Lab4-1 - Conditional Variational Autoenconder

Wen-Jie Tseng (0556146)
Deep Learning and Practice - 2018 Spring Semester

1. INTRODUCTION

Conditional Variational Autoencoder (CVAE) is an extension of Vairational Autoencoder (VAE). The model can be seperated into two parts, an encoder and a decoder. Encoder extracts the features and use them to estimate the parameters of the target data distribution (e.g., μ_{enc} and $log({\sigma_{env}}^2)$. Combine the noise plus one hot vector and feed to the decoder. After decoding, we can obtain the result generated by our model.

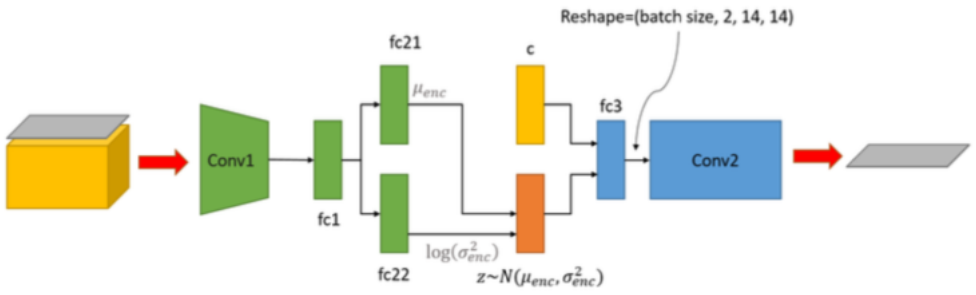
Reshape=(batch size, 2, 14, 14)
fc21
c
μ_{enc}
fc3
Conv1
Conv2
fc1
$\log(\sigma^2_{enc})$
fc22
$z\sim N(\mu_{enc}, \sigma^2_{enc})$

Figure 1. The model architecture of CVAE.

2. EXPERIMENT SETUP

2.1 Implementation of CVAE

Basically, I followed the instructions on Lab description and the code of model architecture are provided as below. In conv2 we have to do upsampling, expand the input noise to output image size. For passing value between different layers, I usually use .size() and .view() make sure my input and output have equal size.

```
self.conv1 = nn.Sequential(
    nn.Conv2d(11, 3, 3, 1, 1),
    nn.ReLU(),
    nn.Conv2d(3, 1, 3, 1, 1),
    nn.ReLU()
)

self.fc1 = nn.Sequential(
    nn.Linear(784, 400, bias=True),
    nn.ReLU()
)
self.fc21 = nn.Linear(400, 20, bias=True)
self.fc22 = nn.Linear(400, 20, bias=True)
self.fc3 = nn.Sequential(
    nn.Linear(30, 392, bias=True),
    nn.ReLU()
)
self.conv2 = nn.Sequential(
    nn.Conv2d(2, 11, 3, 1, 1),
    nn.ReLU(),
    nn.UpsamplingNearest2d(scale_factor=2),
    nn.Conv2d(11, 3, 3, 1, 1),
    nn.ReLU(),
    nn.Conv2d(3, 1, 3, 1, 1),
    nn.Sigmoid()
)
```

2.2 Providing labels as additional input channels to both the encoder and decoder

To attach one hot vector on input and noise, we can use unsqueeze() or view() to increase the dimension. Next use expand() to make two tensors, which we want to concatenate, in equal dimension. For example, the follow code can attach one hot vector to each pixel of input image.

```
    one_hot_tensor = torch.Tensor(one_hot_lst).unsqueeze(-1).cuda()
    one_hot_tensor = torch.unsqueeze(one_hot_tensor, -1)
    one_hot_tensor = one_hot_tensor.expand(-1, -1, 28, 28)
    new_data = torch.cat((data, one_hot_tensor), dim=1)
```

3. RESULT

3.1 Disentanglement experiments

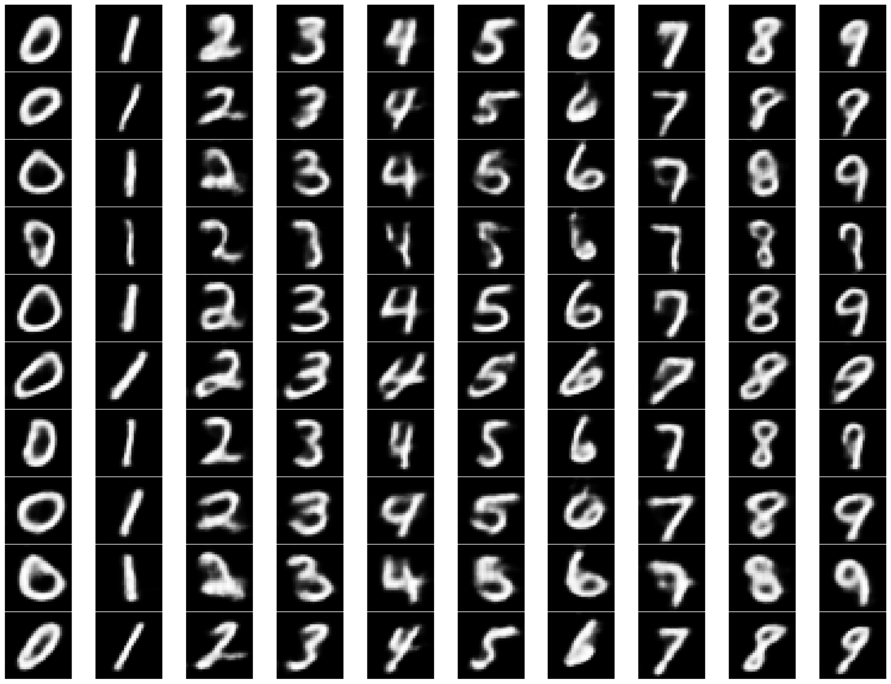

Figure 2. The disentanglement experiment, each row shares different one hot vector but same noise, each column shares different noise but same one hot vector.

3.2 Training loss curve

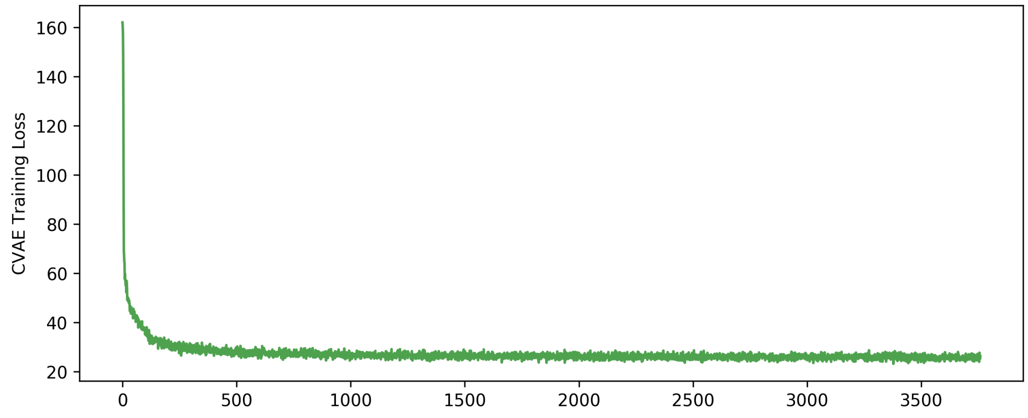

Figure 3. The Loss curve of CVAE. X axis was recorded in each 10 batches.

4. DISCUSSION

Compared to InfoGAN, CVAE performs in a much more stable way. This result in Figure 2 shows that our model can learn the disentangled representation well. However, the output suffers a little bit of blurry.

REFERENCES

1. What if VAE.
2. Pytorch VAE example provided in Lab4.