Assignment 4 (ANN & Deep learning)

Student: Afraz Salim *r-number: r0439731*

Restricted boltzman machine

Restricted boltzman machine is a probabilistic model which is used to learn the hidden representation of data. Given some input data, we want to learn the distribution of that data. RBM is an extended version of hopfield network with some restrictions on the connections.
In this section, we will analyze the effect of number of iterations and number of components in RBMs. More specifically, we will change these parameters and inspect if it has better effect on the result or not.

Number of components in RBM:Experimental analysis

We will first analyze the *components* parameter which represents number of hidden binary units. An exmaple of the training images are shown in figure 2.

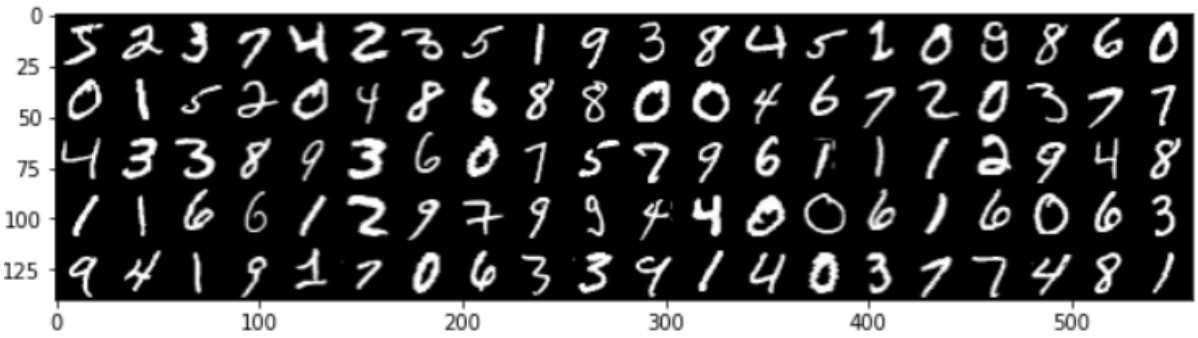
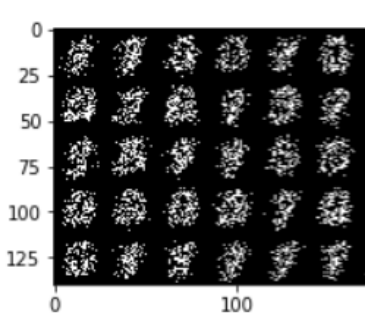


Figura 1: Training images

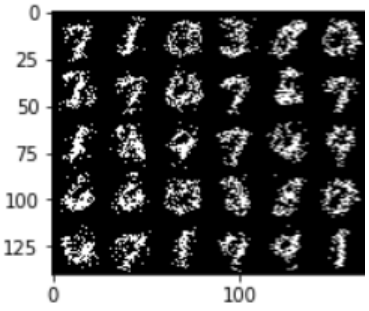
Figura 2: Test images

- What is the effect of different parameters (epochs and components) on training the RBM, evaluate the performance visually by reconstructing unseen test images. Change the number of Gibbs sampling steps, can you explain the result?

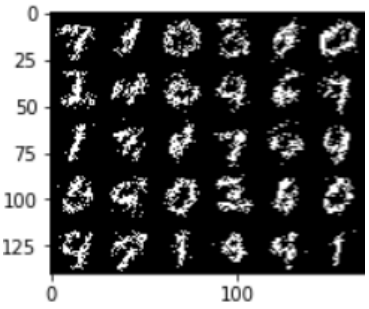
We fix all other parameters except components parameter. By changing the components parameter, we will obtain different results on the test data which we can compare with eachother. Figures on the following page show that by increasing the number of components, recalled pattern looks better.



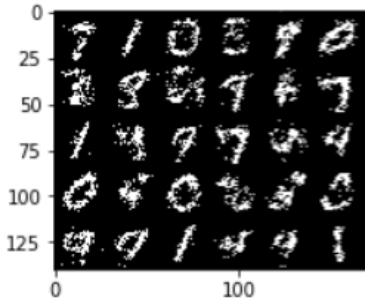
(a) Components = 10



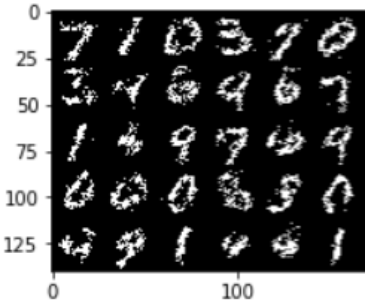
(b) Components = 20



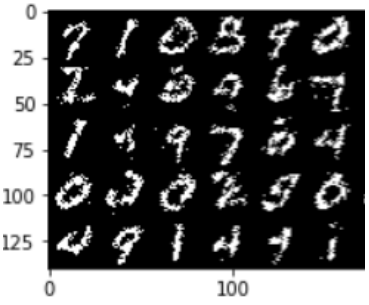
(c) Components = 30



(a) Components = 40



(b) Components = 50



(c) Components = 60

In short, by increasing the number of hidden units, we are increasing the capability of the model the to recall these patterns with higher probability.

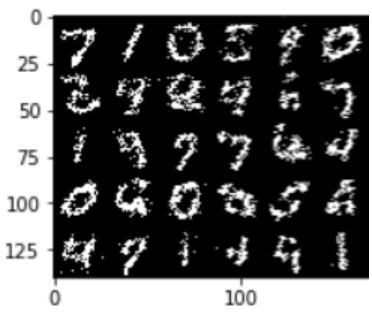
learning_rate	random_state	n_iter	Gibs sampling
0.01	0	10	10

If we have a model with only one hidden unit($k = 1$) and n visible units then we can store a pattern in 2^k different ways. However, by increasing the number of hidden units, we can store some pattern multiple times. If we store a pattern multiple times then it's probability of recalling gets higher.

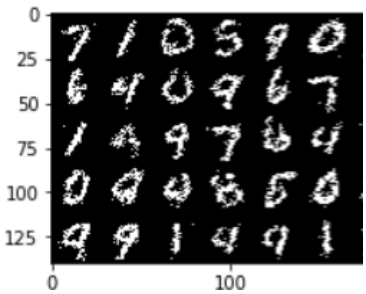
We now pick the image that we get with components= 60 and try to investigate the impact of the iteration parameter. In short, we have following parameters fixed.

learning_rate	random_state	components	Gibs sampling
0.01	0	40	10

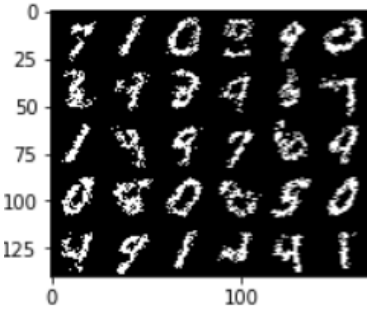
We can see from the following figure that increasing the iterations has not much effect on the result. Increasing the number of iterations makes few images slightly smoother but such increment had not much impact on the result.



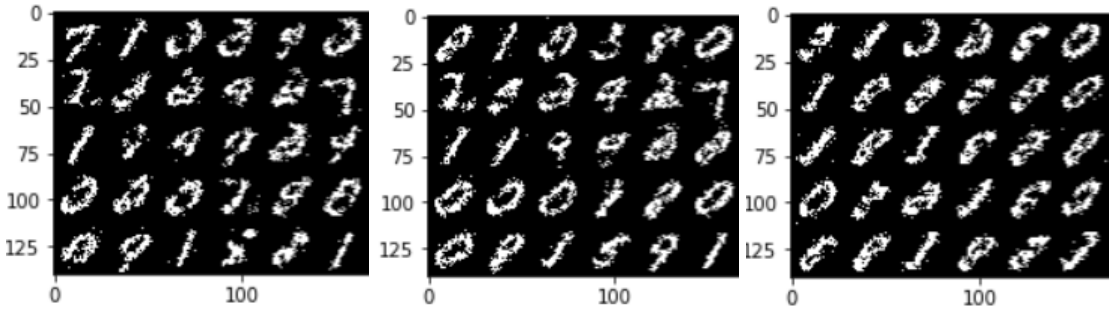
(a) Components = 60 iteration = 40



(b) Components = 60 iteration = 40



(c) Components = 60 iteration = 90



(a) Components = 60 Gibs = 30 (b) Components = 60 Gibs = 50 (c) Components = 60 Gibs = 300

Now again, we fix the parameter and see the effect of changing gibs sampling parameter. For Gibs sampling , we used 10 iterations and 60 components.
From the figure show above, we notice that if we increase the gibs sampling then the images get blur. Gibs sampling is used to estimate the joint distribtuion $P(v, h)$. However, taking too many samples may lead to same estimation for all visible and hidden units.

- Use the RBM to reconstruct missing parts of images1. What is the role of the number of hidden units, learning rate and number of iterations on the performance. What is the effect of removing more rows on the ability of the network to reconstruct? What if you remove rows on different locations (top, middle,...)?

As mentioned in the previous section, hidden units allow us to store some patterns multiple times. Since, we want the probability to be very high for few patterns. The probability of a pattern is determined by how many times such pattern occurs in a sample. Storing a pattern multiple times corresponds to increasing it's probability. Learning rate is same as in common neural network architecture. Large values of learning rates can result in unstable learning while very low rates can result in no training.
First we remove the first 12 rows of the the images and check if the model is able to reconstruct the original image. We can see in figure 7 that the model is able to reconstruct first 12 rows of the images.

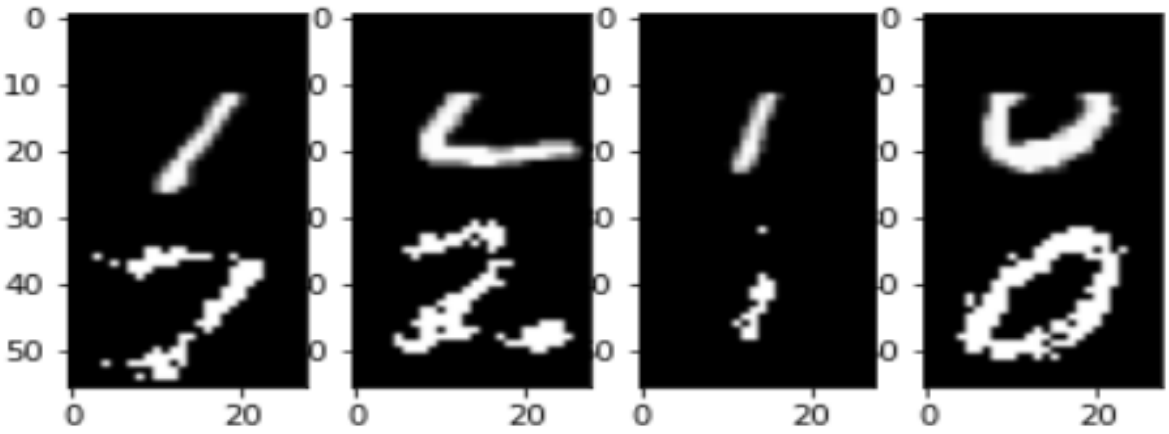


Figura 7: Reconstructed image

Now we remove the multiple rows and find out if the model can reconstruct the missing part.Rows that we remove are from $[0-10], [18-24], [28-30]$.
Figure 8 show the reconstructed images. We see that few only two images (number 7 and number 0) get reconstructed while the other numbers are not reconstructed.

- Load the pretrained DBM that is trained on the MNIST database. Show the filters (interconnection weights) extracted from the previously trained RBM (see exercise 1) and the DBM, what is the difference? Can you explain the difference between filters of the first and second layer of the DBM?

Filters from the simple RBM are shown in figure 9. Filters from deeper layers of DBM are shown in figure 10 and 11. First layer extract only sharp edges. The second layer shows that few images contain full digits.

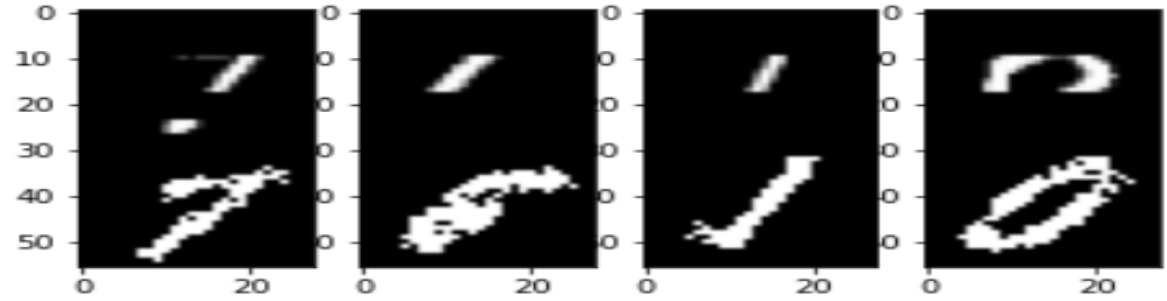


Figura 8: Reconstructed image

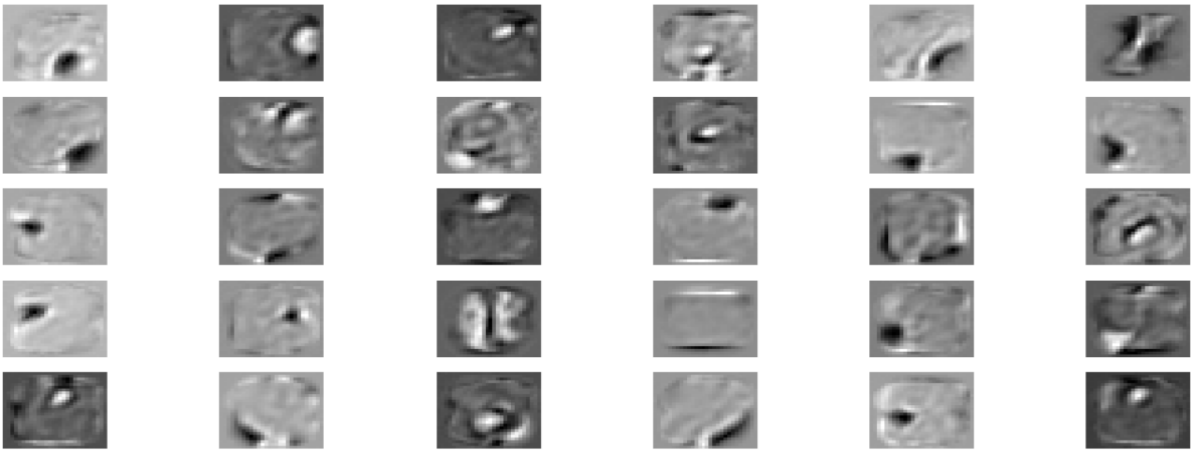


Figura 9: Filters or weights visualization: RBM

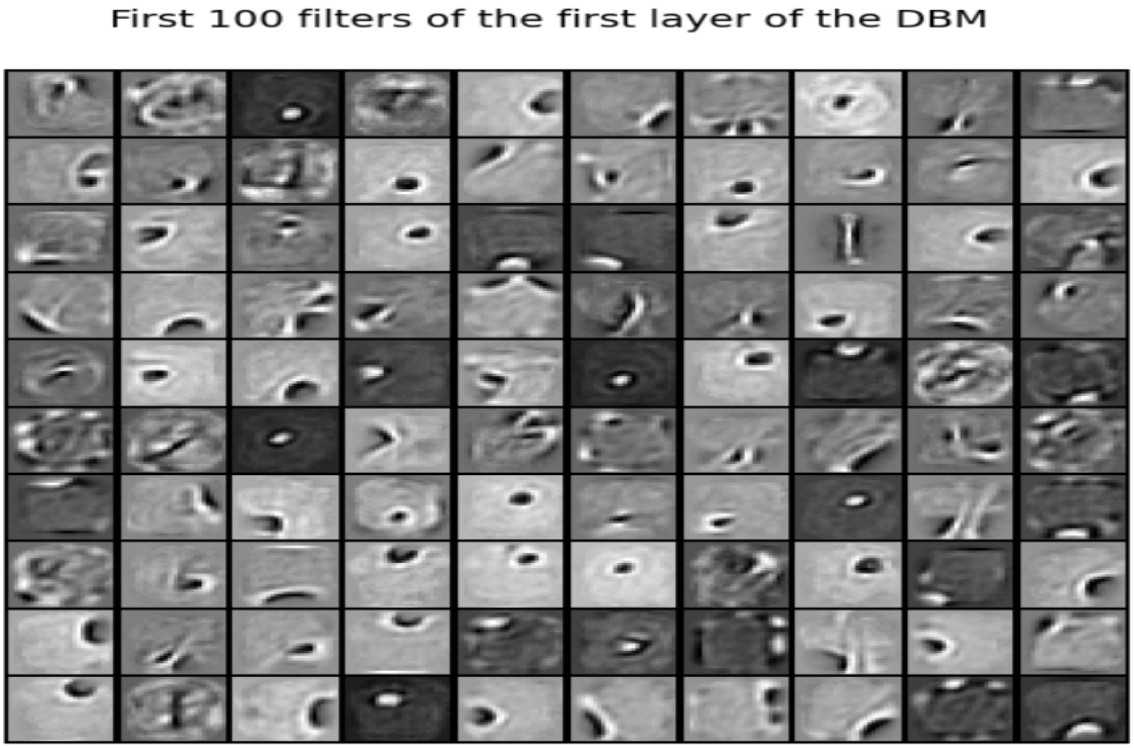


Figura 10: Filters from first layer of DBM

- Sample new images from the DBM. Is the quality better than the RBM from the previous exercise and explain why?

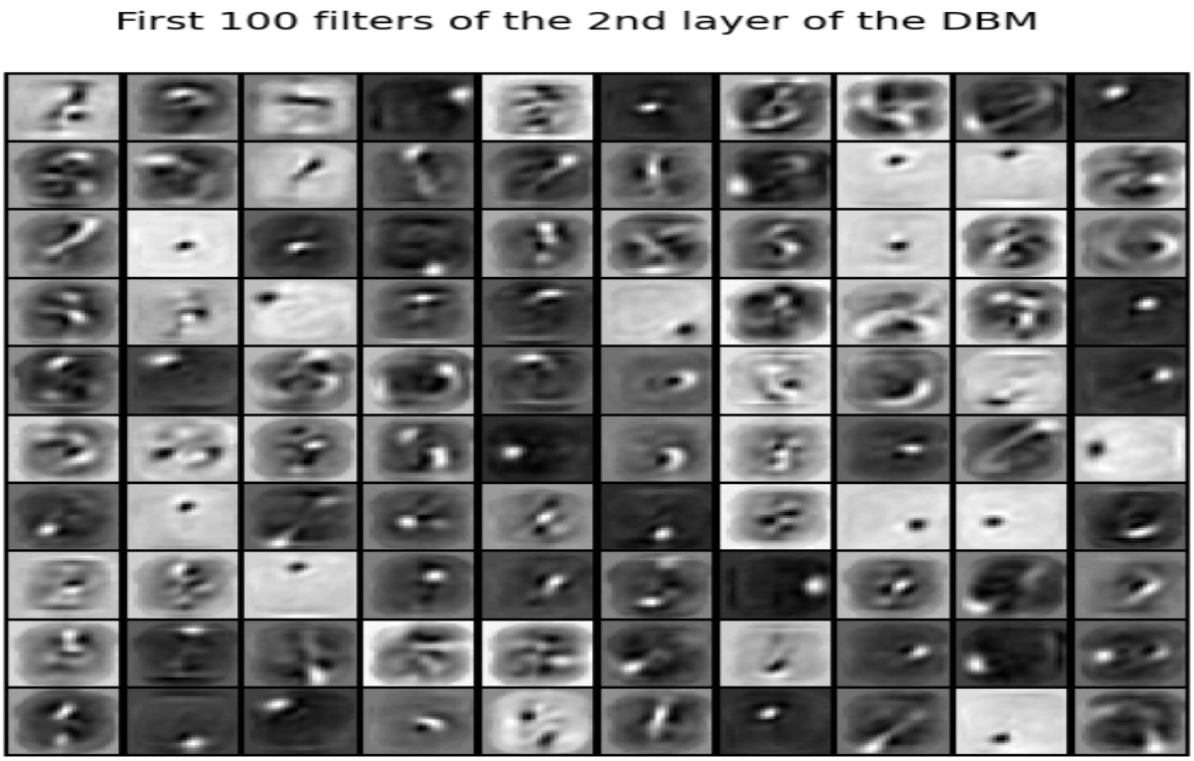
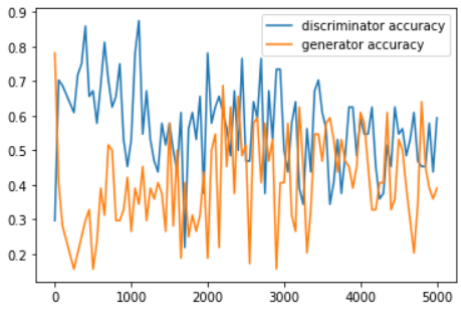


Figura 11: Filters from second layer of DBM

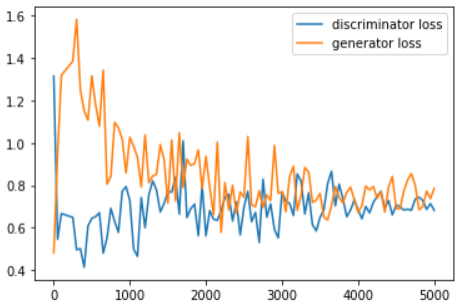
The quality of the images drawn from DBM is much better than those drawn from RBM. As we saw in previous section that the features in second section represent full images. Samples drawn from the RBM miss certain parts in the images. This is not the case with DBM as distribution of the features is captured in a much better way due to the second layer.

DCGAN

We will investigate the stability of the DCGAN by analyzing the loss and accuracy value that we obtain during the training of the model. First we run the model for long enough so that it is trained and is able to generate images. Two examples of generated images are shown in figure 13 and 14. We plot the loss and accuracy curves of generator and discriminator.



(a) Accuracy



(b) Loss

In order to make any statement about stability, we have to compare the loss figure with some other figure. If there is less zigzag pattern in the loss and accuracy curve, then we have a stable learning.

Optimal transport

Optimal transport, as the name suggest, is a way to transfer some data from a predefined distribution to a known distribution. The target distribution is simply a fixed and known shape that needs to be filled. In the context of GANs, discriminator tries to discriminate between real and fake data. On the other hand, generator tries to generate data that looks similar to real data. During this learning process,

Figura 13: Filters from first layer of DBM

Figura 14: Filters from first layer of DBM

the generator is actually moving it's distribution towards the distribution of the real data. However, if we use pixel by pixel transformation then independent of how different two distributions are, we have to move all pixels color. By using wassersteins distance, we are only trying to capture that part of the distribution which is not overlapping. Hence, we can found a transport plan that minimizes the transport cost. Figure 15 shows the results that we obtained using optimal transport.The pixels in images(labeled with reg) generated with Sinkhorn distance have same weight everywhere while simple those generated

with earth movers distance have light regions which indicates that some pixels have higher weight while others have lower weight. Next, we investigate the learning stability of WGAN, WGAN with gradient

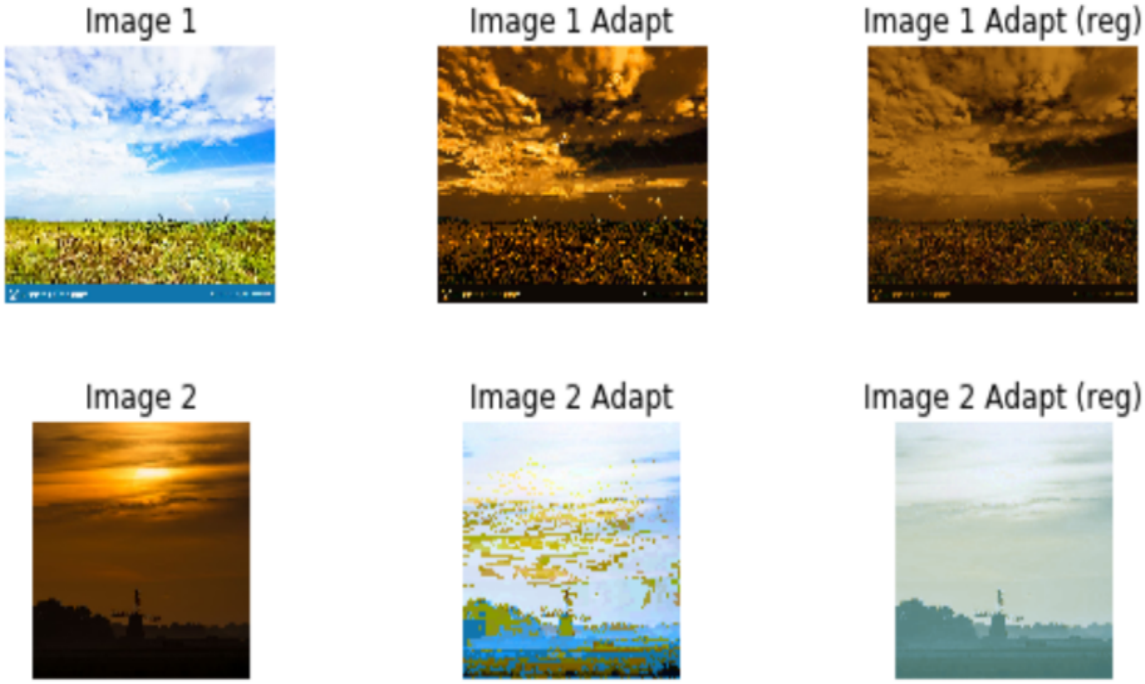
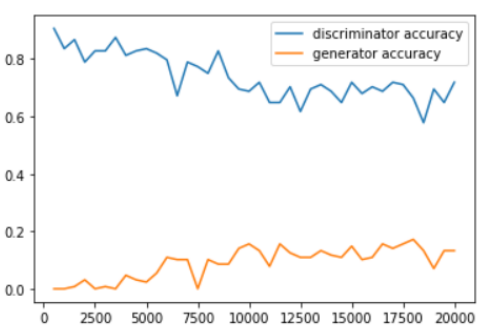
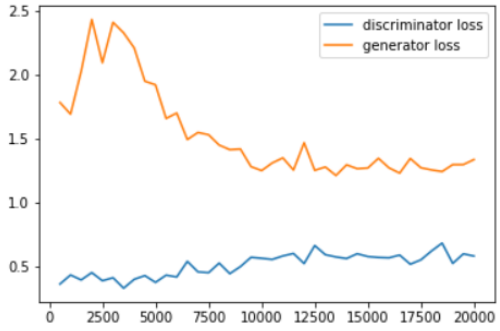


Figura 15: Results of using optimal transport

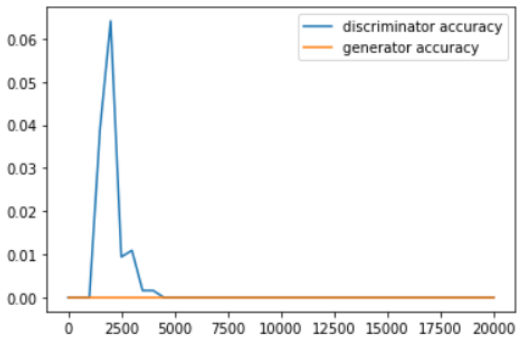
clipping and gradient penalty. Following figures show the loss and accuracy curves. From the first two



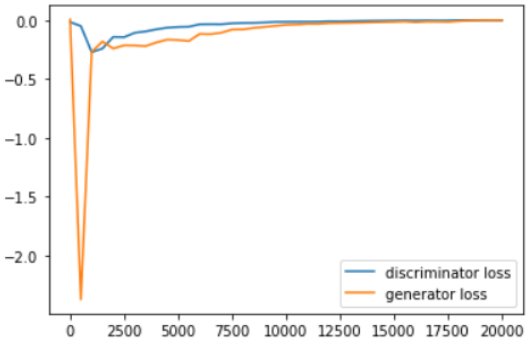
(a) Standard GAN accuracy



(b) Standard GAN loss



(a) WGAN with weight clipping: accuracy



(b) WGAN with weight clipping: loss

figures, we notice that the network converges but very slowly while the last two figures show very strange

behaviour.