67750 | Advanced Practical Course in Machine Learning | Ex 5

Guy Lutsker 207029448

1 Image Generation

In this exercise we were asked to use generate images to imitate images from the MNIST data set using Generative Latent Optimization.

1.1 Implementation

In my implementation I decided to use the first 2000 images from the MNIST data set, and I decided to try and embed the images in a 25 dimensional space - meaning I initialized my embedded random vectors for each of the class vectors and the content vectors in 25 dimensions. I also used a batch size of 32, and 50 epochs. To implement Generative Latent Optimization I used the provided network and I initialized optimizers for the network, and the two embedding matrices (used Adam optimizer). In each iteration in the training loop I used all three optimizers to train the network, as well as to learn the matrices for content and class. A note of following graphs: all of them were generated from randomly picked pictures (even ones where the same digit appears I wrote a code to generate pictures of the same digit randomly), and so you will not get "cherry picked" examples here :)

1.2 Conversions

Firstly I wanted to see how well can I convert a latent code of an image from one class to another, and my first results with no noise and with L1 loss were as follows:

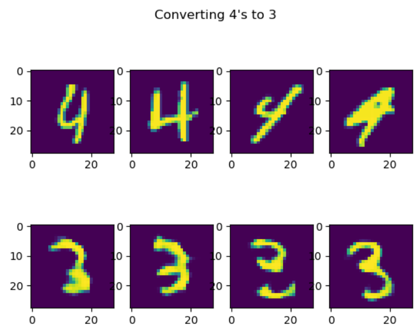


Figure 1: Conversion from 4 class to 3, in L1 loss no noise

These results are interesting and impressive. The network was able to convert these randomly chosen 4 images of 4's to images of 3's. Next I wanted to see all possible conversions, and so I made this picture:
https://www.cs.huji.ac.il/~guy_lutsker/APML.EX5_Conversion_of_4_to_3_Across_sigs.gif

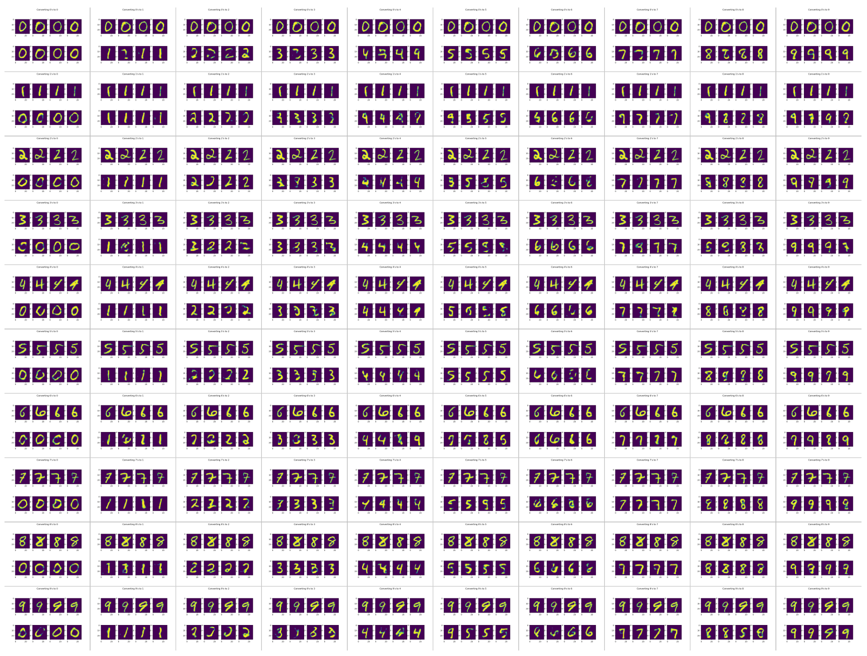


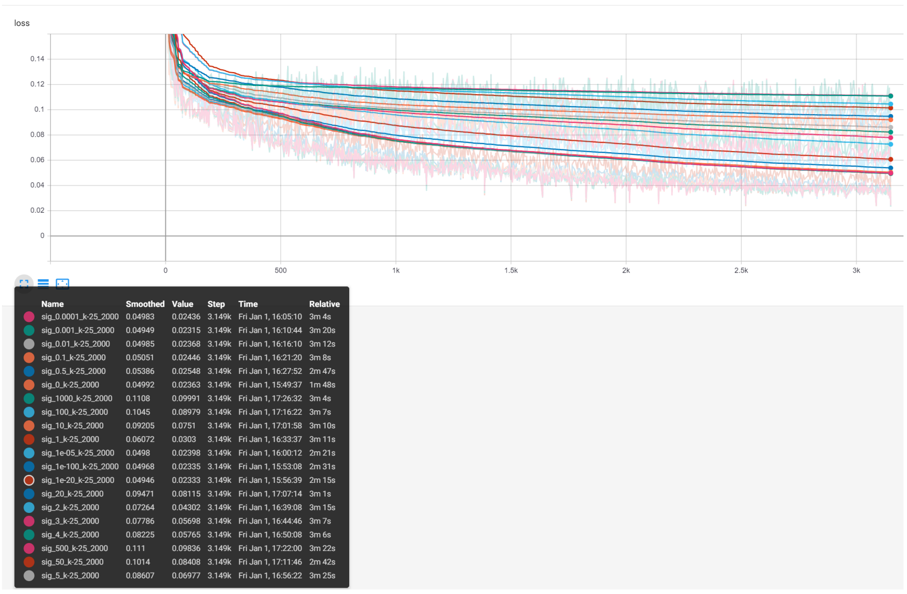
Figure 2: All possible conversions on L1 loss

I also wanted to see how to noise parameter changes our results and so I made these graphs for noise $\sigma \in [0, 1e-100, 1e-20, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 0.5, 1, 2, 3, 4, 5, 10, 20, 50, 100, 500, 1000]$. To visualize the changes the noise parameter I made the following gif:

https://www.cs.huji.ac.il/~guy_lutsker/APML.EX5_All_Possible_Conversions_In_L1_Loss.gif

1.2.1 Noise comparison

Next I wanted to see how noise would affect the loss convergence - I added noise to the content part of the input to the network. And here are the results:



	Name	Smoothed	Value	Step	Time	Relative
●	sig_0.0001_k-25_2000	0.04983	0.02436	3.149k	Fri Jan 1, 16:05:10	3m 4s
●	sig_0.001_k-25_2000	0.04949	0.02315	3.149k	Fri Jan 1, 16:10:44	3m 20s
●	sig_0.01_k-25_2000	0.04985	0.02368	3.149k	Fri Jan 1, 16:16:10	3m 12s
●	sig_0.1_k-25_2000	0.05051	0.02446	3.149k	Fri Jan 1, 16:21:20	3m 8s
●	sig_0.5_k-25_2000	0.05386	0.02548	3.149k	Fri Jan 1, 16:27:52	2m 47s
●	sig_0_k-25_2000	0.04992	0.02363	3.149k	Fri Jan 1, 15:49:37	1m 48s
●	sig_1000_k-25_2000	0.1108	0.09991	3.149k	Fri Jan 1, 17:26:32	3m 4s
●	sig_100_k-25_2000	0.1045	0.08979	3.149k	Fri Jan 1, 17:16:22	3m 7s
●	sig_10_k-25_2000	0.09205	0.0751	3.149k	Fri Jan 1, 17:01:58	3m 10s
●	sig_1_k-25_2000	0.06072	0.0303	3.149k	Fri Jan 1, 16:33:37	3m 11s
●	sig_1e-05_k-25_2000	0.0498	0.02398	3.149k	Fri Jan 1, 16:00:12	2m 21s
●	sig_1e-100_k-25_2000	0.04968	0.02335	3.149k	Fri Jan 1, 15:53:08	2m 31s
●	sig_1e-20_k-25_2000	0.04946	0.02333	3.149k	Fri Jan 1, 15:56:39	2m 15s
●	sig_20_k-25_2000	0.09471	0.08115	3.149k	Fri Jan 1, 17:07:14	3m 1s
●	sig_2_k-25_2000	0.07264	0.04302	3.149k	Fri Jan 1, 16:39:08	3m 15s
●	sig_3_k-25_2000	0.07786	0.05698	3.149k	Fri Jan 1, 16:44:46	3m 7s
●	sig_4_k-25_2000	0.08225	0.05765	3.149k	Fri Jan 1, 16:50:08	3m 6s
●	sig_500_k-25_2000	0.111	0.09836	3.149k	Fri Jan 1, 17:22:00	3m 22s
●	sig_50_k-25_2000	0.1014	0.08408	3.149k	Fri Jan 1, 17:11:46	2m 42s
●	sig_5_k-25_2000	0.08607	0.06977	3.149k	Fri Jan 1, 16:56:22	3m 25s

Figure 3: Loss (L1) on different noise level in training

Unsurprisingly enough we can see a correlation between noise level and final loss convergence, larger noise might “disturb” the loss reduction. In addition Noise comparison between the generated images can make a big difference. Small noise will result in unclear images and, while high noise might take over, and result in blurry images that don’t converge in anything. In addition you asked in the PDF “how classifying the class by the content vector can quantify this effect?”. To answer this question I want to address a problem we might encounter in this kind of analysis. In my understanding of the subject, when training we might get information transfer from the class space to the content space, this is ill-advised as it might cause us problems in the future. And so classifying the class by the content vector can quantify the effect of very low noise since if it will have a high accuracy we can infer we have an unsupported transfer of information we would like to avoid. After looking at the generated plots I decided the best hyper parameters for this task are loss L1, and noise = 0.1.

1.2.2 Learning through Epochs

Another kind of analysis I wanted to try was evaluating how my model “learns” to draw through time (epochs). To visualize the learning process I decided to create the following graph:

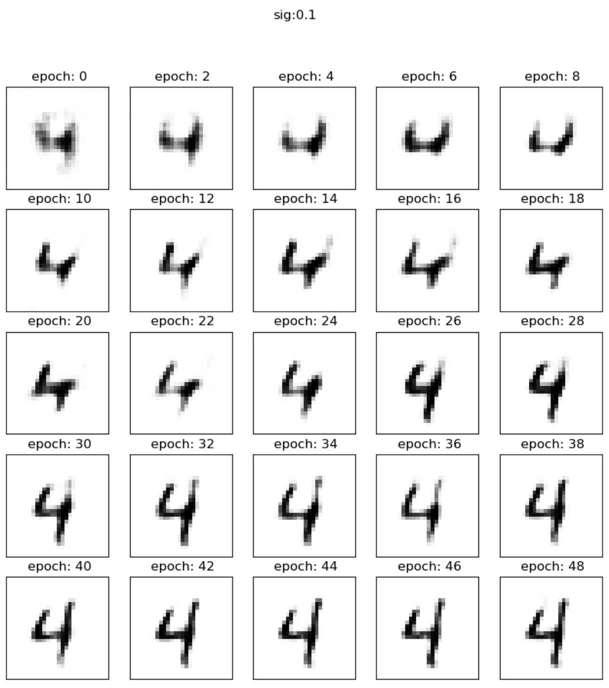


Figure 4: Generation of digit 4 through out training epochs with noise: 0.1 on L1 loss

As we can see the model indeed improves from epoch to epoch as one might expect. In addition I wanted to create the following visualizations(lyx +gifs $\neq$ good idea):

Comparison of different noise levels on learning though epochs (with loss L1/L2/L1+L2):

https://www.cs.huji.ac.il/~guy_lutsker/APML.EX5_Epochs_Learning_In_L1_Loss.gif

https://www.cs.huji.ac.il/~guy_lutsker/APML.EX5_Epochs_Learning_In_L1+L2_Loss.gif

https://www.cs.huji.ac.il/~guy_lutsker/APML.EX5_Epochs_Learning_In_L2_Loss.gif

Also a cool visualization of learning different digits(the first ones in the batch), though epochs(for L1, each frame is 1 epoch):

https://www.cs.huji.ac.il/~guy_lutsker/APML.EX5_Learning_to_draw.gif

1.3 Scratch generation

In addition to conversion of content from one class to another, another interesting idea is to generate images “from scratch”. The process is as follows: generate a random vector in the content space -> choose a class vector -> run

through network and be amazed(or at least you would have been if the data set would have been more cool :)).
Generated examples follow:

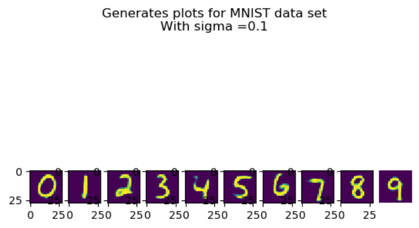


Figure 5: Generation of all digits from scratch from network trained on noise level: 0.1, with L1 loss

And of course, gifs of noise effect on the generated images in L1/L2/L1+L2 loss:
https://www.cs.huji.ac.il/~guy_lutsker/APML.EX5_Generation_Of_Digits_From_Random_LowDimVec_L1.gif
https://www.cs.huji.ac.il/~guy_lutsker/APML.EX5_Generation_Of_Digits_From_Random_LowDimVec_L1+L2.gif
https://www.cs.huji.ac.il/~guy_lutsker/APML.EX5_Generation_Of_Digits_From_Random_LowDimVec_L2.gif