# Intro to NN - ex3

Amit Segal , Oriyan Hermoni

## Practical part

### 1 *Basic Autoencoder*

Trained a basic AE model using the same architecture as as seen in the exercise PDF. The dimension of the latent vectors we chose for this architecture is 10 . We used tSNE to cluster the latent vectors as we achieved a much better visual representation of the clusters. We trained using an Adam optimizer with learning rate 0.01, batch size 32, and 10 epochs.

Figure 1: Dimension reduction of MNIST dataset latent vectors (in Z-space) to 2 dimensions using tSNE


### 2 *Denoising AE*

Trained a basic AE model using the same architecture as as seen in the exercise PDF. The dimension of the latent vectors we chose for this architecture is 10 . We used tSNE to cluster the latent vectors as we achieved a much better visual representation of the clusters. We trained using an Adam optimizer with learning rate 0.01, batch size 32, and 10 epochs.

We chose to add gaussian noise to the dataset, with different means to see the effect of the different levels of noise on a denoising AE. Our noise levels are $\Sigma = [0.05, 0.25, 0.7]$. As the noise increases, the less separation exists between the different clusters, and the more outliers each one contains. Instead of adding regularization terms to our loss, we've instead opted to tweak our model to use a leaky ReLU activation. We've also attached 3 samples from the test set. The images are set s.t. the leftmost image contains the original, clean image from MNIST test set, the middle image is the image after having random noise with the correspoinding sigma added to it, and the rightmost image is the image after denoising by training a decoder whose loss is the output's MSE from the clean image.

Figure 2: Dimension reduction of MNIST dataset latent vectors (in Z-space) to 2 dimensions using tSNE - $\sigma = 0.05$

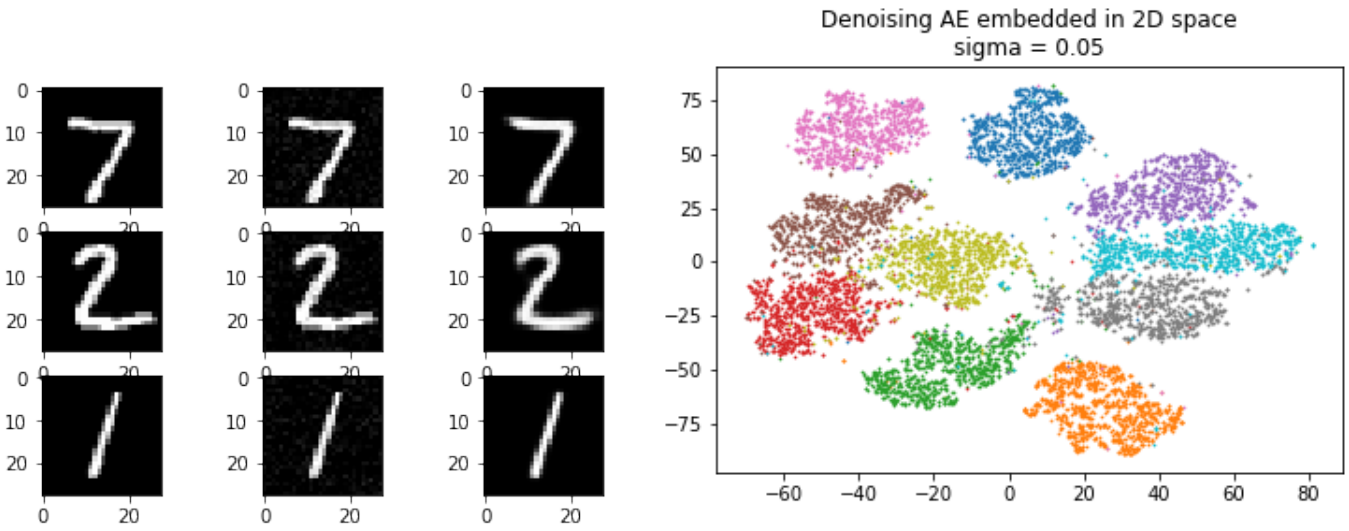



Figure 3: Dimension reduction of MNIST dataset latent vectors (in Z-space) to 2 dimensions using tSNE - $\sigma = 0.25$

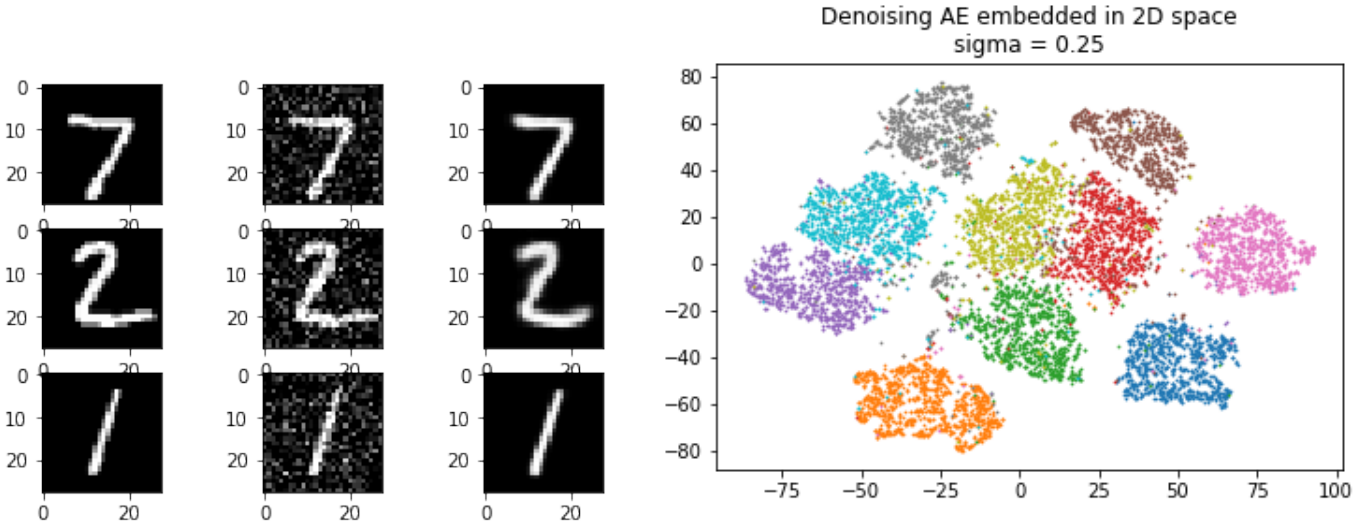
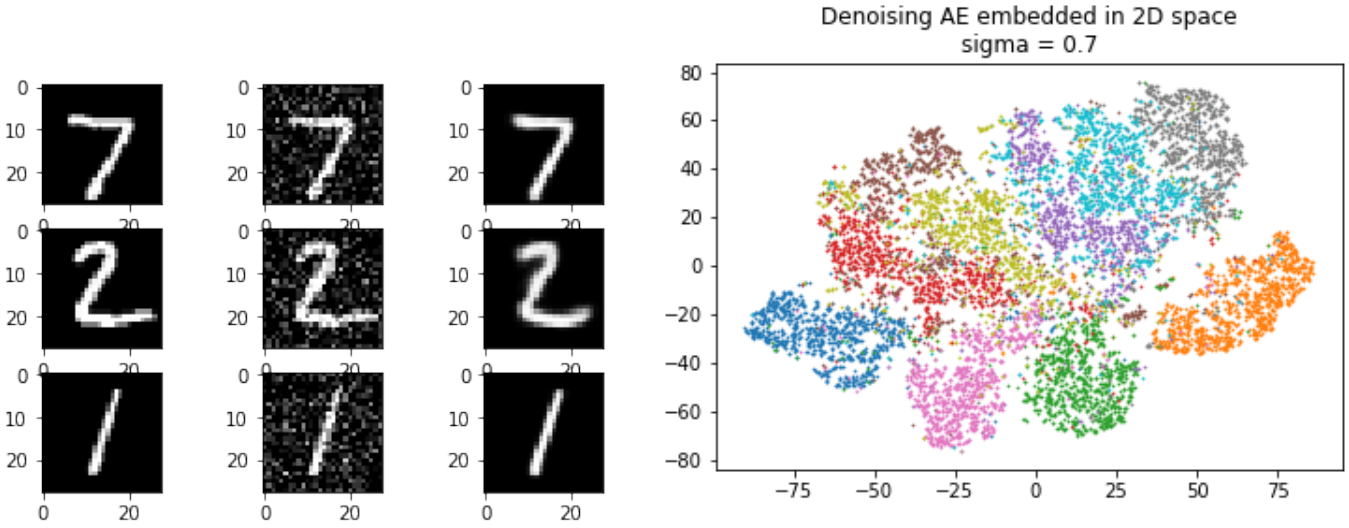



Figure 4: Dimension reduction of MNIST dataset latent vectors (in Z-space) to 2 dimensions using tSNE - $\sigma = 0.7$



## 3 *GAN*

Seeing as we've had trouble with the provided architectures, we eventually used an architecture from one of the GAN examples on Tensorflow's website. For our discriminator, we've used two convolutional layers with 64 and 128 channels and stride = 2 (filter size was 5x5) with leaky ReLU actiavtion, each followed by 30% dropout regularization, followed by a fully connected layer mapping to a single node. For our generator, we've used a fully connected layer mapping to (7, 7, 256) with leaky ReLU activation, followed by the transpose of the discriminator's convolutional layer, each with batch-normalized inputs to its activation, aside for the final layer which uses tanh activation. For our training we've used binary crossentropy loss with logits. Attached are our novel examples:

Figure 5: Novel samples generated by randomly selecting 16 samples $Z \sim N(0, I)$

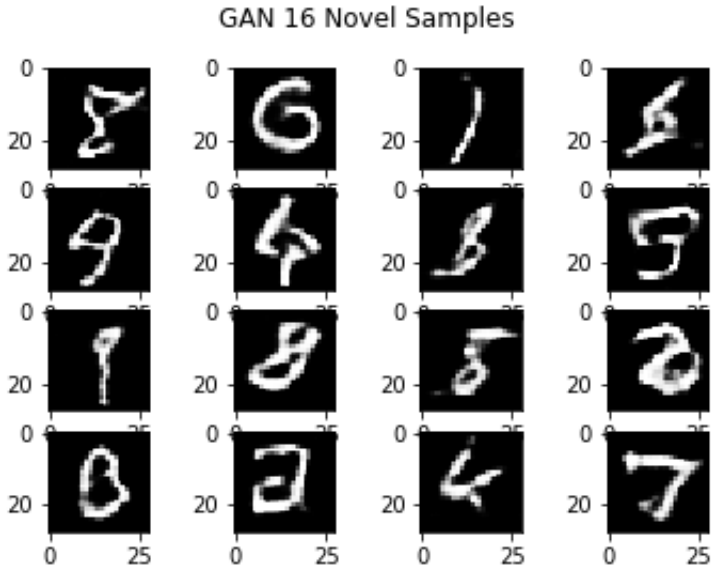
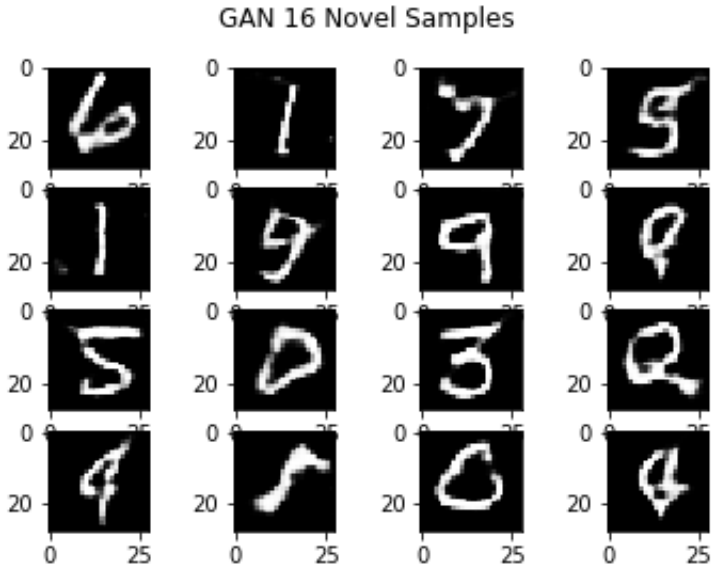


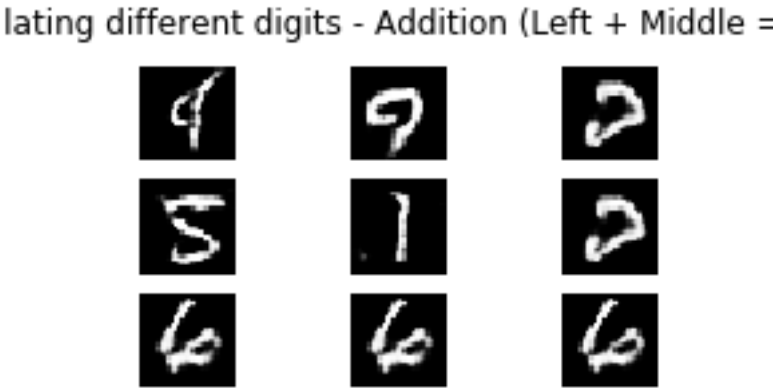

Figure 6: Novel samples generated by randomly selecting 16 samples $Z \sim N(0, I)$




As for our interpolation examples (Google Colab cropped our output image for some reason, apologies):

Figure 7: Interpolation - 20 samples (upper-left to lower-right)

Figure 8: Interpolation of latent vectors into new vector - DIFFERENT digits. Left and middle columns are the input $X + Y$, right column is the result $Z$




Regrettably, we couldn't perform interpolation of latent vectors of the same digits as the model wouldn't return consistent responses, an example of such an interpolation is:

Figure 9: Interpolation of latent vectors into new vector - SAME digits. Left and middle columns are the input $X + Y$, right column is the result $Z$


## 4 *GLO*

Trained GLO model using the same architecture as published by Raanan in the news forum for the GAN , except only using the generator part. The dimension of the latent vectors we chose for this architecture is 74 .

We trained the input latent vectors using an Adam optimizer with learning rate 0.01 , and the model parameters using an Adam optimizer with learning rate 0.001. Training was performed with the MNIST dataset with batch size 32, and 15 epochs (however results started being noticeable after 10 epochs), using an L2 loss instead of a VGG16-based perceptual loss.

We explored the dataset by generating 16 random samples from $Z \sim N(0, I_{74})$ , normalized to the unit sphere and interpreting these novel samples using the model, as well as choosing 5 specific classes with expected results and showing the difference between the original images and the generated images (using the latent vectors after training them). We also interpolated two samples from different classes and same classes in Z-space and explored the results.

We can see that the novel samples appear meaningless - this could happen because we need more training time, or because we need to normalize random samples differently .

In the second plot, we can see that the generated images are almost the same as their original counterparts but somewhat fuzzier . This means that we might achieve better results by training even better, however due to the nature of the L2 loss we don't expect the results to be significantly less fuzzy.

When interpolating digits from the same class , we can see that we get a result that appears to be from the same class . However , when interpolating digits from different classes we don't neccessarily reach results that are as good - we can see that in the first attempt interpolating 3 and 4 we get something that appears to be a 9, however when interpolating a different 4 and a different 3 we get a result that appears to be closer to a 3 than a 4 but in other attempts we got results that don't appear like digits at all. We assume this is as expected.

Figure 10: Novel samples generated by randomly selecting 16 samples $Z \sim N(0, I)$

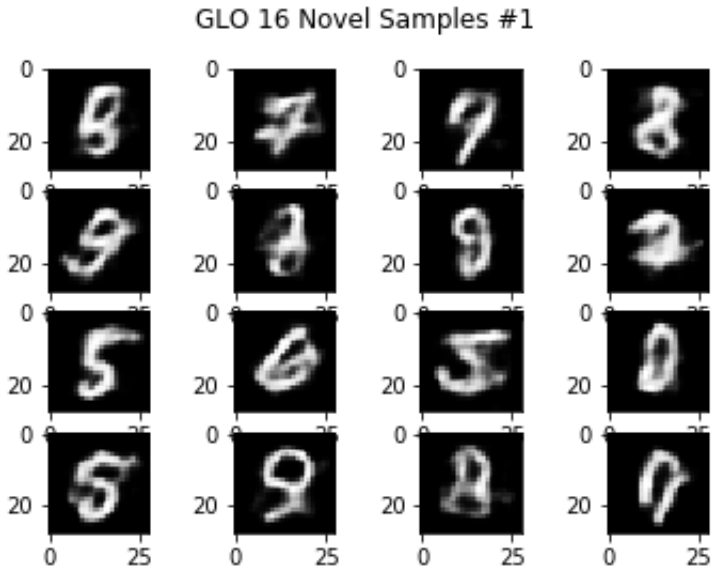



Figure 11: Sampling 4 samples I of each class and comparing original w/ generated using I's latent vector. Left-most column and odd-numbered columns: original images, even-numbered columns: generated images

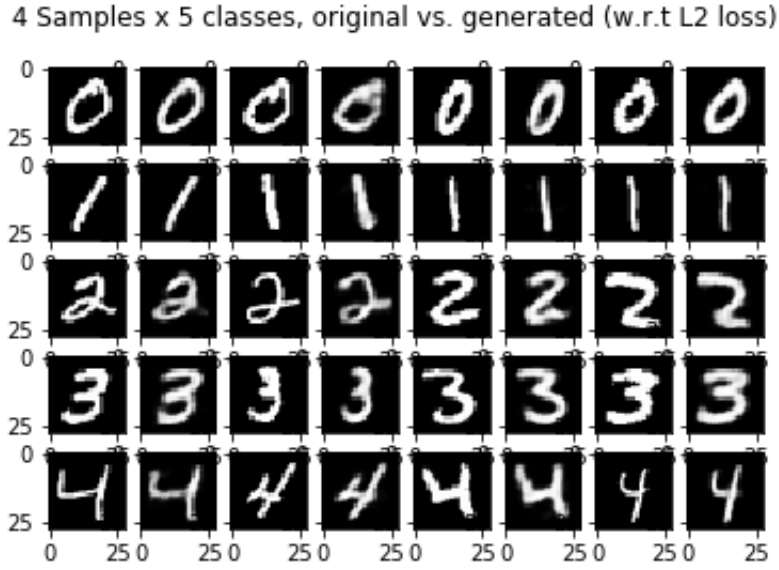


Figure 12: Interpolation of latent vectors into new vector - DIFFERENT digits. Left and middle columns are the input $X + Y$, right column is the result $Z$

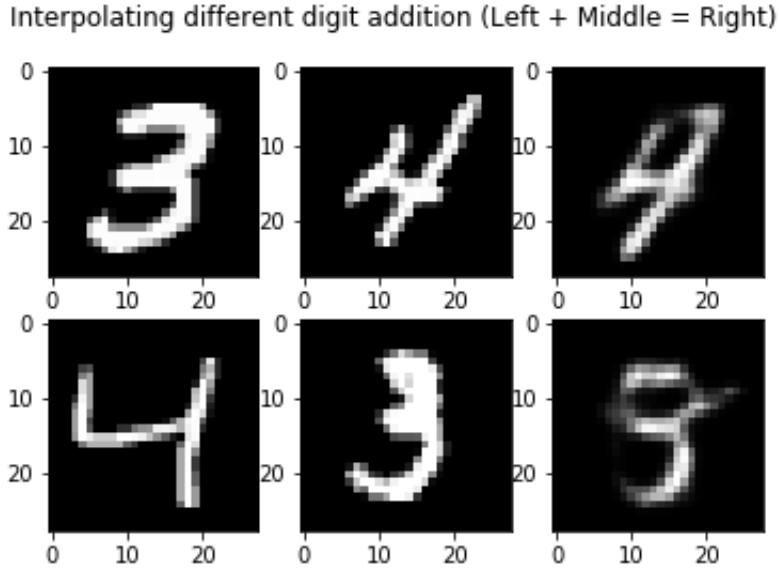



Figure 13: Interpolation of latent vectors into new vector - SAME digits. Left and middle columns are the input $X + Y$, right column is the result $Z$