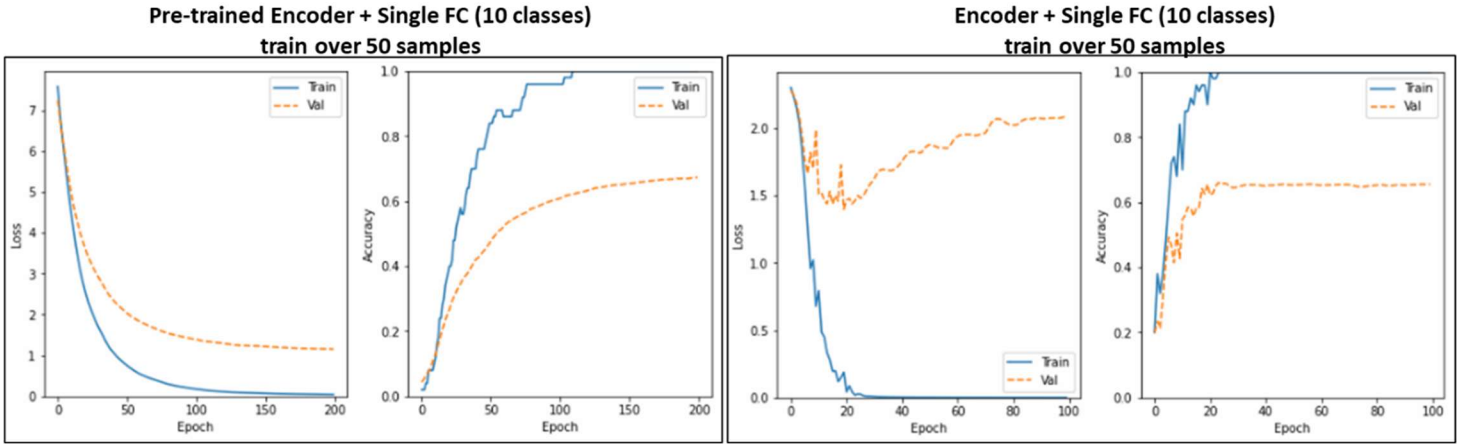
Intro to Deep Learning - Ex3

Guy Kornblit, 308224948; Mohammed Salama, 318983384

Question 2 - Transfer Learning

In this part, we used an encoder that was pre-trained on denoising task for the MNIST images as part of an autoencoder. We tested different architectures of MLP classifiers on top of the encoder to predict the number presented in the image after training on small subsample of the train set. To compare the result of our transfer-learning model, we examine an equivalent model equipped with the same architecture of the encoder followed by the same MLP sub-model as a final classification layer. Namely, the pre-trained model is required to learn a function from the latent space, where the regular model required to learn from the image space (32x32).

For the first trial, we examined an MLP model with a single layer, that maps each coordinate in the encoder latent space to output of dim 10. We trained this model for 200 epochs, and the equivalent model for 100 (until convergence). As shown in the following figure, it seems that the encoder and the FC layer without any pre-training, overfit the 50 train examples quickly (20th Epoch), while the loss over the entire test set (10k samples) remains high. The pre-trained model tends to over fit more smoothly but do generalize much better. To our understanding, the overfit happens since we have a small subset of train data over a lot of epochs, but the difference in generalization come from the representation of each image in the relevant latent space, which achieved by the pre-training of the encoder.



Even though, we checked the results for the regular model after training it entirely for classification, i.e. training an encoder and a classifier on much more examples. We expected to find that given appropriate number of examples the encoder would find a better latent space that more fit to the classification problem than the pre-trained encoder on the denoising task. Thus, producing better performance. So, we trained the described simple encoder+FC layer for 15 epochs, on batch size of 128.

Model	Pre-training (15 epochs)	Training	Test accuracy
Encoder + FC (10)	AE denoising task ($\sigma = 0.35$)	50 samples	0.651
Encoder + FC (10)	Classification task (end-to-end)	-	0.989

Classifiers comparison

In this part we compared different MLP architecture against pre-trained encoder on the denoising task with noise level 0.35 and latent vector dimension of 64. Each model has been trained on the same 50 train examples and tested on the 10k test set. Pre-trained trained for 200 epochs, and the regular model for 100 epochs.

MLP architecture	**Pretrained encoder + MLP - Model Accuracy**	**Encoder + MLP - Model Accuracy**
FC (10, sigmoid)	**0.651**	**0.655**
FC (32, relu) + FC (16, relu) + FC (10, softmax)	**0.647**	**0.54**
FC (64, relu) + FC (32, relu) + FC (16, relu) + FC (10, softmax)	**0.689**	**0.56**
FC (100, relu) + Dropout(0.5) + FC (64, relu) + FC (32, relu) + FC (16, relu) + FC (10, softmax)	**0.739**	**0.669**

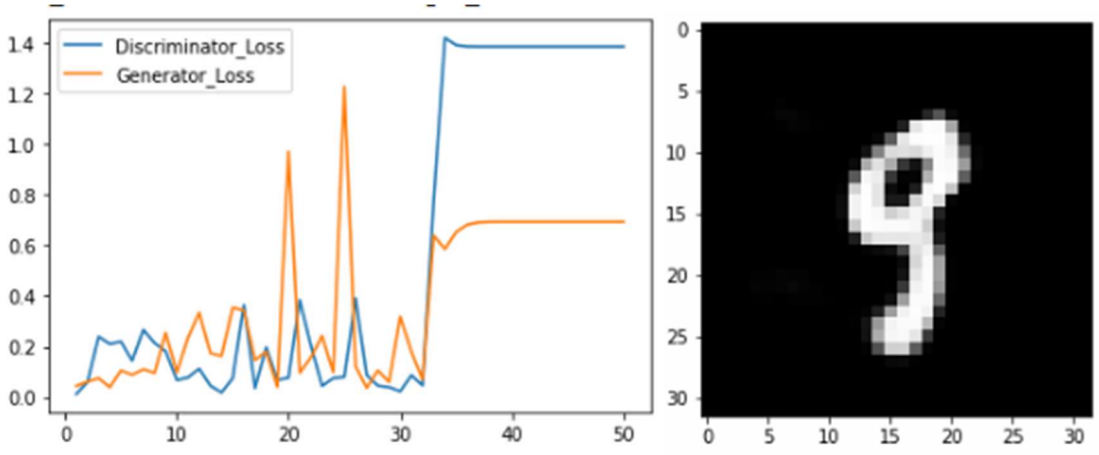
Question 3 - Latent GAN implementation

In this part, we implemented a GAN network over the MNIST latent space, as defined by the pre-trained autoencoder from the first part. We tested different architectures of MLP network to comprise the generator and the discriminator to enable novel latent vector generation that the pre-trained decoder would than convert to an MNIST image.

Baseline Architecture:
Discriminator: Latent vector -> FC (512, relu) -> … -> FC (64, relu) -> FC (1)
Generator: noise shaped image -> FC (512, relu) -> … -> FC (64, relu) -> FC (latent dim).

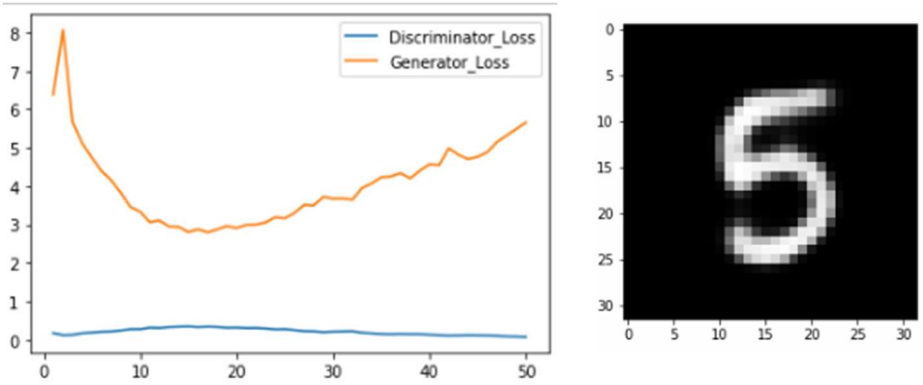
We used Binary Cross Entropy from logits.
First, we used the vanilla approach, which is to train fully the discriminator to differentiate between fake latent vectors that were created by the generator from a standard normal distribution matrix in the shape of MNIST image (32,32,1) and real latent vectors that were created by the pre-trained encoder from the real images.

We expected to see the discriminator saturation during the training process, but it seems that the training achieved the required effect and the generator was able to learn, but reached mode collapse to the digit 9.

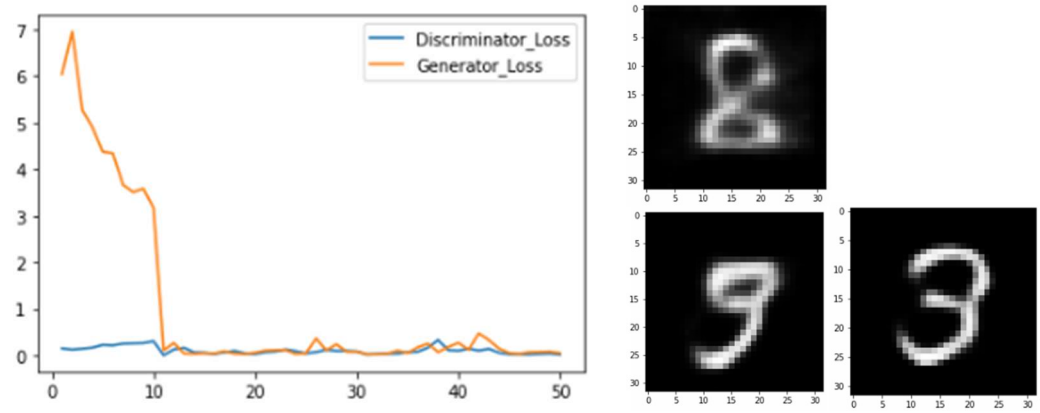


As a next step, we tried the approach that were used to avoid discriminator saturation. we trained the discriminator and the generator simultaneously on each batch. Namely, we trained the discriminator for one train step, and then we trained the generator for a single step (instead of training the generator over an optimized discriminator).

Surprisingly, this train method achieved the exact opposite behavior we expected. The discriminator reached saturation quickly and the generator was not able to learn significantly although it did reach mode collapse to the digit 5.



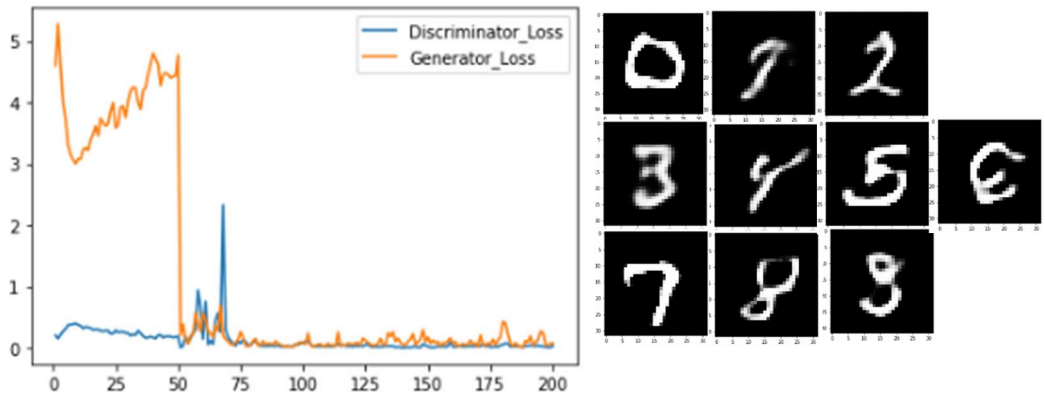
As final step, we tried to balance these two methods by training nets simultaneously for the first 15 epochs and train the rest of them by first optimizing the discriminator, and then the generator. We see that the generator was able to learn mainly in the first 10 epochs, since it had a chance to be optimized step by step with the discriminator (that had a near saturated loss due to the poor performance of the generator). After these 10 epochs, the generator reached convergence. This model collapsed to 3,8,9.



Interpolation Task

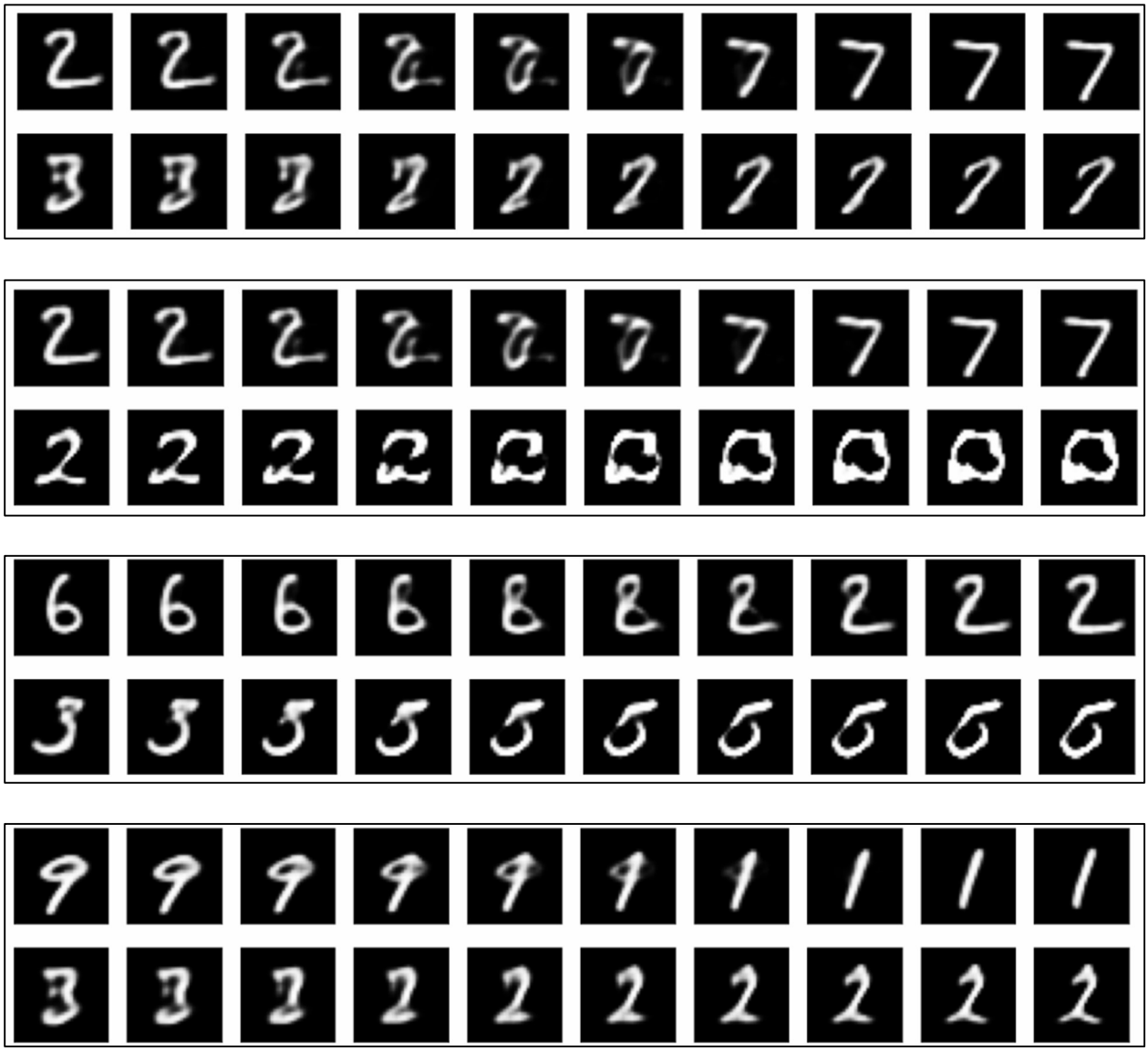
For the next part, we used the following architecture: FC (256, relu) -> FC (128, relu) ->FC(64, relu), for both discriminator and generator hidden layers. We trained this model for 200 iteration where the 50 first iteration trains the two nets batchwise, and the last 150 iteration optimizes the discriminator on the whole dataset and after optimizes the generator.

Final Model results:



We implemented the interpolation task by aL1+(1-a)L2 where L1 and L2 are the latent codes of two different digits and a is a value that increases from 0 to 1. The first row in the resulting image

is AE's latent space interpolation, where the second row is the interpolation over the GAN latent space (first choosing random matrix from the gaussian distribution, producing latent vectors and then preform the linear combination between them).



To our understanding, the interpolation shows the effect of traversing between two points in the AE and the generator latent space. First, the encoder and the decoder learned the latent space, and after the generator learned a mapping from the continuous normal distribution to a "fake" point in the same latent space. Thus, we assume that the mapping of the generator must be more robust to small changes in the input to imitate well the true distribution, even though the generator did not learn how to trick the decoder directly. Therefore, although it seems that for some number

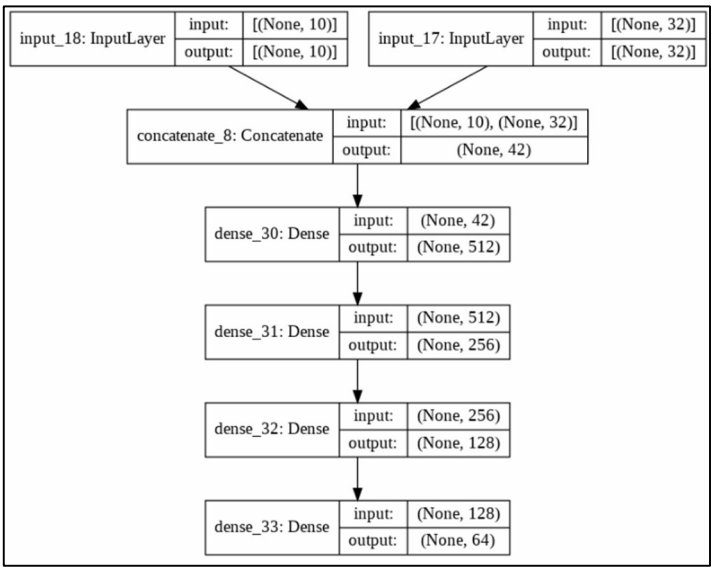
the Autoencoder can produce real digits on the line between two points (6-to-2, 9-to-1), we suspect that the generator performance is better.

Question 4 - Digit specific Generator

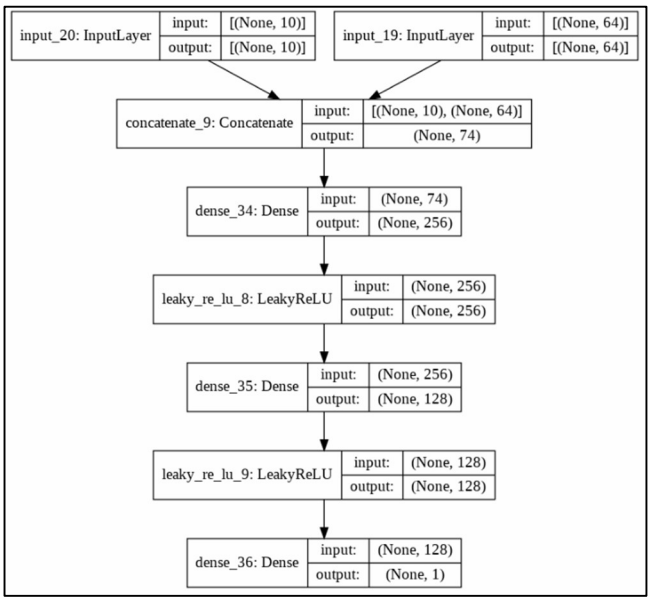
We used a conditional GAN architecture by supplying G with (digit, noise) and D with (digit, real latent vector for the digit image). Namely, we used the labels in the data set to train the model with paired data set and expected that the model could link between a one hot representation of the digit to its latent vector. Our training process involves training the first 50 epochs D and G step by step, and the rest 50 we trained optimizing every net separately. We could create more advanced hidden architecture to get better results.

Architecture:

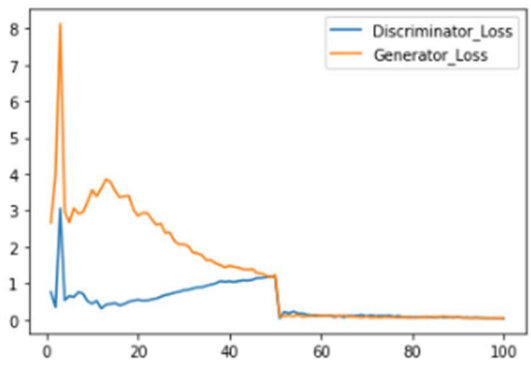
Generator:

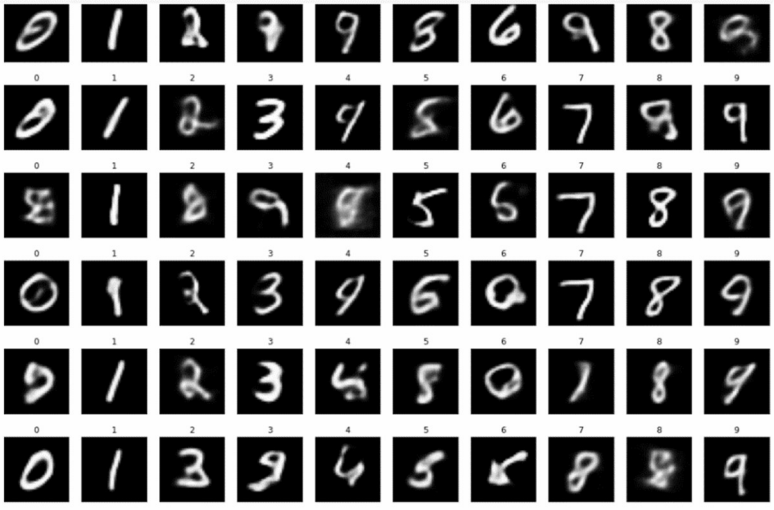


Discriminator:



Results:





Theoretical Questions

Question 1 - Transformers

In a convolution layer, we generate an activation layer with a filter that operates an affine transformation (might include non-linearity) on all channels of a window pixels. We can allow this kind of operation by adding positional encoding to each pixel to strengthen the link between certain pixels. Thus, approximately, we can increase the chance that the pixel neighbors will be chosen by the multi-head self-attention and will go through the same operation by the feed forward function. Even though, the patch will be encoded as a single vector and not a patch of different pixels and features.

If we avoid positional encoding, the basic transformer would be able to find relations between any k pixels in the whole image, in a way that approximately mimic an MLP network, except that there is a stage of embedding (and not affine trans. followed by non-linearity) before the FC layer.

Question 2 - Image Translation

Regular GAN-like approach

As the data become stricter, G will have less degrees of freedom in creating its output, by preserving content that appeared in the real images set. When dealing with unpaired images, D can only learn generalized features that correspond to class B, and G will have to mimic these features. Moving to loosely paired images, G will have to preserve the general contents of the image (the scene etc.) and find a mapping in that subspace. When working with fully paired data, G will have to preserve much more data from A and will learn to change fine visual features on every input to produce its output. We can think of this transition as moving from unsupervised to supervised learning, producing more accurate results.

Cycle GAN

Using unpaired images, both discriminators will only be able to learn generalized features for each class but using the cycle consistency loss adds a restriction to the problem which is that the transformations of both generators will be reproducible. Namely, given any image from either class, the restriction enforces generator A to pass sufficient information such that generator B will be able to return an image like the original one. Naturally, more supervised dataset can narrow the model search space thus creating much reliable generators.

Conditional GAN

This model will highly benefit from the paired datasets (loosely or not). D's ability to distinguish between (I,G(I)) to (I,J) when I, J are paired images will have meaning only if D could actually distinguish between the true mapping and a fake mapping thus encouraging G to mimic it. . Otherwise, for unpaired images, G will learn images that potentially correspond to the general class but with weak connection to the content of the input image.