EX 3 – VAE, GAN, WGAN

Nimrod Curtis, 311230924, nimrodcurtis@mail.tau.ac.il
Alon Mizrahi, 312284706, alonmizrahi2@mail.tau.ac.il

Spring 2023

1 Theory

1.1 Q1

Supposed the given distributions in the assignment - P and Q. The Distance between the distributions calculated for the case of $\theta \neq 0$ and $\theta = 0$ using the KL-divergence, JS-divergence and the Wasserstein distance as follow:

1. $\theta \neq 0$
   (a) $D_{kl}(P||Q) = \sum_{x=0,y\ U(0,1)} 1 \cdot log(1/0) = \infty$
   (b) $D_{JS}(P||Q) = 0.5 \cdot (\sum_{x=0,y\ U(0,1)} 1 \cdot log(\frac{1}{0.5}) + \sum_{x=0,y\ U(0,1)} 1 \cdot log\frac{1}{0.5}) = log(2)$
   (c) $W(P,Q) = |\theta|$
2. $\theta = 0$
   (a) $D_{kl}(P||Q) = 0$
   (b) $D_{JS}(P||Q) = 0$
   (c) $W(P,Q) = 0$
3. The Wasserstein distance is continuous with respect to the underlying probability distributions. This property ensures that small changes in the distributions lead to small changes in the computed distance. In contrast, KL divergence and JS divergence can be sensitive to small changes, potentially leading to instability in certain scenarios such as our case when Kullback–Leibler divergence is exploding to infinity when no overlap between P, Q.

2 Practical

2.1 Q2

In a Generative Adversarial Network (GAN), the convergence of the generator and discriminator refers to the point where both components reach an equilibrium or stability in their training process. The goal of a GAN is to train a generator network to generate realistic data samples that resemble the training data distribution, while a discriminator network is simultaneously trained to distinguish between real and generated samples. During training, the generator and discriminator networks engage in a competitive game. The generator aims to produce samples that can fool the discriminator into classifying them as real, while the discriminator aims to accurately distinguish between real and generated samples. Through this iterative process, both networks learn and improve their performance. In Figure 1 you can see the convergence of the network that starts already after 5000 epochs. In addition, it can be seen that after 9000 epochs, the KL and JS graphs also converge and reach roughly the same value. In the figure, the purple dots represent the output from the generator, while the green dots represent the ground truth. You can see the similarity and the creation of the circle in both cases.

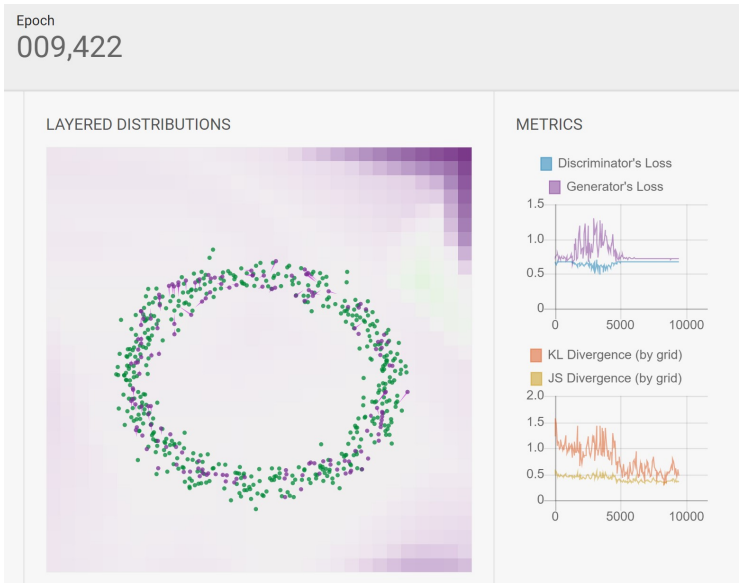


Figure 1: Online GAN results

2.2 Q3 - Variational Autoencoder + SVM

In this question we will implement VAE + SVM, according to the M1 model from the paper "Semi-supervised Learning with Deep Generative Models", by Kingsma et al. We will do this on the Fashion MNIST data set. The VAE is structured as follows: Encoder is composed of sequential two layers of size 600 each, following by a Relu. We also include batch norm after the first layer. Decoder is also composed of two layer with size 600 each, following by a Relu

activation. After the final layer we used sigmoid. Latent vector size of 50. We defined the loss as binary cross entropy with sum reduction for reconstruction loss and KL divergence term. batch size of 100, Adam optimizer and $3e^{-}4$ learning rate. As defined in the paper. for classification we used SVM from sklearn with RBF as the kernel.

You can see in the comparison table that better results were obtained in the article, we will note that in the article they performed TSVM while we performed only SVM, and another big difference is that in the article they worked on the Numbers MNIST dataset while we worked on the Fashion MNIST dataset.

Amount of labels	M1 + TSVM (Paper)	M1 + SVM (our)
100	0.882	0.790
600	0.945	0.823
1000	0.958	0.831
3000	0.965	0.843

Table 1: Convergence results Paper vs our

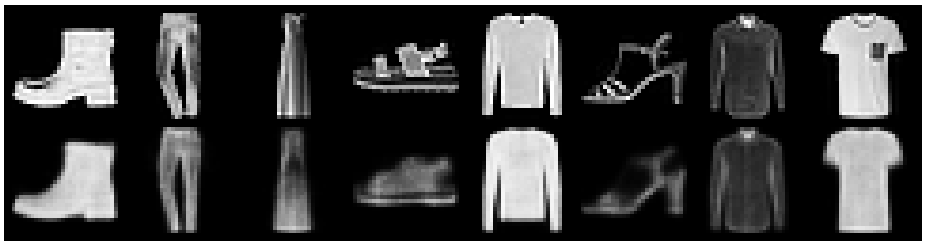


Figure 2: Recunstuction examples

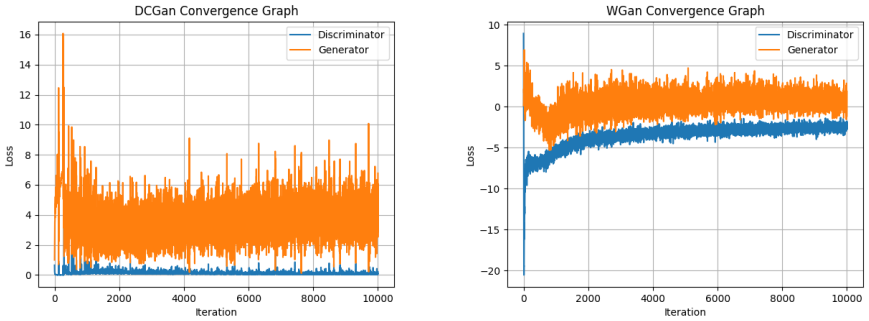
2.3 Q4 - DCGAN, WGAN

In this section, we conducted experiments using different GAN architectures: DCGAN and WGAN with Gradient Penalty. We followed the architecture described in "Improved Training of Wasserstein GANs" by Gulrajani et al. and "Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks".

The generator in DCGAN consists of two sequential blocks of Convolutional Transpose, followed by Batch Normalization and ReLU activation. It also includes an initial input block of Fully Connected layer, Batch Normalization, and ReLU activation. The complete architecture can be seen in the attached code. On the other hand, the DCGAN discriminator is composed of three sequential 2D Convolutional layers with LeakyReLU activation and Batch Normalization. The WGAN discriminator, however, does not include batch normalization layers.

After several design tests, we achieved satisfactory convergence graphs. For the DCGAN, we used (0.5, 0.999) as the beta values, a learning rate of 2×10^{-4}, and 1 iteration of the Critic. For the WGAN, we used (0.5, 0.9) as the beta values, a learning rate of 1×10^{-4}, and 5 iterations of the Critic.

The convergence graph is shown in Figure 3, and the visual outputs can be seen in Figure 4. From our experiments, we observed that the WGAN demonstrated more stable convergence compared to the DCGAN. When visually comparing the generated results, both GAN architectures managed to produce satisfactory results for certain classes.



(a) (b)

Figure 3: (a) DCGan convergence, (b) WGan convergence.

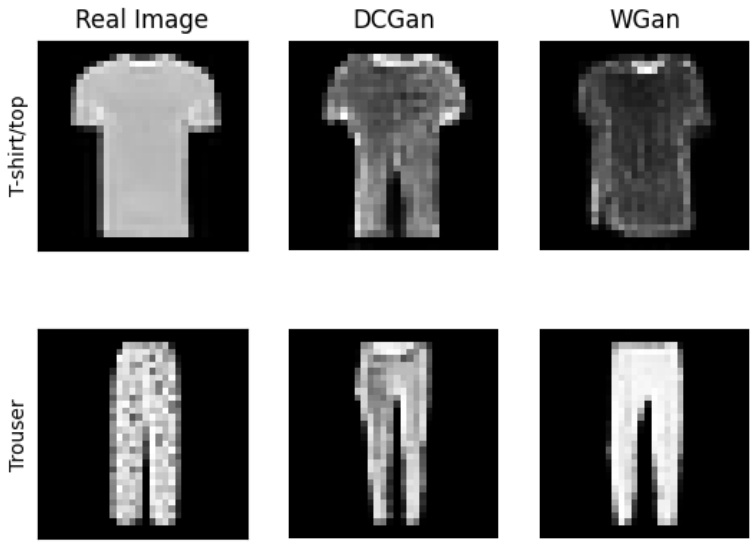


Figure 4: Comparison between WGAN and DCGAN generated images to true labels