*EX3 – Deep Learning*

*Osher Shukrun 316017763 | Moriya Mimran 305480071*

*Theory*

# Q1

1. In the case , the Kullback-Leibler Divergence becomes infinite since ’s support is not contained within ’s support.

This can be seen in the formula, along the line on which is nonzero (the line , ):

As for the Jensen-Shannon Divergence, it won’t be infinite. We’ll calculate it by definition. In the region where is nonzero, this time, we get the following:

And due to symmetry, we get that:

All in all,

Finally, for the Wasserstein Distance, we’ll find the minimal cost of “moving” to become .

For each point in , the closest point in will be , and the distance between them will be . This is the minimal cost of “moving” the point in to . Therefore, since is constant in our problem:

And this is the Wasserstein Distance between the two distributions.

1. When , the two distributions become identical. Therefore,

And, since ,

Also, as in item (a) we have proven that , we get that:

1. The advantage of the Wasserstein Distance over KL and JS is that it not only shows that the two distributions are indeed different (or “distant”), but it also shows *how* distant they are from each other.

For every we have received that and – they are constant, whether (and the two distributions are extremely close) or (and the two distributions are extremely distant).

Meanwhile, the Wasserstein Distance “kept” the information of the distance between , and resulted in a distance being equal to , which represents how “geometrically” distant the two distributions are.

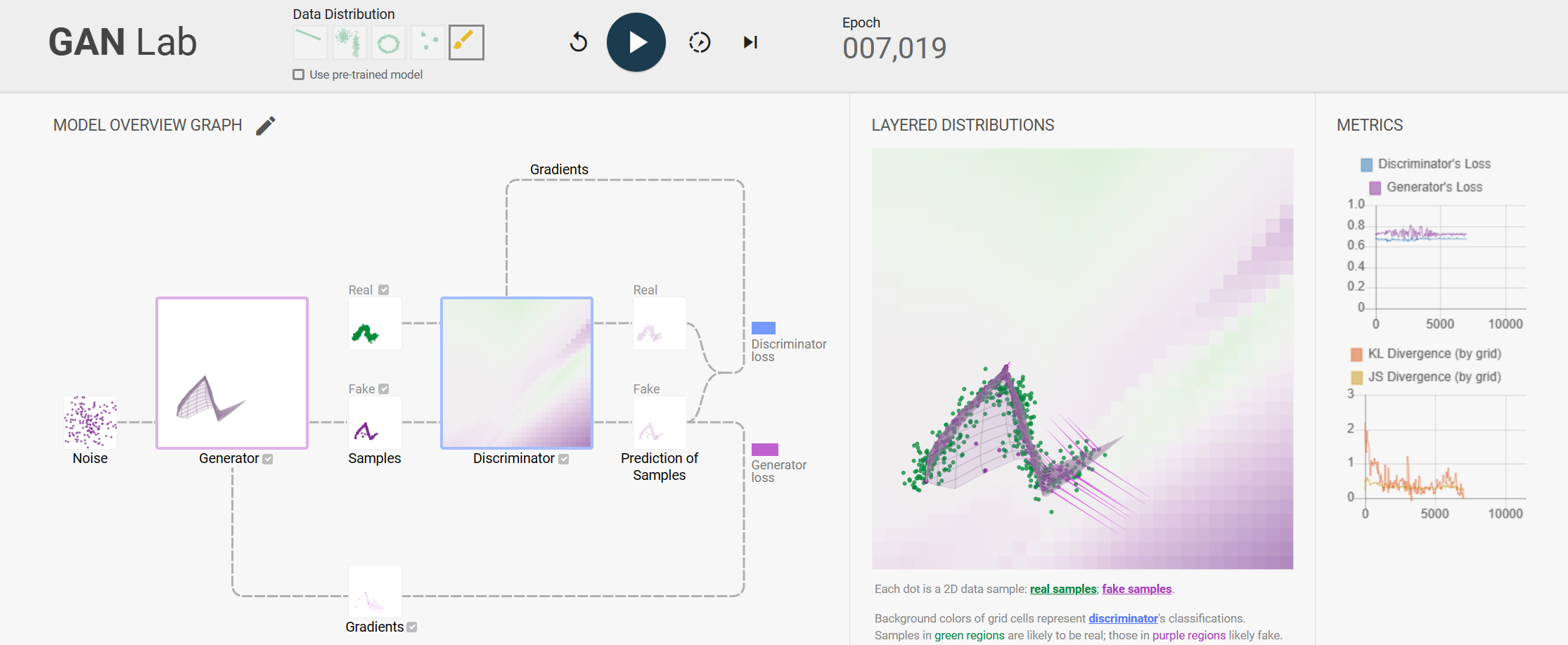
*Practical*

# Q2

The GAN (Generated Adversarial Networks) is a model that consists of two neural networks: a generator and a discriminator.

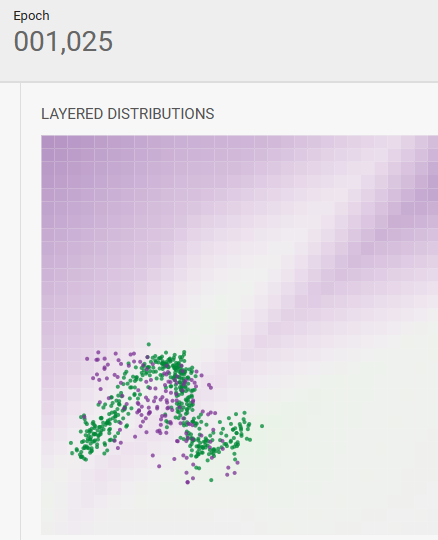
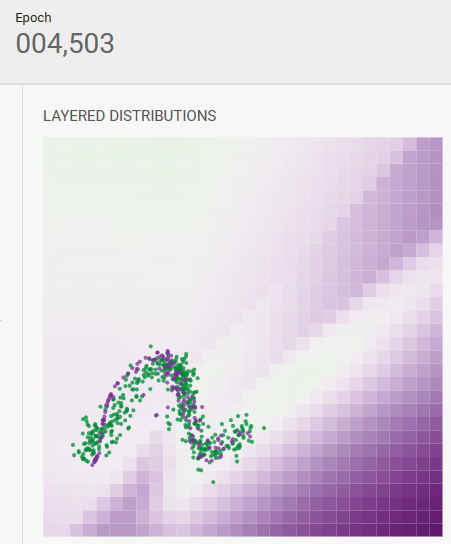
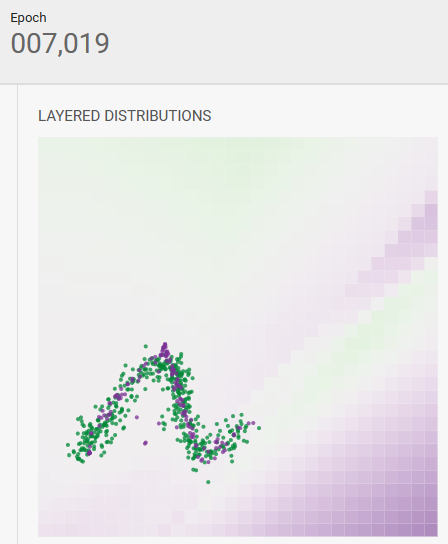
The **generator** takes a random noise vector, (noise can be either 1D or 2D and distribution type uniform or Gaussian), as input and transforms it into a fake sample (i.e., a multi-dimensional vector)

The **discriminator,** which is a binary classifier, determines whether it is a real sample or fake sample (represents the probability that the sample is real rather than fake).



During training, the generator and discriminator networks play a two-player minimax game. The generator tries to produce synthetic data that can fool the discriminator, while the discriminator aims to accurately classify the real and fake data samples. This adversarial process drives the networks to improve iteratively.

The discriminator’s performance can be interpreted through the ‘LAYERED DISTRIBUTIONS’:

Background colors of grid cells represent discriminator's classifications. Samples in green regions are likely to be real; those in purple regions are likely fake.

In the beginning of the training, distribution of fake samples and real samples do not match well. As the training progresses, the discriminator is performing well, since most real samples lie on its classification surface’s green region (and fake samples on purple region).

A screenshot of a graph

Description automatically generated with medium confidence

GAN Lab provides the most used approaches for comparing distributions, Kullback-Leibler (KL) and Jensen-Shannon (JS) divergence values by discretizing the 2D continuous space (via the grid). Used to evaluate the quality of generated samples. **These measures help assess how well a generated distribution matches the true distribution it aims to approximate.**

Formally, the KL divergence value is defined as:

JS divergence is defined as:

Were, ‘M’ representing the average distribution:

where is the probability density of the real samples in the i-th cell, calculated by dividing the number of real samples in the i-th cell by the total number of real samples; is similarly defined for the fake examples.

**The gradients** are used to update the weights of the network during the training process through a technique called gradient descent.

Gradients for generator are computed for each fake sample by backpropagating the generator’s loss through the graph.

A screenshot of a screen

Description automatically generated with low confidenceA screenshot of a computer screen

Description automatically generated with low confidence

This snapshot of gradients indicates how each sample should move to, in order to decrease the loss value. The fake sample’s gradients point to green areas, and so they will decrease as the training progresses.

# Q3

semi-supervised learning via a variational autoencoder

The idea of **Variational Autoencoder** (short for **VAE**), Instead of encoding the information into a vector, VAEs encode the information into a probability space, and map it into a distribution.

The **encoder model of a VAE** will output parameters describing a distribution for each dimension in the latent space. Since we're assuming that our prior follows a normal distribution, we'll output two vectors describing **the mean and variance of the latent state distributions**.

A picture containing text, diagram, line, font

Description automatically generated

The relationship between the data input  and the latent encoding vector  can be fully defined by:

* The prior is a spherical Gaussian distribution:

The prior represents the underlying distribution of all the data.

* Likelihood function:

is a suitable likelihood function (e.g., a Gaussian or Bernoulli distribution) whose probabilities are formed by a non-linear transformation, parameters of a set of latent variables z.

* Posterior

The posterior represents the conditional distribution given a particular data point, essentially, this is the distribution that the data point has been encoded as.

In our model, we introduce a fixed-form distribution with parameters - that approximates the true posterior distribution .

We then follow the variational principle to derive a lower bound on the marginal likelihood of the model – these bound forms our objective function and ensures that our approximate posterior is as close as possible to the true posterior (detailed below).

A picture containing text, screenshot, font, line

Description automatically generated

is a Gaussian inference network for the latent variable z:

The estimated posterior  should be very close to the real one . We can use [Kullback-Leibler divergence](https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence) to quantify the distance between these two distributions. measures how much information is lost if the distribution Y is used to represent X.

In our case we want to minimize  with respect to .

A picture containing text, font, handwriting, document

Description automatically generated

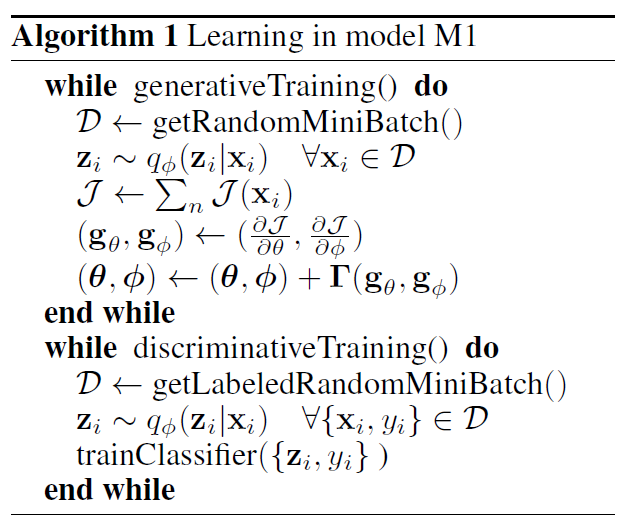
After rearranging the left- and right-hand side of the equation, we can define the loss function under the assumptions above:

when learning the true distributions: we want to maximize the (log-)likelihood of generating real data (that is and minimize the difference between the real and estimated posterior distributions (the term KL).

the variational bound on the marginal likelihood for a single data point is:

By minimizing the loss, we are maximizing the lower bound of the probability of generating real data samples.

The inference network is used during training of the model using both the labelled and unlabeled data sets.



Support Vector Machines (SVMs) are a popular choice for image classification tasks like MNIST and Fashion MNIST (which are grayscale images). When using SVMs, the choice of the kernel depends on the specific characteristics of the data and the problem at hand.

After several experiments with different kernels and hyperparameters to find the best combination for our specific task, it seems that **the Radial Basis Function (RBF) kernel** (also known as the Gaussian kernel) tends to perform the best.

A picture containing text, screenshot, number, font

Description automatically generated

The Fashion MNIST dataset is more challenging than MNIST because the images contain more complex patterns and textures compared to simple handwritten digits. Consequently, achieving the same level of accuracy on Fashion MNIST as on MNIST can be more difficult.

reference:

<https://github.com/screddy1313/VAE/blob/master/vae_mnist_cnn.ipynb>

# Q4

Wasserstein GANs architectures CIFAR10

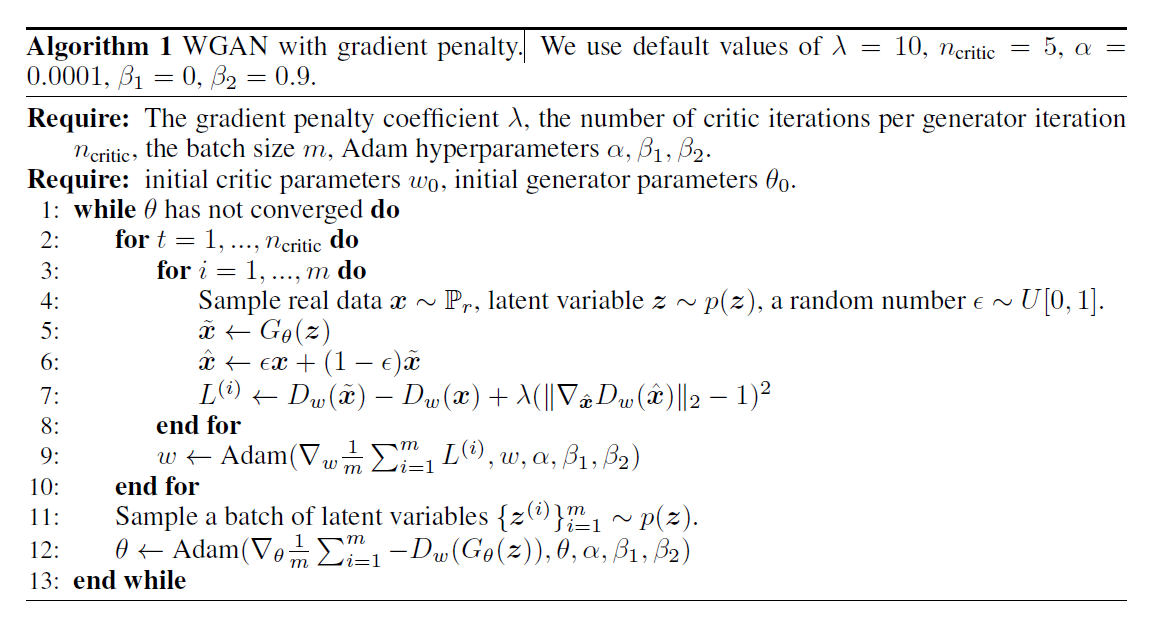
We implemented WGAN algorithm with **gradient penalty** architecture.

A picture containing text, font, screenshot, line

Description automatically generated

WGAN with GP is implemented by using the Wasserstein distance as the loss function in the WGAN framework, along with a gradient penalty term added to the discriminator's loss to enforce the Lipschitz constraint and improve stability during training.

The gradient penalty encourages the gradient of the discriminator's output with respect to the input samples to have a norm of 1. This penalty term helps to prevent the discriminator from becoming too powerful and improves training stability. The gradient penalty is typically calculated using the L2 norm of the gradients and added to the discriminator's loss.

****

implemented WGAN algorithm with **gradient penalty** architecture.

A picture containing text, font, screenshot, line

Description automatically generated

WGAN with GP is implemented by using the Wasserstein distance as the loss function in the WGAN framework, along with a gradient penalty term added to the discriminator's loss to enforce the Lipschitz constraint and improve stability during training.

The gradient penalty encourages the gradient of the discriminator's output with respect to the input samples to have a norm of 1. This penalty term helps to prevent the discriminator from becoming too powerful and improves training stability. The gradient penalty is typically calculated using the L2 norm of the gradients and added to the discriminator's loss.

**A math equations and formulas

Description automatically generated with medium confidence**

A picture containing text, screenshot, display, plot

Description automatically generated

A set of **real images** and their classification for comparison with the two popular variants of the generative adversarial network (GAN) architecture:

A picture containing screenshot

Description automatically generated

A picture containing screenshot

Description automatically generated

DCGAN is a well-known and widely used GAN variant.

We adapted the architecture to the requirements mentioned in the literature [1].

A picture containing text, screenshot, font, white

Description automatically generated

[1] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015.

Each of the GANs architecture (DCGAN and WGAN\_GP) consists of generator, discriminator and training functions separately. We train the models with the Adam optimizer with the same hyperparameters for both models (learning rate and betas).

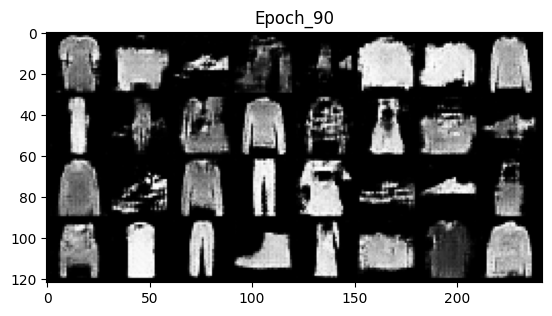
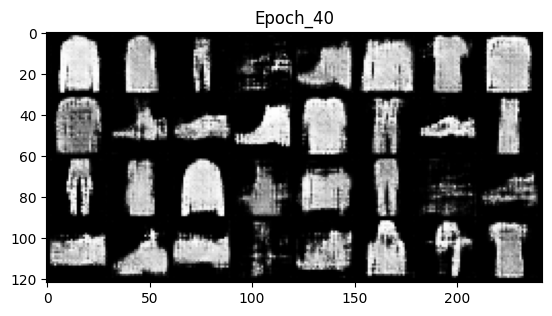
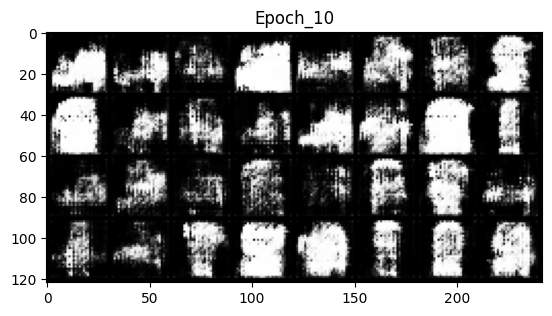
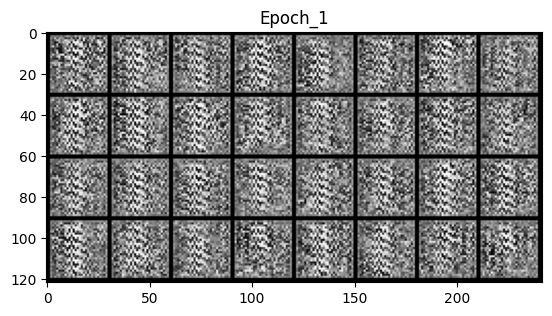
## Generated fake images during the training process after a number of epochs of training.

We can see that the success of convergence in GAN training can depend on various factors, such as the specific dataset, architecture choices, hyperparameters, and training procedures.

The number of failed convergences can vary greatly based on these factors and the specific implementation details.

When we experimented with different strategies, many iterations on the training process and hyperparameter changes, we could eventually lead to successful convergences in the GAN training of the models.

## WGAN\_GP

A picture containing screenshot, black and white, x-ray film, art

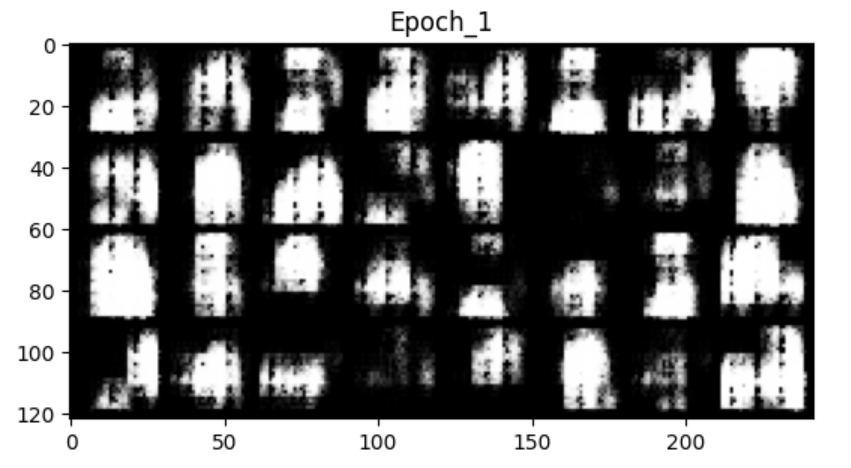
Description automatically generatedA collage of different types of clothes

Description automatically generated with low confidenceA picture containing text, screenshot, black and white, white

Description automatically generated

For the WGAN\_PD model starting from **~epoch95** we got clear images which were generated by the generator which look good. (Failed until **~epoch35**)

## DCGAN

A picture containing screenshot, text, black and white, black

Description automatically generatedA picture containing text, screenshot, black and white, black

Description automatically generatedA picture containing text, black and white, screenshot, white

Description automatically generatedA picture containing screenshot, white, black and white, text

Description automatically generatedA picture containing screenshot, black and white, white, appliance

Description automatically generatedA picture containing text, black and white, white, screenshot

Description automatically generated

For the DCGAN model starting from **~40epoch** we got clear images which were generated by the generator which look good. (Failed until **~20epoch**)