Name Surname: Duygu Serbes
Student Number: 2020700171

CMPE 597 Sp. Tp. Deep Learning
Spring 2021 Project III
Due: June 23 by 11.59pm

In this project, a VAE is implemented, where the encoder is an LSTM network and the decoder is a convolutional network. MNIST dataset is used, which consists of 28×28 hand written digits images.

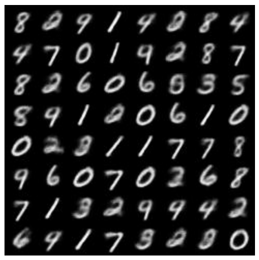
Network

As a base model, the model explained below is constructed. Fine tuning of hyper-parameter will be done next parts.

Adam optimizer is chosen with learning rate 0.001. Batch size is chosen as 64. Iteration will be done until 50th epoch due to time restriction during fine-tuning process.

Encoder is single layer LSTM. Input and sequence length equal to 28 due to the size of the images. Initial hidden size equals 128 which is given the two parallel linear layer to get latent space which dimension equals to 2. The mean and log variance values are used to sampling. Decoder part consists of two transpose convolution later after linear layer, which produce 6272 dimensional vector for CNNs to be transformed to multichannel feature maps. The 128 x 7 x 7 tensors given CNNs, which uses kernel size 4, spatial 2 and padding 1.

During experiments, I observed that increasing latent space cause decrease in validation and test loss and generated images become realistic significantly. For instance, when latent space equals to 6 the minimum validation loss is 6950.32 whereas when latent space dimension is 2, the minimum validation loss equals 9101.28 with base model. The output of the VAEs can be seen on Figure 1.


(a) 2 dimensional latent space

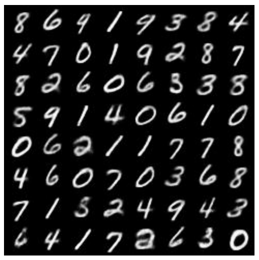

(b) 6 dimensional latent space

Figure 1: Output of the VAE interns of latent space

Original VAE paper, "Auto-Encoding Variational Bayes", found that high number of latent variables does not cause more overfitting, which is explained by the regularizing effect of the lower bound [1]. The paper provides 2D, 5D, 10D and 20D random generated images.

The reason of that situation is explained in Asperti's blog like that increasing the size of the latent space have the information at a greater level of detail, yet to be benefit from that high level information, complex and highly nonlinear transformation can be needed [2]. 2 dimensional latent space will be used rest of the project due to high number of benchmark model is available to be compared in terms of performance of the network.

Determination of Hidden Size

Original VAE paper uses 500 neurons in hidden layer, but they used fully connected layers in encoder and decoder part. They they determine hidden layer size based on previous literature, and indicated that dimension of hidden layer does not affect performance of the model in sensitive way [1].

Different hidden dimensions are used in these experiment as seen on Table 1, which shows the model loss values which is determined by choosing model which gives minimum validation loss [3]. As a result of these 3 experiment, hidden dimension 128 is selected due to its 9101.28 validation loss. Additional experiments are not required because of the small differences of loss values among different hidden dimensions. Also, the plot of the loss values belong to these 3 model is given Figure 2 until epoch 50.

Table 1: Hidden Dimensionality Experiment Results

Hidden Size	**Train loss**	**Validation Loss**
64	9428.99	9309.64
128	9157.46	9101.28
256	9222.29	9178.56

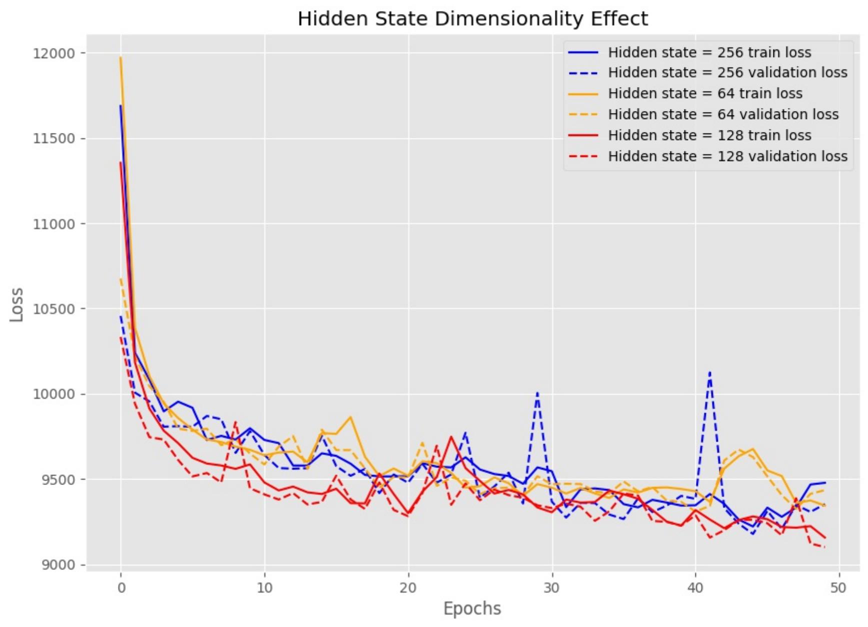


Figure 2: Hidden Dimensionality Experiment Loss Values

Convolutional Decoder

In that part different number of convolution layers are iterated ranged from 2 to 4. Kernel size, stride values and padding are arranged to keep desired output dimensions of the images.

As far as I read from literature transposed convolutions can have chequered board artifacts in generated images because of uneven overlap at some pixels. It can be fixed by selecting kernel size divisible by the stride, which is taken into consideration while constructing decoder [4]. Also stride 1 in the final transpose convolution have a chance to get better result because it is effective at struggling dampening artifacts [5]. Also it is suggested to use upsampling to reduce artifacts [5].

In all experiments, relu activation function is used except the final transpose convolution layer, which uses sigmoid activation function.

As 2 layer model, base model is used. There, After 2 dimensional sampling, linear layer increase the dimension to 6272 and then which is resized to [128x7x7] tensors. The first transpose convolution layer decreases the depth to 64 with kernel size 4, stride 2, and padding 1. It results [64x14x14] tensors. The second transpose convolution layer decreases the filters to 1 with same size of kernel, stride and padding with first transpose convolution layer, it results [1x28x28] tensors.

As 3 layer model, the decoder parts starts with fully connected layer which produces 3136 dimensional input, which is resized into [64 x 4 x4] tensors. First and second transpose convolution layer make the input channel number half with kernel size 4, stride 2 and padding 1. The third convolution later decrease channel number to 1 using kernel size 3, stride 1 and padding 1.

As 4 layer model, sampling layer dimension increases to 64 then which is resized to [64x1x1] tensors. The first transpose convolution layer decrease the channel number to 56 with kernel size 4, stride 1 and padding 0. The second transpose convolution layer has 28 output channel with kernel size 4, stride 2, padding 1. The third convolution layer has 14 channel size with kernel 3, stride 2 and padding 1. The last transpose convolution layer decreases the channel number to image channel which is 1 for MNIST dataset with kernel 2 stride 2 and padding 2.

Another 4 layer transpose CNN architecture performance is measured. The difference between third model is that the linear layer dimension before the first convolution layer. Here, latent space increased to 6272 and it directly transformed to the 1x1 kernel shape. The situation increases the number of parameter significantly and speed of the algorithm drops although the performance does not become better.

Also, another different 4 layer transpose CNN architecture is implemented, where 6272 dimensional fully connected output resize to [128x7x7] tensors. Channel number dropped by 2 after first deconvolution layer which has 56 channels. Kernel size is chosen as 3 except third transpose convolution layer. Padding equals to 1 except the final layer. As a result of this experiment, validation loss drops up to 9158.79 but after 15th epoch the loss values started to increase and after 20 epoch it stopped at epoch 35, because early stopping patience value determined as 20. Because the architecture is not stable, it is not selected.

The final iteration of 4 layer transpose CNN takes 3136 dimensional input vector and resize it to [64x4x4] tensors. All kernel size equals to 4 except last layer which has kernel size 3. Channel size dropped 64, 56, 28, 14 and 1 respectively. All padding values equal to 1 except third layer where padding is 2. In first and third layer, stride equals to 2 whereas rest of model uses stride 1.

As a result of these experiment there are no significant difference between first, second, third and sixth model. Because minimum validation loss is obtained from third model. So rest of the project 4 layer decoder is used by choosing sixth model.

Table 2: Decoder Experiments Results

Model Number	**Number of Transpose Convolution Layer**	**Train loss**	**Validation Loss**
#1	2	9157.46	9101.28
#2	3	9307.56	9142.16
#3	4	9160.4	9094.46
#4	4	9334.76	9395.92
#5	4	9168.57	9158.79
#6	4	9178.97	9093.10

Training

With the determined hidden dimension, and decoder part, the model trained with different latent space dimensions, 2, 5, 10 and 20 as carried out in the original VAE paper. The all result will be presented in Table 3. The loss values and regulatization terms are plotted Figure 3 and 4 respectively.

Table 3: Experiments based on Latent Space Dimension

Latent Space Dimension	Train loss	Validation Loss
2	9051.85	8989.65
5	6828.89	6954.74
10	5935.08	6035.80
20	5866.78	5939.02

For all models, general loss values drops up to certain points as architecture allows. When the latent space is 2, the loss terms is not very smooth for each epoch, significant up and down can be observed especially between epoch 10 and 20. Through at the end of the 100 epoch, the loss value tends to increase. Also, with latent space 5, the iteration stops at epoch 93 due to early stopping principle, where during lat 20 epoch, there is no validation loss improvement. Latent space 10 and 20 have very similar loss values.

On the other hand, KL divergence in the loss function behaves as regularization term. when we look at at 4, KL loss values increase during training. It fluctuates more when latent space equals to 2. Yet, here it has the lowest bound among other latent space dimensions. The reason behind the regularization term increase might be that the need of construct 10 classes for sampling.

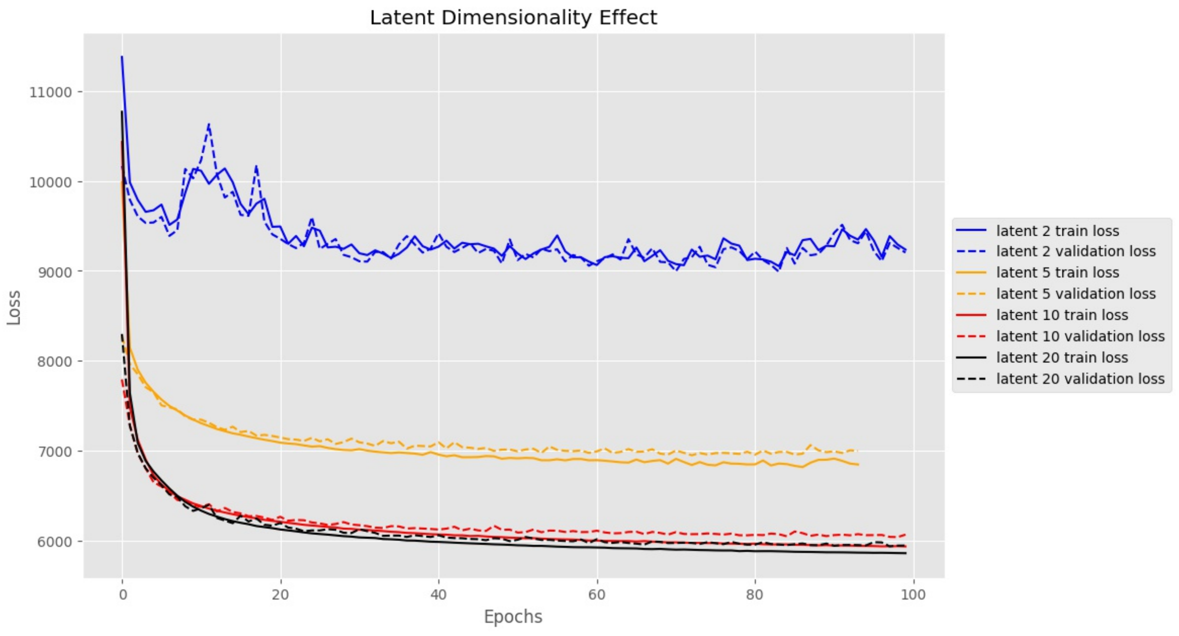


Figure 3: Loss Values

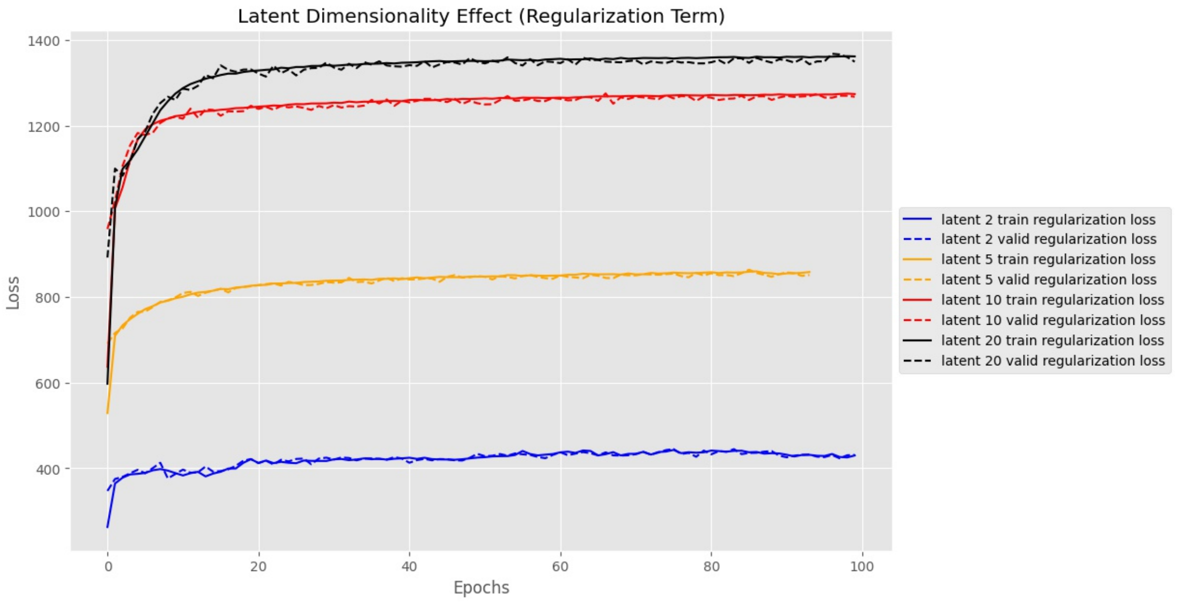
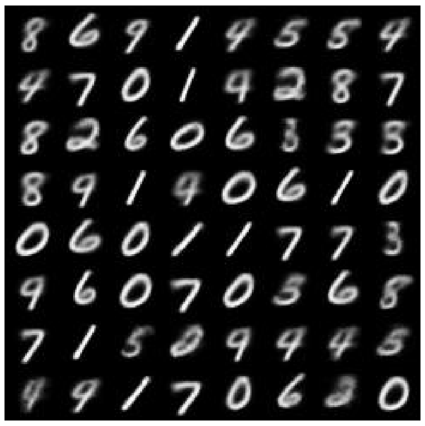
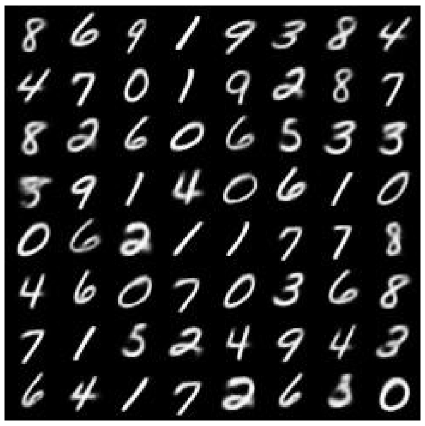
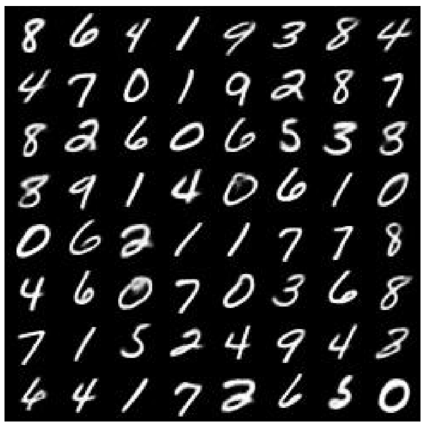


Figure 4: Regularization Term Loss

As a result of these experiments, minimum validation loss obtained from 20D lateral space. The result of the last batch of the epoch will be seen on Figure. There are significant difference between generated samples which uses different latent space dimension.


(a) latent 2


(b) latent 5


(c) latent 10

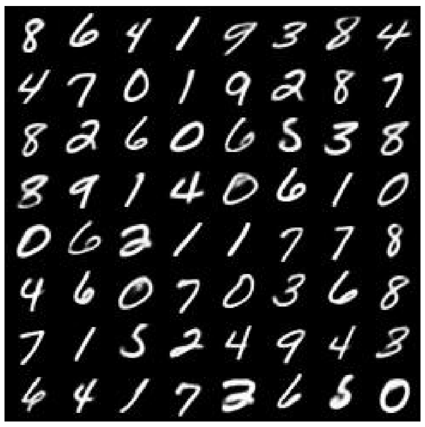

(d) latent 20

Figure 5: Generated images with different size of latent space

Generated Images

Display a grid of sampled digits

Using 2D lateral space model, 100 images are generated from 100 grid vectors between -1 and 1 [6]. The result can be seen on Figure 7. The decoder is successful especially digit 0, 1, and 9. But there are confusion between 2, 3, and 6. 3 and 5 conflicts with each other but by certain vectors it performs good.

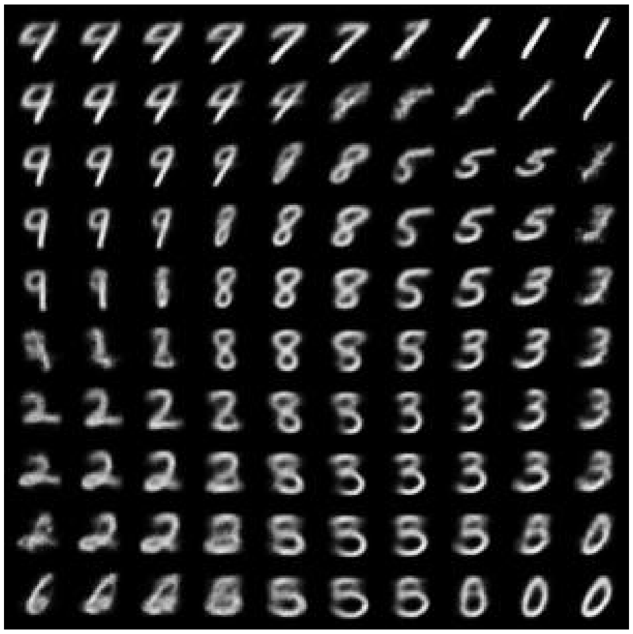


Figure 6: Generated Images

Display randomly sampled digits

In 2 lateral space, 100 vector generated with mean value 0 and variance 1. Using these vector, 100 digits are generated as seen on Figure 7. When I analyzed the results, 0 and 1 are most clear digits. Then 7 and 9 are also successful. But 2, 3 and 5 are the most worst digits. To be honest, their appearance are very similar and confusion among these digits can be acceptable and fair to a certain extent.

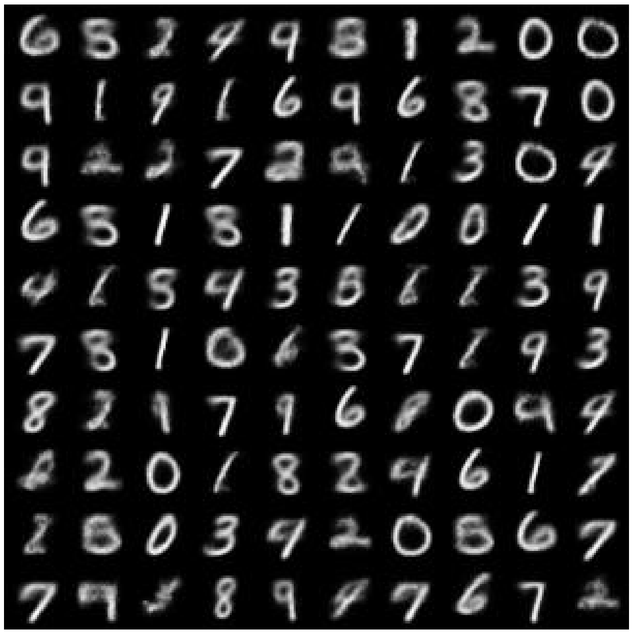


Figure 7: Generated Images

References

[1] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” 2014.

[2] A. Asperti. About variational autoencoders. [Online]. Available: http://www.cs.unibo.it/~asperti/variational.html

[3] Nutan, “Deconvolution and checkerboard artifacts,” *Distill*, 2016. [Online]. Available: http://distill.pub/2016/deconv-checkerboard

[4] D. Mishra. Transposed convolution demystified. [Online]. Available: https://towardsdatascience.com/transposed-convolution-demystified-84ca81b4baba

[5] A. Odena, V. Dumoulin, and C. Olah, “Deconvolution and checkerboard artifacts,” *Distill*, 2016. [Online]. Available: http://distill.pub/2016/deconv-checkerboard

[6] Convolutional variational autoencoder. [Online]. Available: https://www.tensorflow.org/tutorials/generative/cvae

[7] Udacity Deep Learning Nanodegree Program Self-Notes.

[8] Bogazici University, CMPE597 Deep Learning Class Self-Notes.