Deep Learning Homework 7

Leow Wen Bin (2022280384)

May 2023

1 Variational Lower Bound

Consider the variational lower bound for a single example, in the case of a typical variational autoencoder:

$L(\theta, \phi, x) = E_{q(z|x;\phi)}[logp(x|z;\theta)] - KL(q(z|x;\phi)||p(z;\theta))$

In the above, z corresponds to the latent variable (vector of length 40), x corresponds to the image data, q represents the encoder, p represents the decoder, ϕ represents the parameters of the encoder, and θ represents the parameters of the decoder.

Essentially, the equation above sets the objective as difference between the following:

- $E_{q(z|x;\phi)}[logp(x|z;\theta)]$: the expectation of the decoder's log likelihood of observing x given z, taken with respect to the encoder's distribution of z given x.
- $KL(q(z|x;\phi)||p(z;\theta))$: The KL divergence between the encoder's posterior of z given x and the decoder's likelihood of z.

The above equation is not suitable for our model, since our model is conditioned on y (the class label of the image). We note that the class label is given as an input into both the encoder q and the decoder p. As such, the new variational lower bound of the new model should incorporate y into the posteriors of the original equation.

Hence, the new variational lower bound would be:

$L(\theta, \phi, x, y) = E_{q(z|x,y;\phi)}[logp(x|z,y;\theta)] - KL(q(z|x,y;\phi)||p(z|y;\theta))$

The above only applies to one datum (x,y). For the whole dataset, it becomes:

$L(\theta, \phi, D) = \Sigma_i(E_{q(z_i|x_i,y_i;\phi)}[logp(x_i|z_i,y_i;\theta)] - KL(q(z_i|x_i,y_i;\phi)||p(z_i|y_i;\theta)))$

2 Implementation

The implementation is done entirely in Python. There are also the following dependencies:

1. Pytorch

2. Torchvision
3. Matplotlib
4. Numpy

The code is provided as a single notebook. The cells can be run in order from top to bottom.

3 Data Preprocessing

The MNIST dataset is provided by Pytorch. To preprocess it into a form that is suitable for training, we convert the images to Tensor objects. We also perform rounding on all the elements of the tensors (corresponding to pixels) in order to binarize the data, as stated in the assignment requirements.

4 Model

The latent variable z has a dimensionality of 40, as required in the assignment document.

4.1 Q Model (Encoder)

The Q Model (encoder) takes an input vector of size 794. The first 784 elements correspond to the flattened image data, while the last 10 elements are a one-hot encoding of the class.

This input of 794 elements is fed to a fully-connected layer with 256 hidden units and a ReLU activation. This is in turn given to a pair of fully-connected layers with output size of 40. One of these hidden layers represents the mean of z, while the other represents the standard deviation of z. Both of these layers have sigmoid activations.

4.2 P Model (Decoder)

The decoder accepts an input vector of size 50. The first 40 elements correspond to z, while the last 10 represent the one-hot encoding of the class. This input is given to a fully-connected layer with hidden size of 256 and ReLU activation. The output of this layer is given to the final layer, which is a fully-connected layer of output size 784 and sigmoid activation. The reason for choosing a sigmoid activation here is to ensure the outputs are in the range from 0 to 1, which is easy to represent as an image.

4.3 Overall architecture

In the overall architecture, the input to the model (consisting of the image data and the class) is first given to the encoder. The encoder's output represent the

mean and standard deviation of z. Using this information and the reparameterization trick, a sample of z is generated. This is then given to the decoder along with the one-hot encoding of the class to generate the final image. The model also returns the distribution of z in order to perform normalization.

5 Training

The model is trained using binary cross-entropy loss to calculate the difference between the input and output images. By minimizing the BCELoss, we train the model to regenerate the same output image as the input image. We also perform regularization by penalizing the KL-divergence between z and the standard normal distribution. We train for 11 epochs using an Adam optimizer, batch size of 64, and learning rate of 0.001.

6 Results

The graph below shows how the losses of the model changed as the training progressed.

The next three images show some sample handwritten digits that were generated by the model after different amounts of training. For each image, all 10 digits were generated. For each digit, 10 sample images were generated by sampling z from a standard normal distribution. Evidently the model was already able to generate recognizable, decent digits after just one iteration. However, increasing the number of training epochs resulted in the digits becoming more clear and defined.

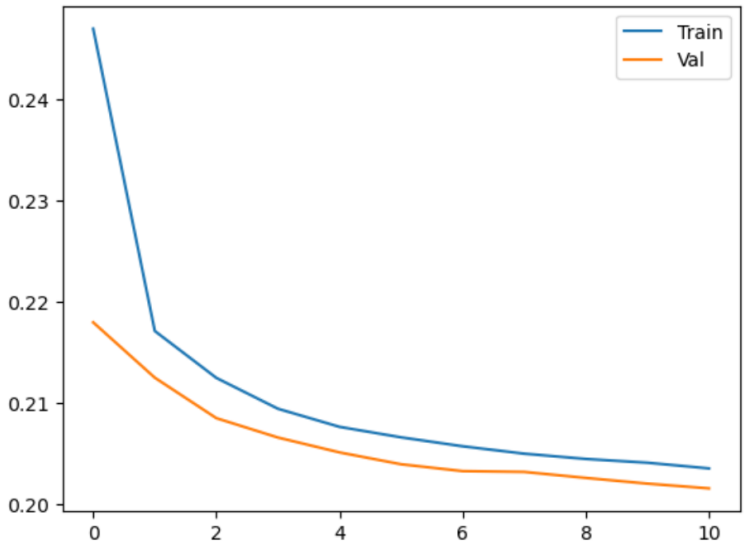

Figure 1: Graph of losses over iterations

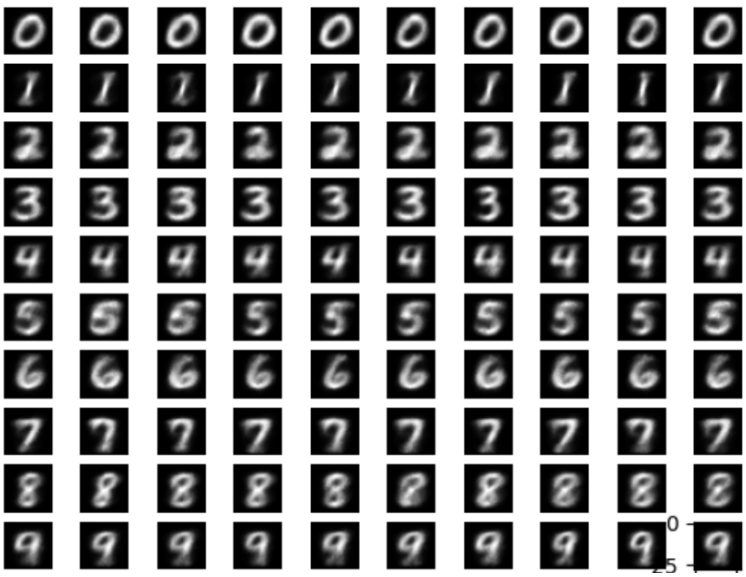

Figure 2: Images after 1 iteration of training

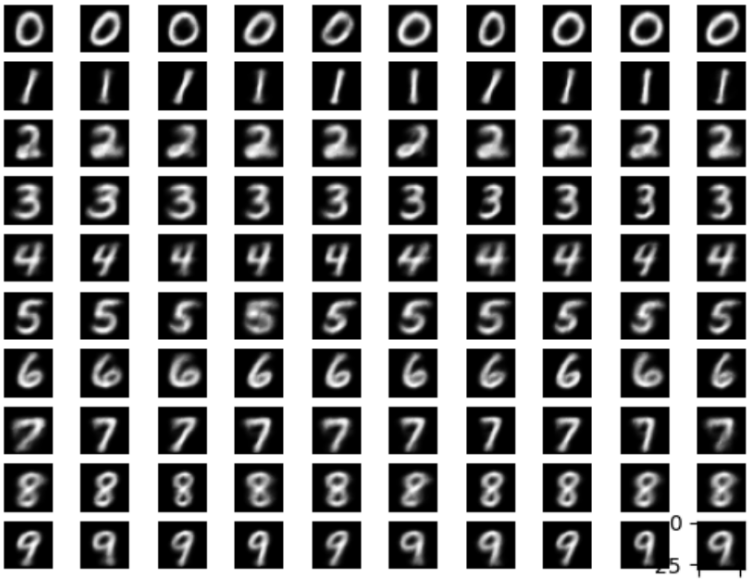

Figure 3: Images after 6 iterations of training

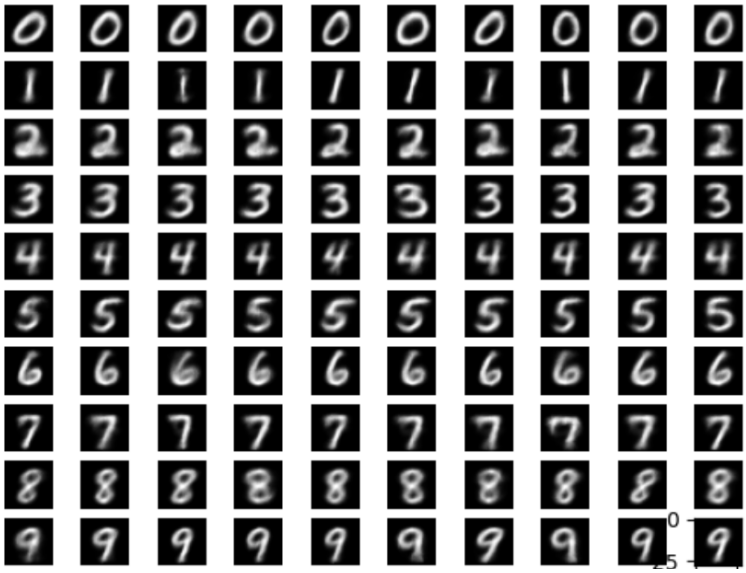

Figure 4: Images after 11 iterations of training