ID1: 208560086
Name1: Naor Cohen

ID2: 301613501
Name2: Or Trabelsi

EX3 - VAE, GAN, WGAN

Question 1

a. For the case where $\theta \neq 0$:

i. **Kullback-Leibler (KL) Divergence is given by:**

$$D_{KL}(p\|q) = \iint_{\substack{x,y\in P \\ x,y\in Q}} p(x,y) \cdot log\left(\frac{p(x,y)}{q(x,y)}\right) dxdy$$

In our scenario, P is supported on $\{(0,y): y \in [0,1]\}$, while Q is supported on $\{(\theta,y): y \in [0,1]\}$. Since the supports of P and Q are disjoint when $\theta \neq 0$, for any point in the support of P, Q has zero probability, and vice versa. Since the KL divergence involves taking the logarithm of the ratio of the probabilities, and the denominator will be zero for all terms, the KL divergence is infinite in this case. This is consistent with the interpretation of KL divergence, which is not defined for distributions with disjoint supports.
Since in both cases y is uniformly distributed in $[0,1]$, we can say that $p(0,y) = 1$, thus:

$$D_{KL}(p\|q) = \int_{\substack{x,y\in P \\ x,y\in Q}} p(0,y) \cdot \log\left(\frac{2 \cdot p(0,y)}{p(0,y) + q(\theta,y)}\right) dy = \int_0^1 1 \cdot \log\left(\frac{1}{0}\right) dy = \infty$$

ii. **Jensen-Shannon (JS) Divergence**

Denote $M \triangleq \frac{p+q}{2}$

Since, the KL divergence is infinite in this case, the JS divergence will also be infinite. Hence,

$$D_{JS}(p\|q) = \frac{1}{2} D_{KL}(p\|M) + \frac{1}{2} D_{KL}(q\|M) = \infty$$

iii. **Wasserstein Distance**

$$W(p,q) = \inf_{\gamma\in\Pi(P,Q)} \int \|x-y\|^2 d\gamma$$

In this particular case, the x-coordinate is 0 for P and θ for Q, and y follows the same distribution for both P and Q. Therefore, the Wasserstein distance simplifies to:

$$W(p,q) = |0-\theta| = |\theta|$$

b. For the case where $\theta = 0$:

i. KL Divergence

$$D_{KL}(p\|q) = \iint_{\substack{x,y\in P \\ x,y\in Q}} p(x,y)\, log\left(\frac{p(x,y)}{q(x,y)}\right) dxdy$$

Since $x = 0 \quad \forall (x,y) \in P$, and $x = \theta = 0 \quad \forall (x,y) \in Q$ we have:

$$P(x,y) = P(0,y) = 1, \qquad Q(x,y) = Q(0,y)$$

so that, in both cases y is uniformly distributed in [0,1], and we get:

$$D_{KL}(p\|q) = \int_{\substack{y\in P \\ y\in Q}} p(0,y)\cdot log\left(\frac{p(0,y)}{q(0,y)}\right) dy = \int_0^1 1\cdot log\left(\frac{1}{1}\right) dy = 0$$

ii. Jensen-Shannon (JS) Divergence

Similarly to the calculation above, we have

$$\begin{aligned} D_{JS}(p\|q) &= \frac{1}{2} D_{KL}\left(p\|\frac{p+q}{2}\right) + \frac{1}{2} D_{KL}\left(q\|\frac{p+q}{2}\right) \\ &= \frac{1}{2}\int_{\substack{x,y\in P \\ x,y\in Q}} p(0,y)\log\left(\frac{2\cdot p(0,y)}{p(0,y)+q(0,y)}\right) dy \\ &+ \frac{1}{2}\int_{\substack{x,y\in P \\ x,y\in Q}} q(\theta,y)\log\left(\frac{2\cdot q(\theta,y)}{q(\theta,y)+p(0,y)}\right) dy \\ &= \frac{1}{2}\int_{\substack{y\in P \\ y\in Q}} 1\cdot\log\left(\frac{2\cdot 1}{2}\right) dy + \frac{1}{2}\int_{\substack{y\in P \\ y\in Q}} p(0,y)\cdot\log\left(\frac{2\cdot 1}{2}\right) dy = 0 \end{aligned}$$

iii. Wasserstein Distance

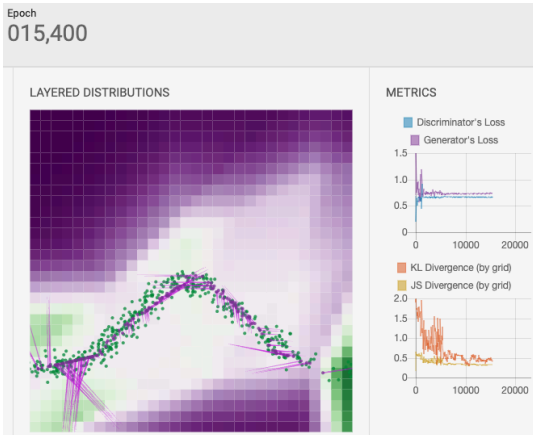
The Wasserstein distance, measures the minimum cost of transforming one distribution into another. In this case, since $x = 0 \quad \forall (x,y) \in P$, and $x = \theta \quad \forall (x,y) \in Q$, the Wasserstein distance can be calculated as the absolute difference between the two values of y when $x = \theta$ for the distributions P and Q, hence:

$$W(p,q) = |0-\theta| = \theta = 0$$

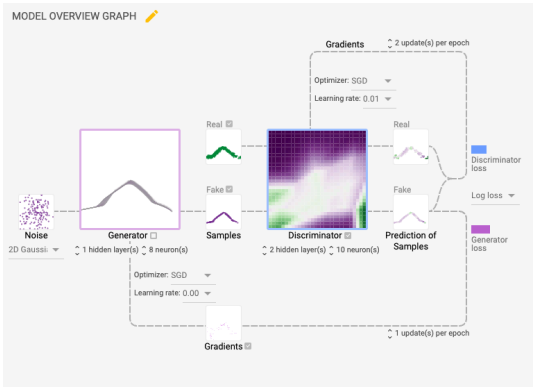
c. From the above results, we see that the Wasserstein distance is finite and meaningful even when the distributions P and Q have disjointed supports. In contrast, KL and JS divergences are undefined (infinity) when the supports of P and Q are disjoint. This means that Wasserstein distance can provide a sensible measure of the distance between distributions regardless of whether their supports overlap or not, making it more robust and useful in practice, especially when comparing distributions with different supports.

Question 2

Following several experiments, we observed varied outcomes - some converged while others did not. We decided to test with a sample distribution that wasn't linearly distributed or linearly separable. We found that the more complex the distribution, the harder it seemed for the generator to achieve convergence.



In our final experiment, we managed to train a GAN that could successfully generate synthetic samples that closely resembled the real data. The overlap between the fake and real samples was quite impressive. From our results, it was evident that areas distant from the real samples were more likely to consist of fake samples, as indicated by a darker shade of purple, which was in line with our expectations. Conversely, as we moved closer to regions populated by real samples, the heatmap became lighter in color. This suggests that the discriminator struggled to differentiate between the fake and real samples.



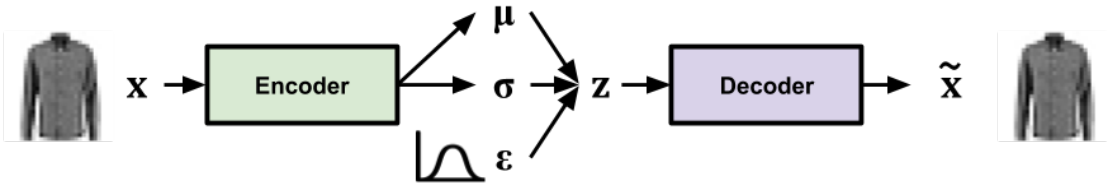
Concerning the gradients, we noticed that in regions with a green backdrop, the generator was still attempting to adjust according to the discriminator's current classification map. Meanwhile, in the white regions (where the green background is absent), the gradients were minimal or even nonexistent.

Question 3

In the paper *"Semi-supervised Learning with Deep Generative Models"* The authors suggest to construct a deep generative model (i.e., M1 Algorithm) of the data that is able to provide a more robust set of latent features, instead of a linear embedding, or features obtained from a regular auto-encoder.

Embeddings allow for a clustering of related observations in a latent feature space that allows for accurate classification, even with a limited number of labels.

So, in this exercise, we used the M1 Algorithm to create a robust set of latent features and then test it with an SVM classification model, in order to compare our results to the paper's results.



The image was taken from here.

In the following table and graph we represent a comparison of the papers results.

Paper Results*
MNIST

Table 1: Benchmark results of semi-supervised classification on MNIST with few labels.

N	NN	CNN	TSVM	CAE	MTC	AtlasRBF	M1+TSVM	M2	M1+M2
100	25.81	22.98	16.81	13.47	12.03	8.10 (± 0.95)	11.82 (± 0.25)	11.97 (± 1.71)	**3.33** (± 0.14)
600	11.44	7.68	6.16	6.3	5.13	–	5.72 (± 0.049)	4.94 (± 0.13)	**2.59** (± 0.05)
1000	10.7	6.45	5.38	4.77	3.64	3.68 (± 0.12)	4.24 (± 0.07)	3.60 (± 0.56)	**2.40** (± 0.02)
3000	6.04	3.35	3.45	3.22	2.57	–	3.49 (± 0.04)	3.92 (± 0.63)	**2.18** (± 0.04)

* The table was taken from the original paper.

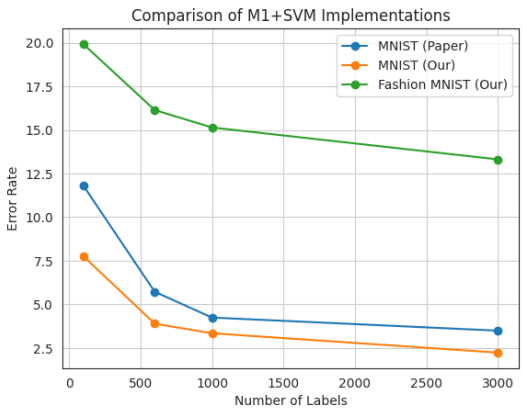
Our Results
MNIST

Labels	Error Rate
100	7.77
600	3.89
1000	3.34
3000	2.23

Our Results
Fashion MNIST

Labels	Error Rate
100	19.93
600	16.14
1000	15.14
3000	13.31

Comparision on Graph



In our experiment we worked with the RBF (Radial Basis Function) Kernel function, which defined as:

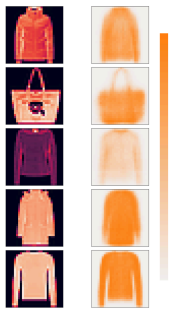
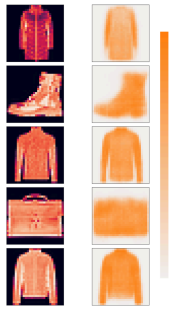
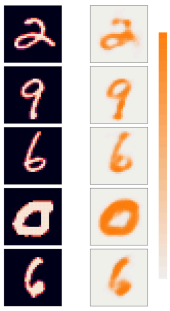
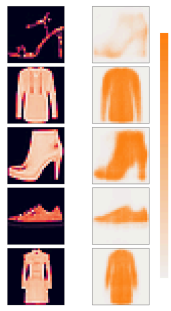
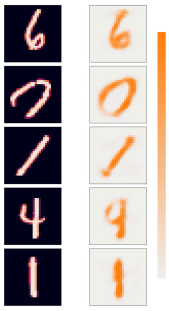
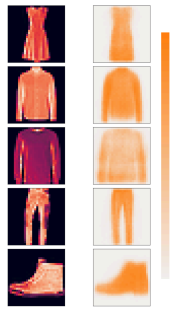
$$K(x,\hat{x}) = e^{\gamma\|x-\hat{x}\|^2}, \qquad \gamma = \frac{1}{N_{features} * Var(x)}$$

The error rates for the Fashion MNIST dataset are consistently higher across all label counts compared to the paper's reported results.

This suggests that the M1+TSVM implementation on the Fashion MNIST dataset may not perform as well as the original implementation on the MNIST dataset, as outlined in the paper. The error rates for the Fashion MNIST dataset are notably higher, indicating a lower accuracy of the model in distinguishing between different clothing items.

To ensure that the higher error rates obtained in the Fashion MNIST experiment were not solely due to implementation issues, we conducted additional tests on the MNIST dataset. The results of this analysis were encouraging, as they demonstrated error rates that were slightly better than those reported in the paper. These findings provide some validation for the reliability of our implementation.

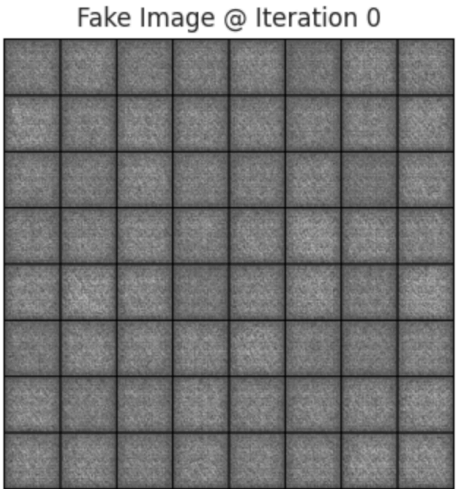
The purpose of the following captures is to demonstrate the impact of varying label amounts during training. To achieve this, we randomly generated five distinct labels and compared the original samples with those reconstructed from the VAE (Variational Autoencoder) latent features. These samples were obtained/generated during the 5th epoch of the training process. However, it's important to note that the first epoch produced similar noisy samples, as depicted in the image on the right.

Labels	MNIST	Fashion MNIST
100		
600		
1000		
3000		

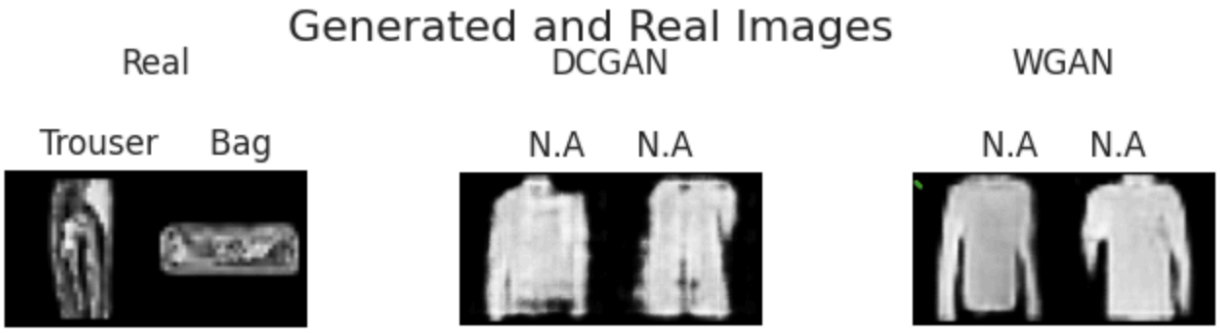
This is another illustration that help us to understand how the number of labels affect the results.

Question 4

In both GAN Methods we started the train with a noise as shown in the grid image that illustrate 64 noise-images.

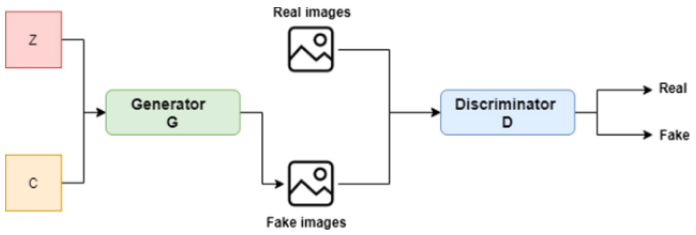


And finally, we've got the following results.



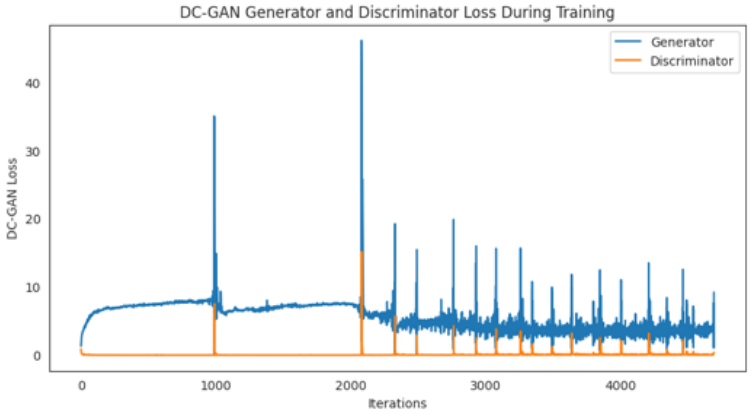
In our implementation of WGAN, we chose the simpler weight clipping method known as WGAN-CP. By using this approach, we observed faster convergence of the networks. After only 1000 iterations, we began to see distinct clothing items, whereas the basic DC-GAN required around 4000 iterations to exhibit clear parts. Furthermore, the WGAN produced smoother results compared to the basic DC-GAN.

During the training sessions, we noticed that the loss function of the DC-GAN diverged in approximately 1 out of 2 (or 3) sessions, resulting in unclear results that necessitated retraining. This divergence in the loss function could potentially be attributed to gradient vanishing, as discussed in the paper *"Improved Training of Wasserstein GANs."* Conversely, the WGAN consistently demonstrated convergence in nearly every training session. Additionally, we observed some fluctuations in the loss functions, indicating the model's ongoing attempts to improve itself.

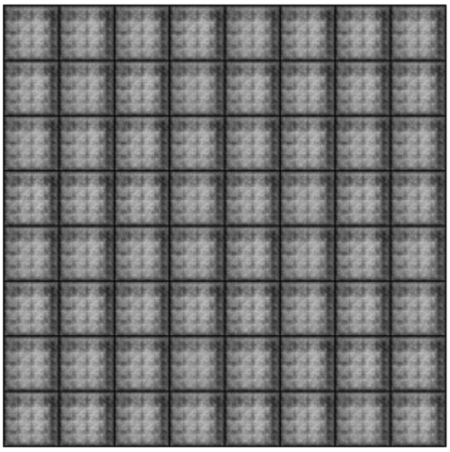


The image was taken from here.

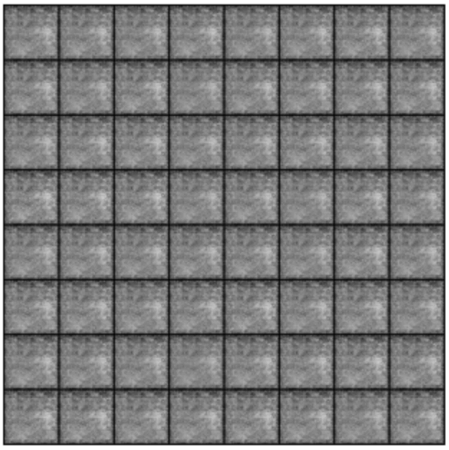
DC-GAN



Fake Image @ Iteration 1000



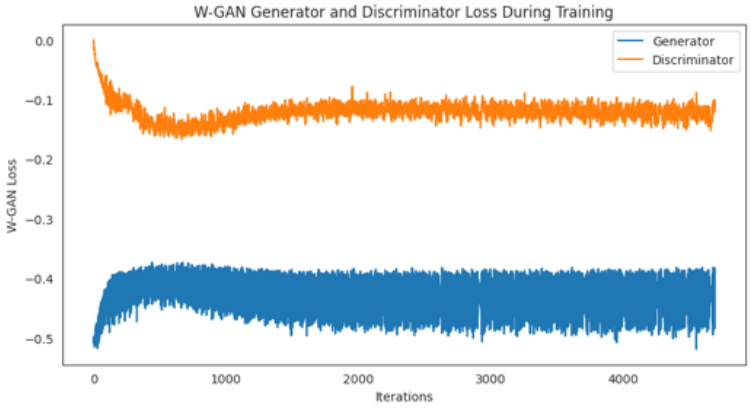
Fake Image @ Iteration 2000



Fake Image @ Iteration 3000

Fake Image @ Iteration 4000

WGAN-CP ("Weights Clipping")



Fake Image @ Iteration 1000

Fake Image @ Iteration 2000

Fake Image @ Iteration 3000

Fake Image @ Iteration 4000

References

[1] Semi-supervised Learning with Deep Generative Models

[2] WGAN-CP paper

[3] WGAN-GP paper

[4] DCGAN paper

[5] GAN — Wasserstein GAN & WGAN-GP. Training GAN is hard. Models may never… | by Jonathan Hui | Medium

[6] Common Training Loss Curve of DCGAN and WGAN - CV Notes (ddlee.cc)

[7] PyTorch-Wasserstein GAN(WGAN) | Kaggle