1 Summary

Even though I was not able to fully understand the mathematics behind the VAE, I understood how the model should be set up so that I am able to train the VAE on MNIST dataset and generate some hand written digits. I also found that `pytorch` has a `datasets` class which contains standard data loaders for various datasets so I used that one instead of my custom data loader since it would be more compatible with `pytorch`. I was able to implement the VAE using `pytorch` and taking some help from `stackoverflow`. I am generating the samples from 20-D latent space. Given that, My model has the following architecture $784 \to 400 \to 200 \to 20 \to 200 \to 400 \to 784$. The appendix in the paper [1] was helpful in clearing out some confusion as well. I trained the model on training data for 30 epochs, 128 batch size, and 10^{-3} learning rate. Also, I used ReLU activation function and adam optimizer as I found that these make the network converge faster. For loss, I used Binary Cross Entropy + KL term (as indicated in Appendix B of the paper [1]). Initially I used only BCE loss but the generated samples were not hand written digit-like at all. So I had to look up on the internet about the loss function of VAE.

2 Model Code

```
class VAE(nn.Module):
    def __init__(self):
        super(VAE, self).__init__()

        self.fc1 = nn.Linear(784, 400)
        self.fc2 = nn.Linear(400, 200)
        self.fc31 = nn.Linear(200, 20)
        self.fc32 = nn.Linear(200, 20)
        self.fc4 = nn.Linear(20, 200)
        self.fc5 = nn.Linear(200, 400)
        self.fc6 = nn.Linear(400, 784)

    def encode(self, x):
        h1 = F.relu(self.fc1(x))
        h2 = F.relu(self.fc2(h1))
        return self.fc31(h2), self.fc32(h2)

    def reparameterize(self, mu, logvar):
        std = torch.exp(0.5 * logvar)
        eps = torch.randn_like(std)
        return mu + eps * std

    def decode(self, z):
        h3 = F.relu(self.fc4(z))
        h4 = F.relu(self.fc5(h3))
        return torch.sigmoid(self.fc6(h4))

    def forward(self, x):
        mu, logvar = self.encode(x.view(-1, 784))
        z = self.reparameterize(mu, logvar)
```

```
31          return self.decode(z), mu, logvar
```

3 Generated Samples

Here are some generated samples from epoch 1 and epoch 30

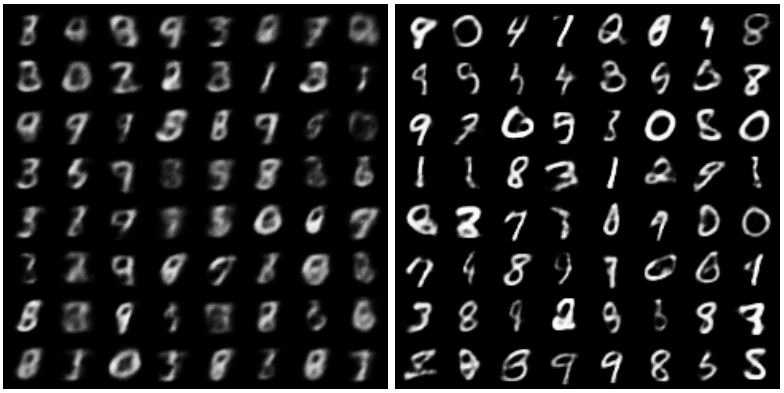

Figure 1: Epoch 1 and Epoch 30

We can clearly see that the features of the digits are much sharper after training for 30 epochs. The loss at the start of the training was 183 and towards the end of training it was reduced to 99.

4 Next Steps

Now that I am able to generate some samples from the VAE, I would like to do the same for GAN as well (I am able to generate some samples from GAN but they are not good). After successfully generating the samples from both, I would like to compare those samples and see which model is better in terms of training required and convergence time. In parallel, I would like to improve my fundamental understanding of GAN and VAE by reading more about it.

References

[1] `https://arxiv.org/pdf/1312.6114.pdf`