Deep Learning Homework 3

İpek Erdoğan,2019700174

December 1, 2021

Explanations and Discussions

My base model has a single-layer LSTM as Encoder and 2 transpose convolutional layers as Decoder. There are 2 fully connected layers at the end of the encoder to get mean and variance vectors. I have added Batch Norm layers after both of the 2 transpose convolutional layers. A sampler unit takes these vectors to sample a latent representation which fits to input data distribution. In this sampling stage, I have benefited from reparameterization trick. This trick aims to change sampling method in a way to make it differentiable and prevent backpropagation algorithm to get stuck in the sampler unit.

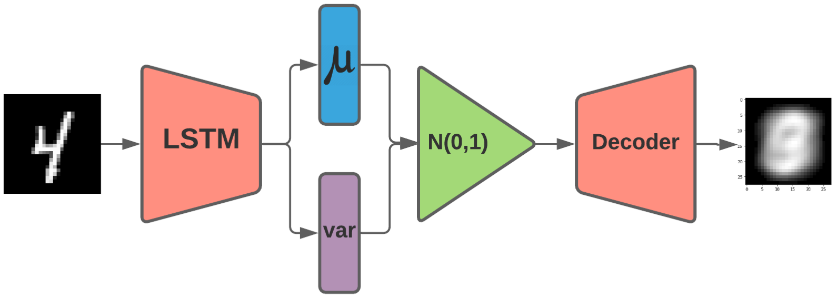
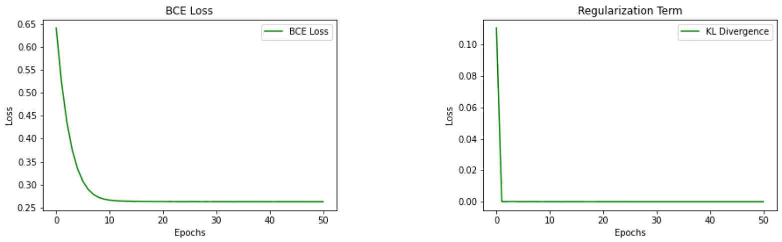


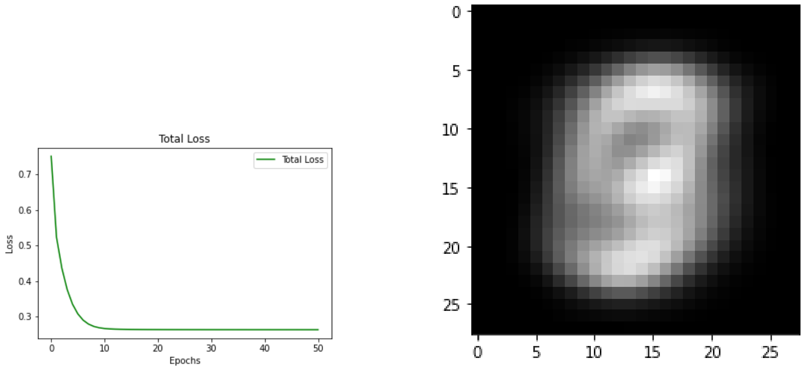
Figure 1: General Model

2 conv-layered base model didn't give satisfying results as a start. KL divergence dropped to zero so fast that model couldn't learn at all. Latent representation dimension was 128 and LSTM hidden layer dimension was 64.



(a) lr=0.01 (b) lr=0.001

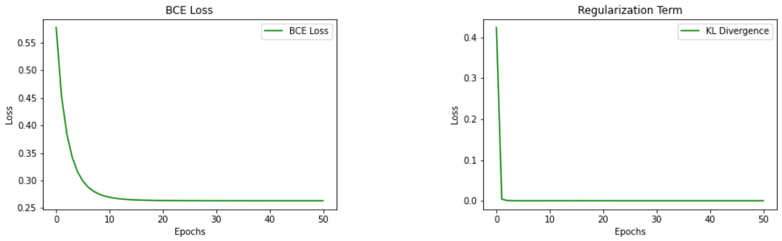
Figure 2: BCE Loss and Regularization Term



(a) lr=0.01 (b) lr=0.001

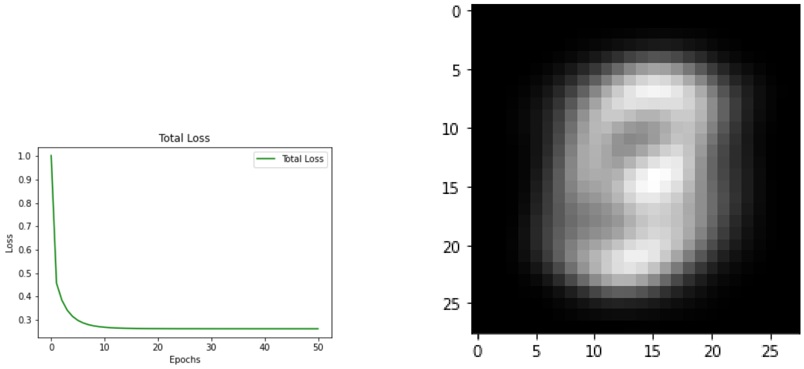
Figure 3: Total Loss and a Sample(Generated)

I have realized that I made a mistake in decoder's output part. I added two activation functions sequentially, so I removed one of them but this didn't effect the performance at all. Then I made LSTM hidden layer's dimension 128, to be sure that model was not missing any information due to small hidden layer size.



(a) lr=0.01 (b) lr=0.001

Figure 4: BCE Loss and Regularization Term

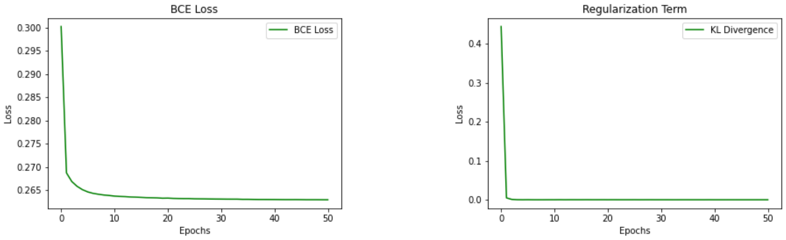


(a) lr=0.01

(b) lr=0.001

Figure 5: Total Loss and a Sample(Generated)

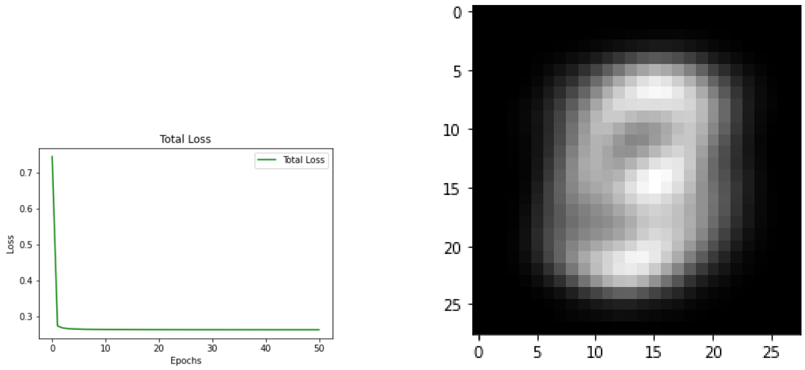
When I try to sample different images with different (randomly sampled) vectors, the results was always the same. This made me think that my model was not able to learn well so I deleted the Batch Norm layers. My mistake here was adding Batch Norm to the base model. I should have start with basic layers. Yet, this didn't improved performance much.



(a) lr=0.01

(b) lr=0.001

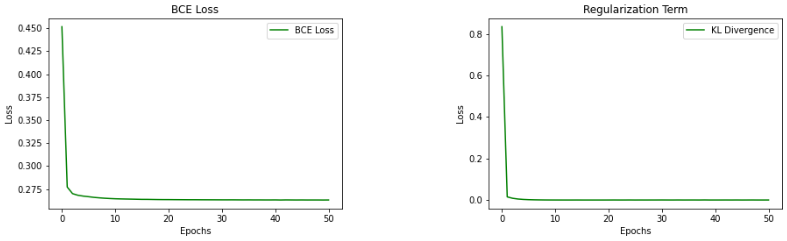
Figure 6: BCE Loss and Regularization Term



(a) lr=0.01 (b) lr=0.001

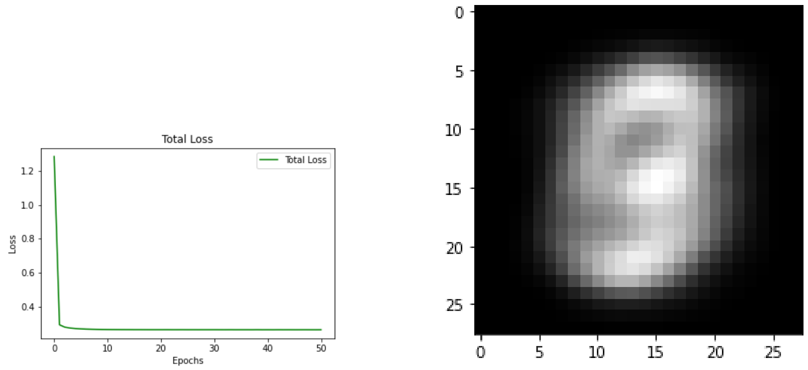
Figure 7: Total Loss and a Sample(Generated)

Then I have changed the dimensions of the linear layer which is at the beginning of the decoder. And deleted the Batch Norm layer which was following this linear layer, too.



(a) lr=0.01 (b) lr=0.001

Figure 8: BCE Loss and Regularization Term

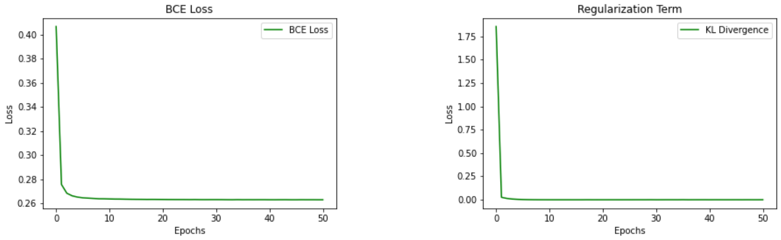


(a) lr=0.01

(b) lr=0.001

Figure 9: Total Loss and a Sample(Generated)

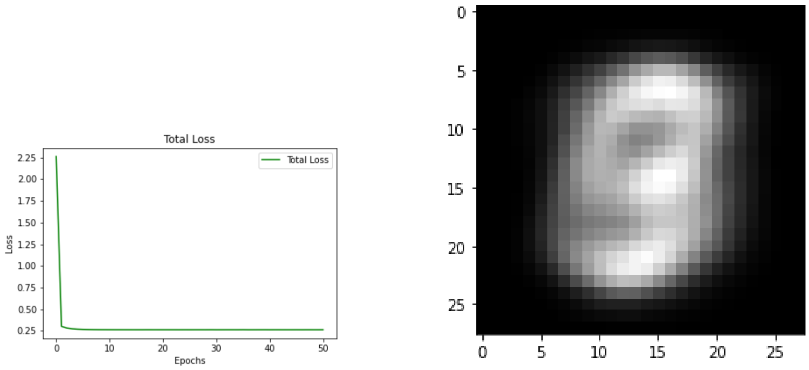
Finally, I was convinced that the big problem was not about these changes and decided to change my decoder architecture. I got inspired from VAE examples in internet (please see the References section). I added two more transpose convolution layers to my decoder and got rid of the padding in all of these layers since my inputs were already small and clear.



(a) lr=0.01

(b) lr=0.001

Figure 10: BCE Loss and Regularization Term



(a) lr=0.01

(b) lr=0.001

Figure 11: Total Loss and a Sample(Generated)

I have finally added dropout layers after 3 of 4 transpose convolutional layers but this didn't help too. With a better encoder, results may be different.

Additional Notes

Lecture slides, Pytorch Documentations [1], Pytorch Tutorial [2] and "Deep Learning" book [3] have been used during this study. Also I have inspired from different examples in internet [4], [5], [6], [7].

References

[1] "Pytorch Documentation", `https://pytorch.org/docs/master/nn.html`.

[2] "Pytorch Tutorial", `https://github.com/pytorch/tutorials/blob/master/beginner_source/blitz/cifar10_tutorial.py`.

[3] Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016, `http://www.deeplearningbook.org`.

[4] "Teaching a Variational Autoencoder", `https://towardsdatascience.com/teaching-a-variational-autoencoder-vae-to-draw-mnist-characters-978675c95776`.

[5] "The Reparameterization Trick", `https://sassafras13.github.io/ReparamTrick/`.

[6] "VAE Pytorch Tutorial", `https://github.com/Jackson-Kang/Pytorch-VAE-tutorial/blob/master/01_Variational_AutoEncoder.ipynb`.

[7] "Generative Models - Variational Autoencoders", `https://atcold.github.io/pytorch-Deep-Learning/en/week08/08-3/`.