1 Summary

Previously the GAN that I was trying to implement was from a different paper [1] in which they used convolutional neural networks to implement GAN. Since the GAN architecture proposed in the original paper was very different [2], I implemented the model inspired by the original paper in order to be able to fairly compare the results with VAE. First I created the Generator and Discriminator networks. The generator has the following input and output structure

$$100 \xrightarrow{\text{LeakyReLU}} 128 \xrightarrow{\text{BatchNorm,LeakyReLU}} 256 \xrightarrow{\text{BatchNorm,LeakyReLU}} 512 \xrightarrow{tanh} 784$$

Where 100 is the input dimension of the latent space and 784 are the dimensions (28×28) of the output image. Each arrow indicates a linear transformation of type $y = xA^T + b$. The discriminator has the following input and output structure

$$784 \xrightarrow{\text{LeakyReLU}} 512 \xrightarrow{\text{LeakyReLU}} 256 \xrightarrow{\text{LeakyReLU}} 128 \xrightarrow{sigmoid} 1$$

Where 784 is the input dimension of image and 1 is the dimension of the output ($[0, 1]$). Each arrow indicates a linear transformation of type $y = xA^T + b$.

I used Binary Cross Entropy loss function and Adam Optimizer with 0.001 learning rate, $\beta_1 = 0.5$, and $\beta_2 = 0.9999$.

For training: For each batch, First I sampled vectors from the latent space and used them as input to generate images through generator and then ran those generated images through discriminator and ran BCE loss against the results of discriminator (true label 1 for generated images) and then ran back propagation on the generator and ran an optimizer step on generator. Then to train the discriminator I ran generated images as well as training set images through the discriminator and calculated the BCE loss against the results (true label 1 for all training images and label 0 for all generated images). After calculating the loss on the discriminator, I ran back propagation on the discriminator and ran an optimizer step on the discriminator.

After every epoch, I sampled 64 vectors from the latent space and ran them through the generator to generate images of some hand written digits. I have included the images in the Appendix A.

2 Future Work

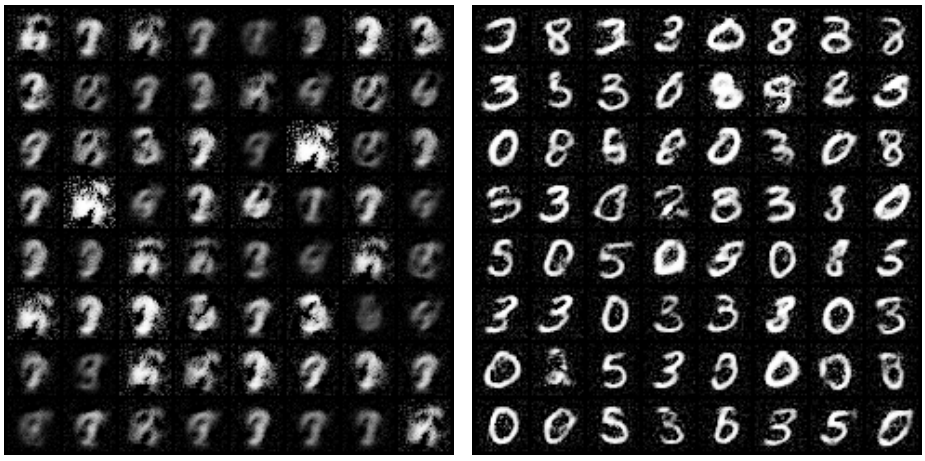
As I have finished implementing both GAN and VAE, It's time to wrap up the study and report the findings in a single paper.

References

[1] `https://arxiv.org/pdf/1511.06434.pdf`

[2] `https://arxiv.org/pdf/1406.2661.pdf`

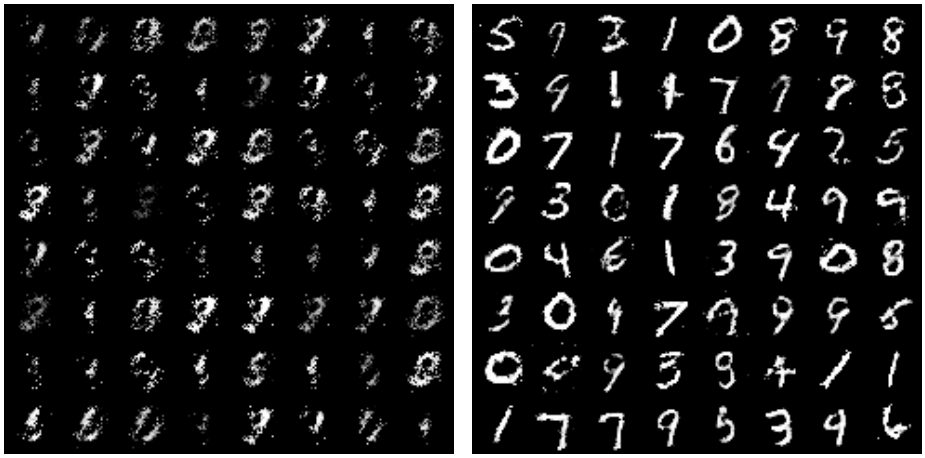
A Generated Images



(a) Epoch: 1 (b) Epoch: 30

Figure 1: Generated Images without input image normalization

As we can see there is some white noise even after 30 epochs. So I tried normalizing the input images to range $[-0.5, 0.5]$.



(a) Epoch: 1 (b) Epoch: 30

Figure 2: Generated Images with input image normalization