ELEC 475 Lab 1

MLP Autoencoder

Jasmine Klein | 20154586

Matthew Valiquette |20151953

September 25th, 2023

The system was initially coded using the VSCode IDE, rather than PyCharm so the command did not work. Here is the updated version, thank you for your understanding😊

# Model Details

Our model is a simple MLP autoencoder with 748 (28x28) inputs in the first layer (L1) and 392 inputs in the second layer (L2) into a bottleneck of size 8. The image is then decoded back to 392, then decoded back to 784 (28x28) using a sigmoid function. All neurons are using the Relu function other than the last layer for decoding from 392 to 784 (28x28). The model was created in our model.py module and has three methods; forward, encode, and decode. Forward is encode and decode combined and was used for training.

A screen shot of a computer program

Description automatically generated

Figure : our model, a MLP Autoencoder with 4 fully connected layers.

# Training Details

An argument parser was used to extract the hyperparameters specified in as command line arguments. To train our model, the command **python train.py -z 8 -e 50 -b 2048 -s MLP.8.pth -p loss.MLP.8.png** can be entered on a PyCharm terminal and the following function will run.

A computer screen shot of a program code

Description automatically generated

Figure : train function in train module

Our network was trained using the Adam optimizer with a learning rate of 1e-3 and a weight decay of 1e-5. We used the MSE loss function and an Exponential LR scheduler with a gamma value of 0.9. We trained using the MSNIST training set, a batch size of 2048 and 50 epochs as specified in the command.

A screen shot of a computer code

Description automatically generated

Figure 3: argument parser to retrieve hyperparameters

A screen shot of a computer program

Description automatically generated

Figure : dataset loading for our test and evaluation set and model hyper parameter initializations.

The system worked well. The results were as expected with a loss of about 0.03. The loss plot behaved as expected as seen in the graph below it continued to diminish across all 50 epochs. A graph with a blue line

Description automatically generated

Figure : loss plot for step 3

# Results

Our lab1 source directory includes the modules lab1.py, train.py, test.py and model.py. To execute our cod and test our trained model, the command **python lab1.py –l MLP.8.pth** can be entered on a PyCharm terminal. This will instantiate a version of our model with our trained network weights and will run the test and interpolate functions from the test model. We initially had our step 4 and step 5 were in two separate functions; testFunc() and testWithNoise(). However, for simplicity we combined the two into one function, test().

A screen shot of a computer program

Description automatically generated

Figure : code to instantiate run steps 4,5 and 6

When you run the above command, the test function will prompt the user to enter a number that will be used to index images in the MNIST dataset.

A black background with white text

Description automatically generated

Figure : user input prompt after running python lab1.py -l MLP.8.png command in PyCharm terminal

The image at the inputted index and the images at the next two images will be used for testing. We used torch.rand() to add noise to our input images and evaluated our model again. The model did not perform as well with the addition of noise as shown below. The outputted results and corresponding figures discussed below.

A screen shot of a computer program

Description automatically generated

Figure : test function in test module

A screenshot of a test image

Description automatically generatedA collage of numbers

Description automatically generated

Figure : Step 4 output Figure 9: Step 5 output

We took the bottleneck of two separate images and interpolating them together, mixing their feature space. The linear interpolation was done for n=8 steps as shown below in the following figure.

A screen shot of a computer program

Description automatically generated

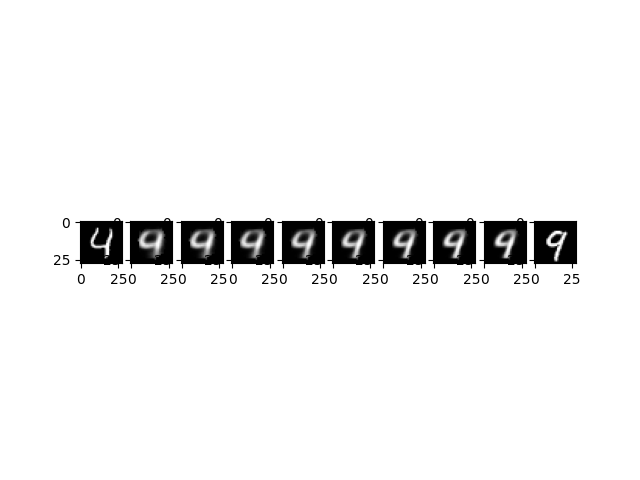


Figure 10: Step 6 output