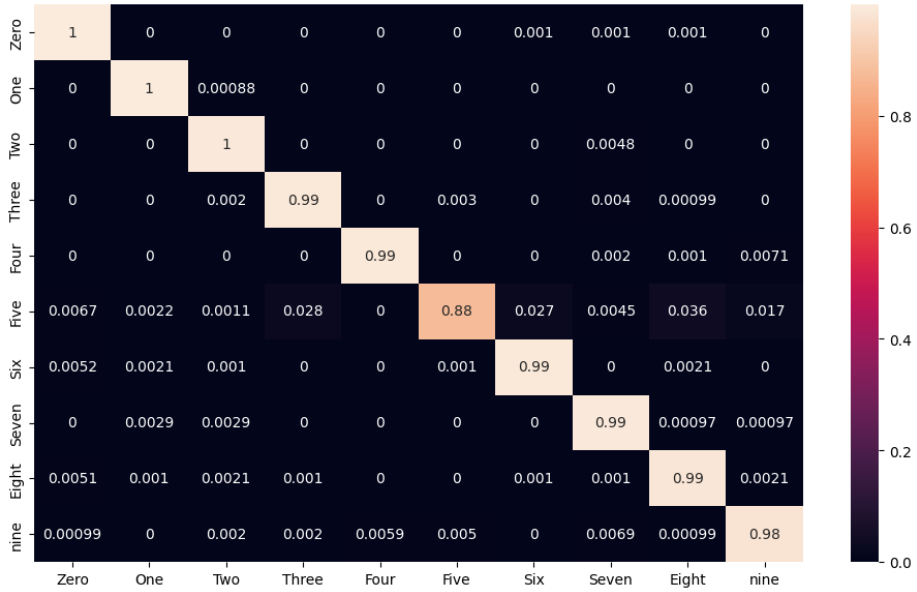
(b) confusion matrix

Figure: 75% imb.

Experimental Results



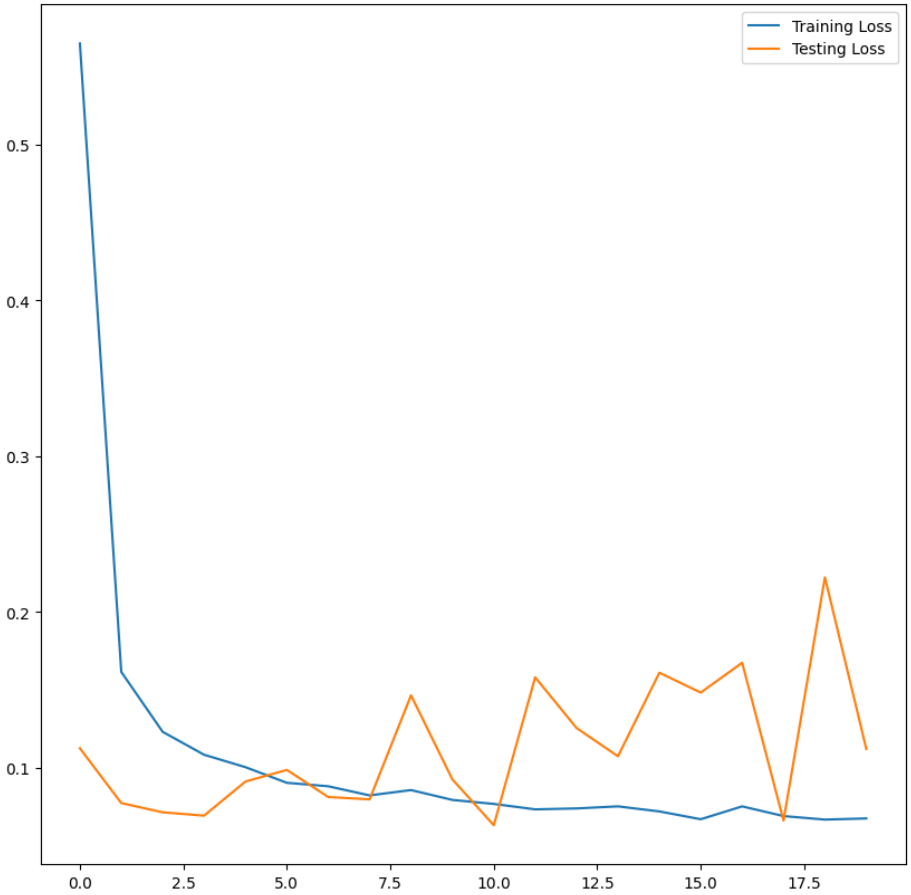
(a) accuracy



(b) confusion matrix

Figure: 50% imb.

Experimental Results

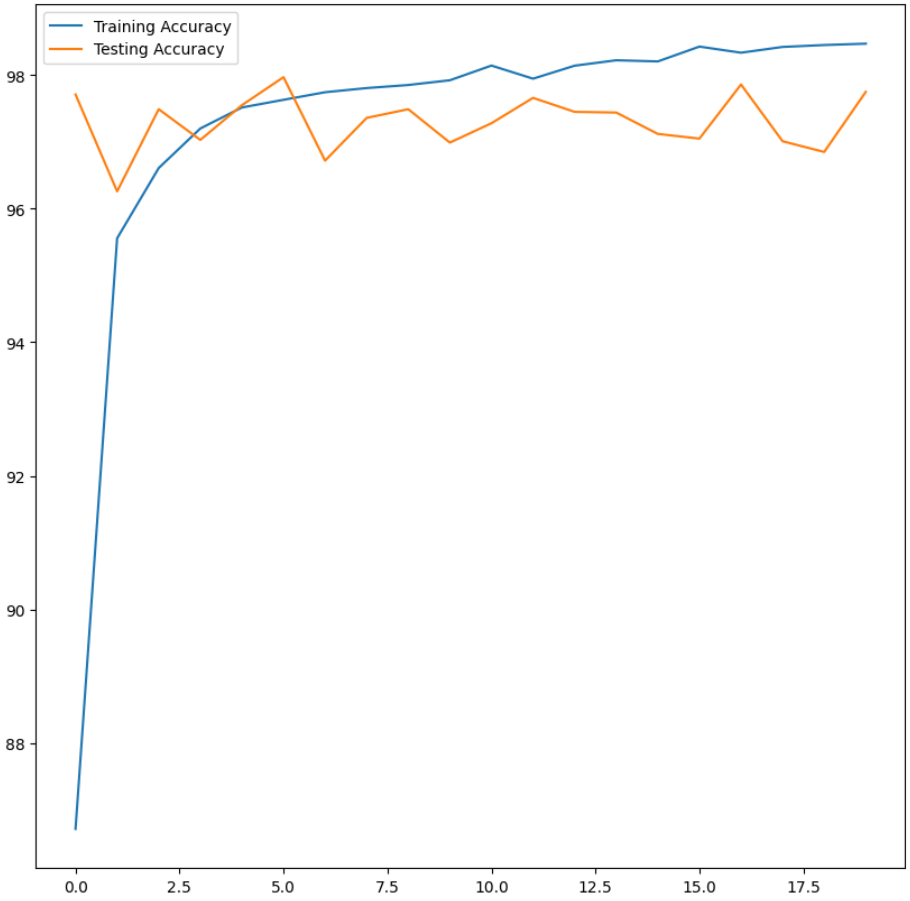


(a) accuracy

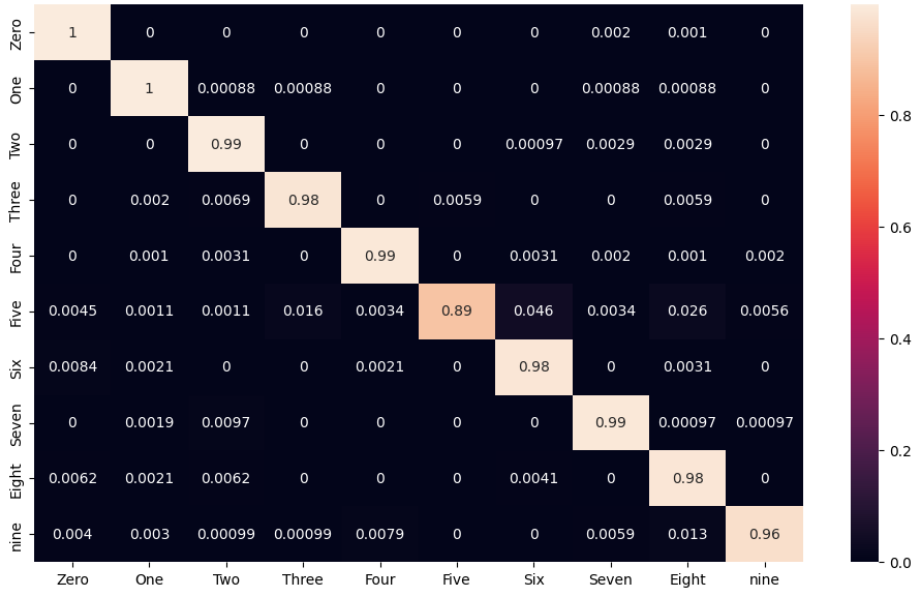
(b) confusion matrix

Figure: 50% imb.

Experimental Results



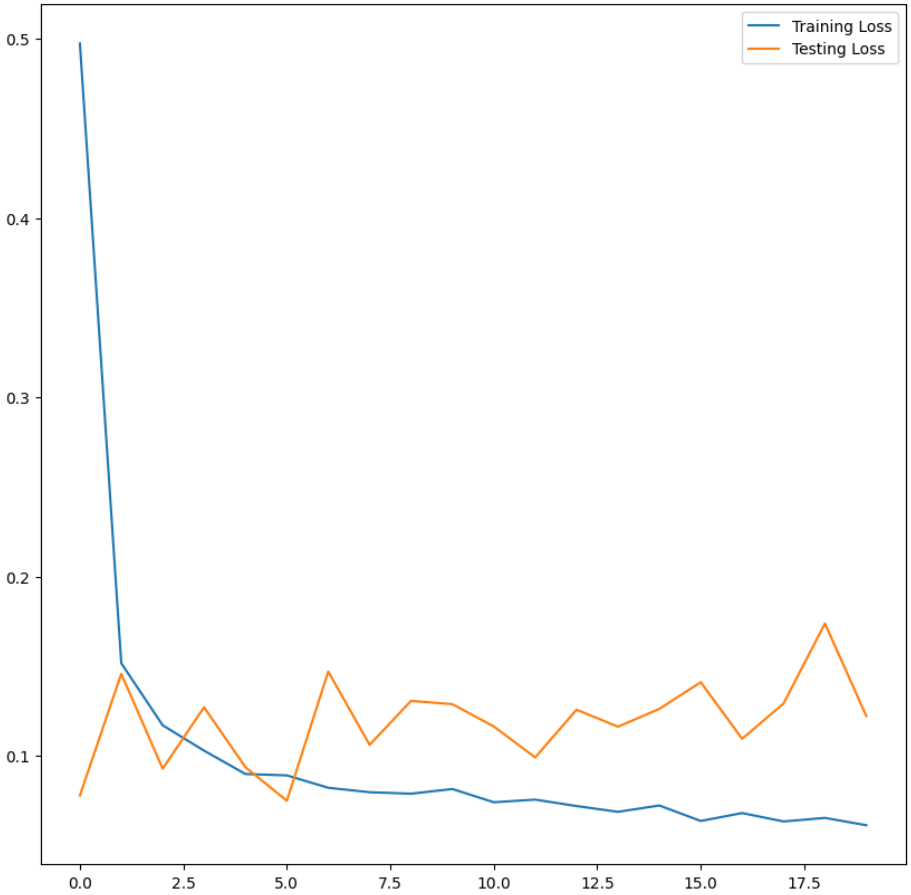
(a) accuracy



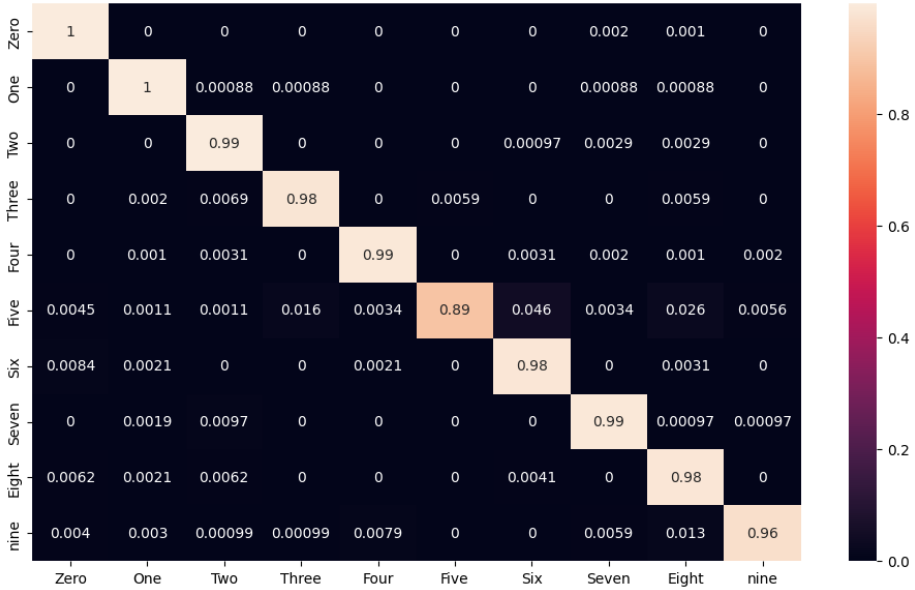
(b) confusion matrix

Figure: 10% imb.

Experimental Results



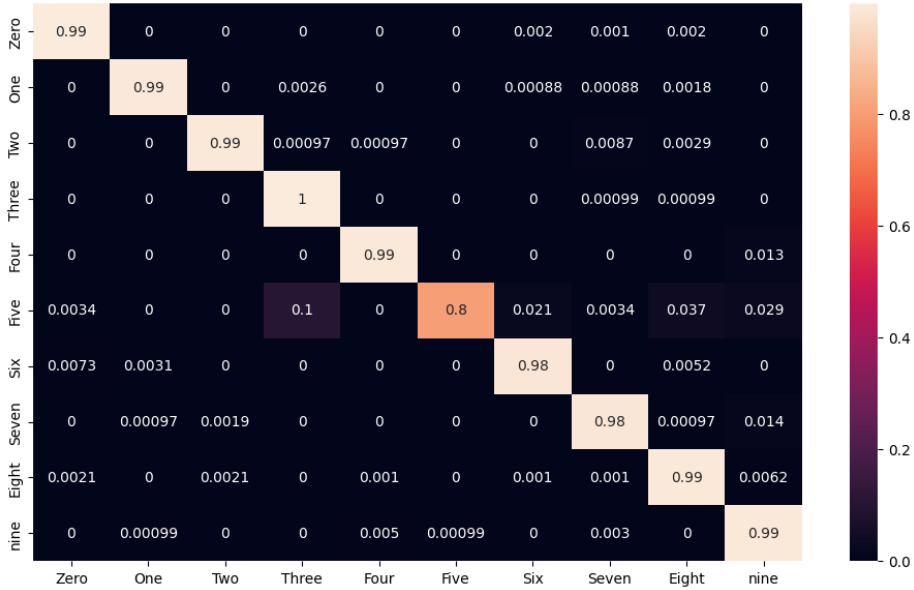
(a) accuracy



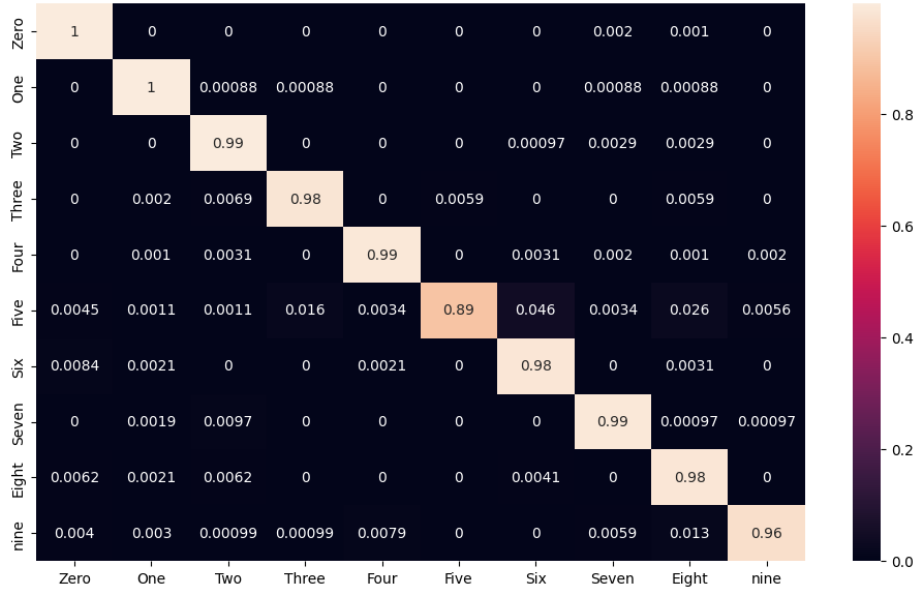
(b) confusion matrix

Figure: 10% imb.

Comparison with traditional data augm. techniques



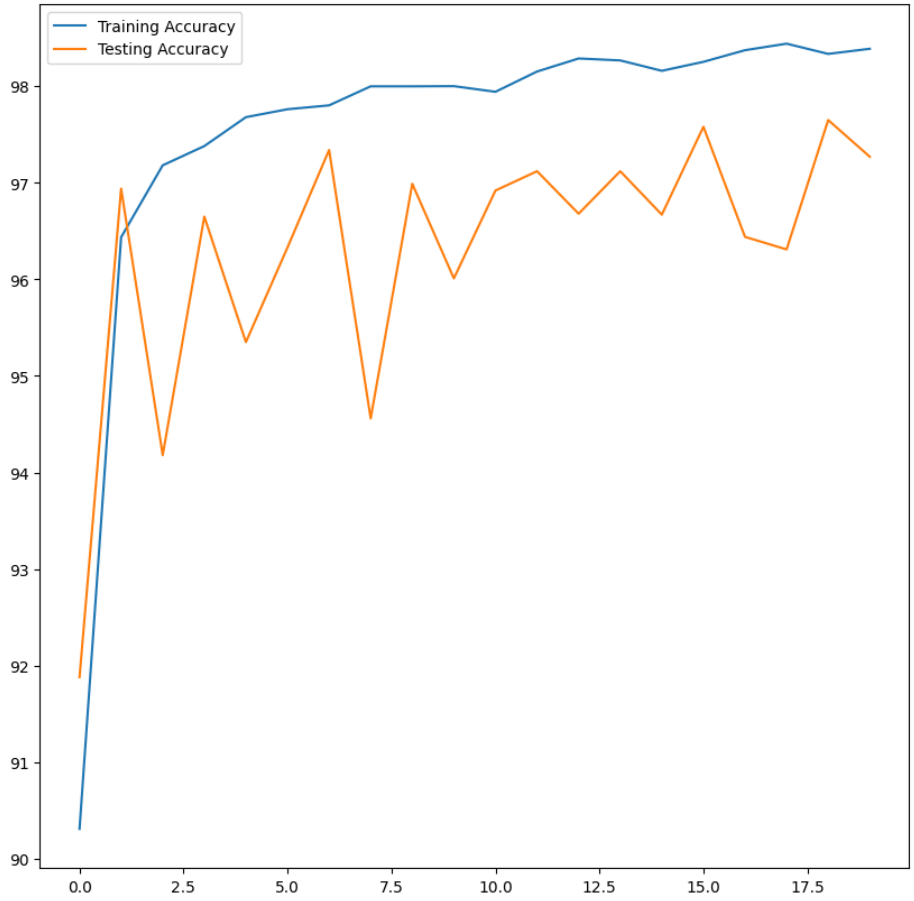
(a) Aumented CM



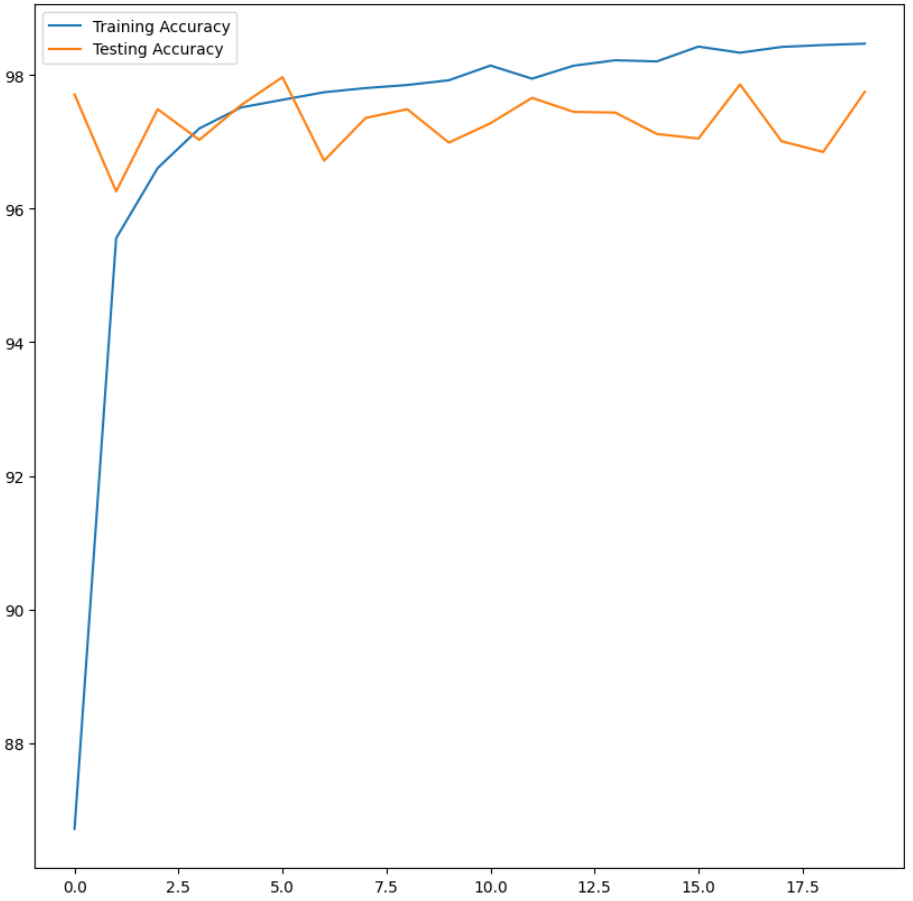
(b) GAN confusion matrix

Figure: GAN vs (crop, rotation etc)

Comparison with traditional data augm. techniques



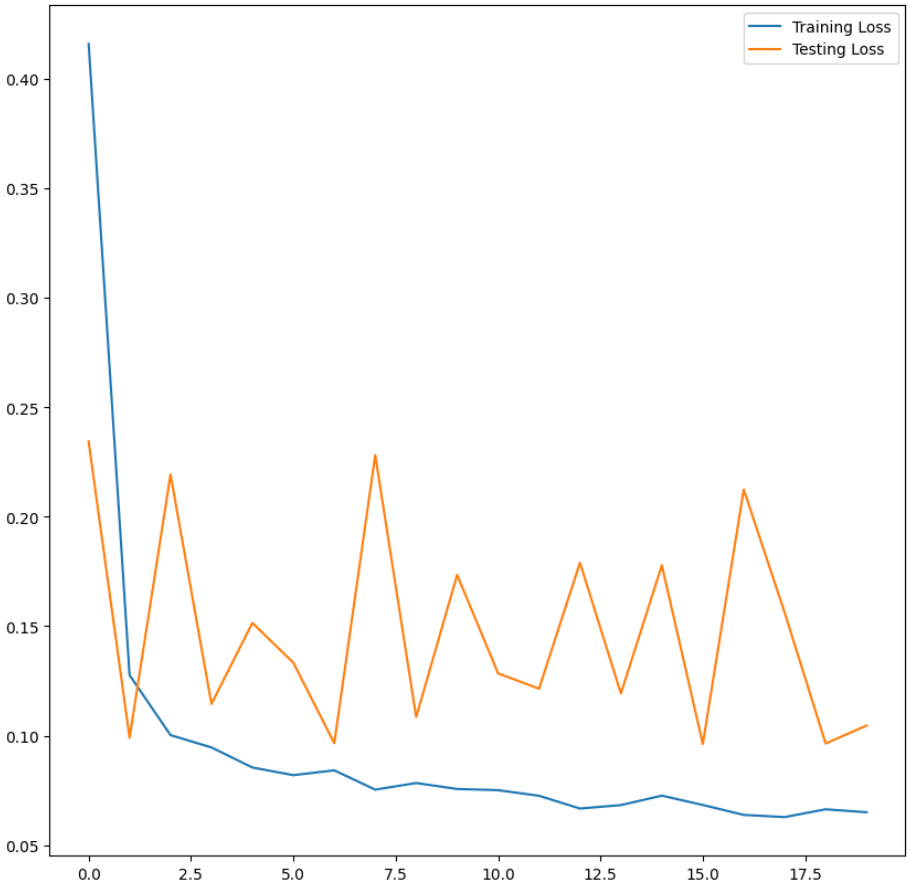
(a) AUG. accuracy



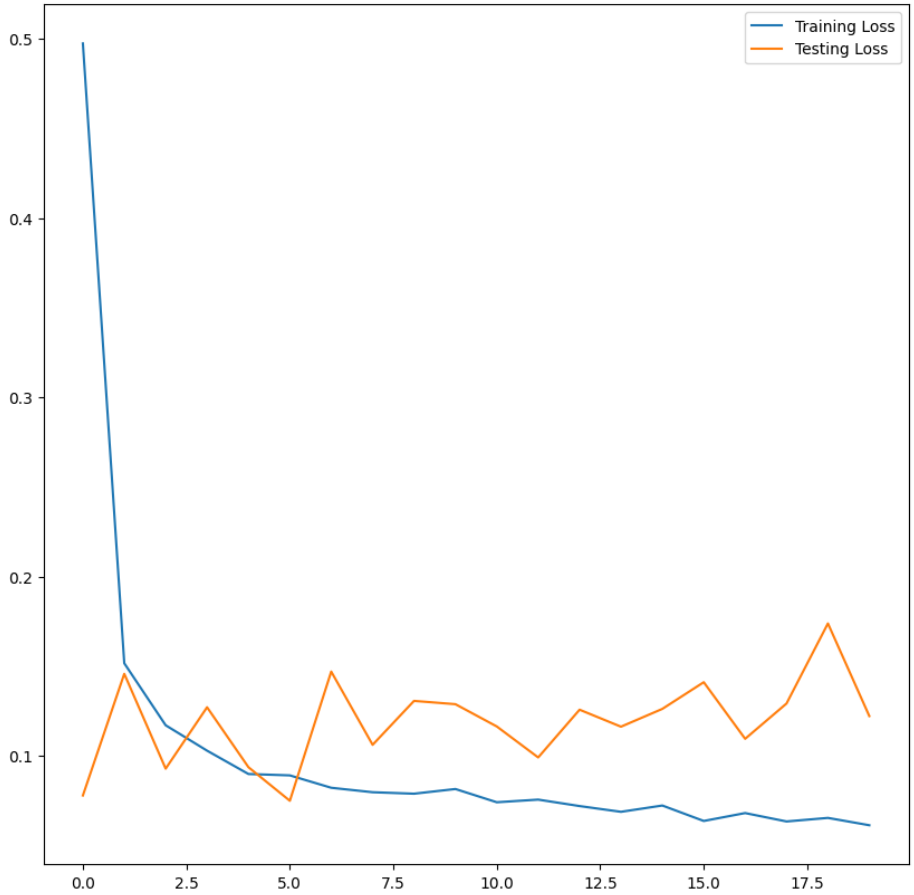
(b) GAN accuracy

Figure: GAN vs (crop, rotation etc)

Comparison with traditional data augm. techniques



(a) AUG. Loss



(b) GAN Loss

Figure: GAN vs (crop, rotation etc)

Conclusion

Comparison summing points:

- The ai - generated data significantly improved the accuracy of the base classifier although the data did not capture all the statistical properties of the testing samples. However, this was also a problem when we used crop-rotation and other typical data aumentation techniques.
- The gans method offered better results compared to the traditional way of augmenting data, although it required significantly more time.
- Factrors to consider when deciding whether to use generative Ai for data aumentation:
  problem and data complexity, availability of data - data complexity ratio, time constraints.

Conclusion

Suggestions for further investigation:

- Combining VAEs with GANs for data aumentation
- Combining traditional aumentation techniques + generative AI methods

Thank you!