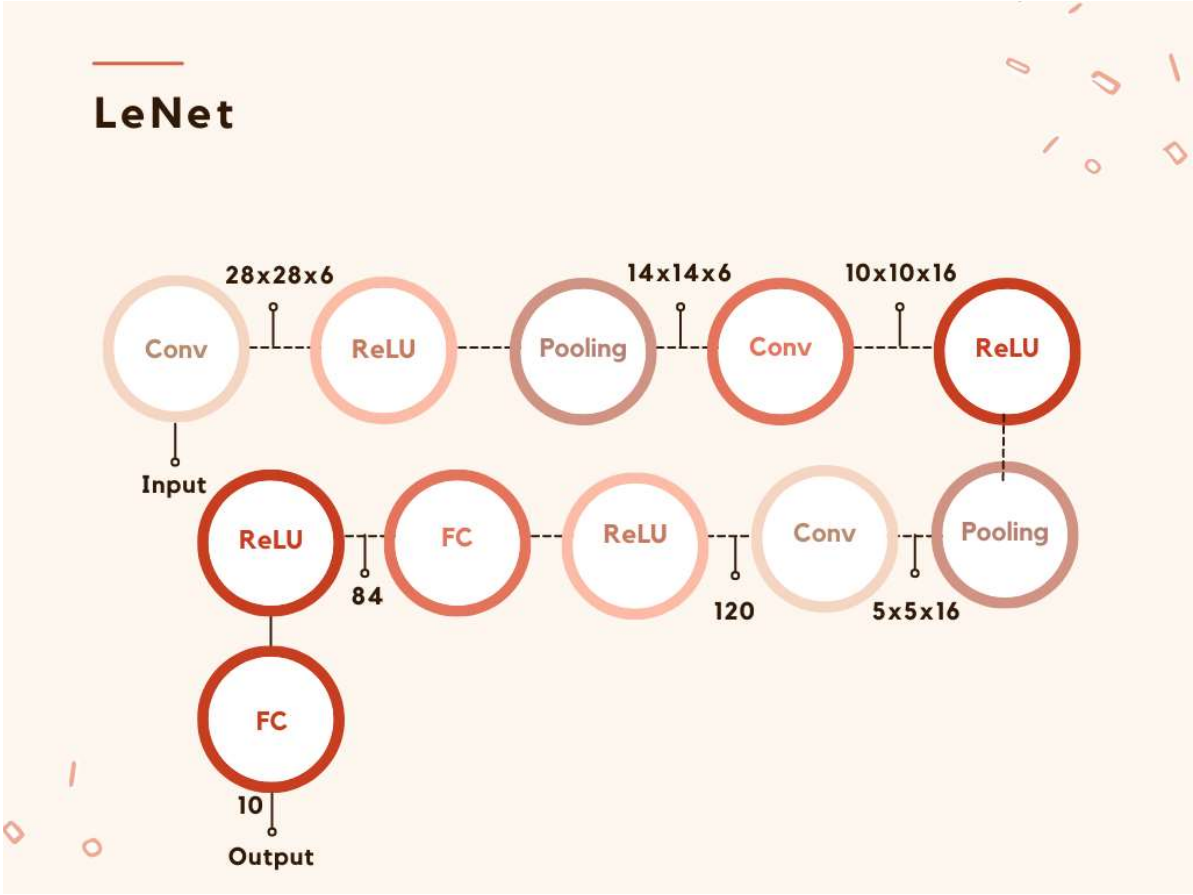
Machine Learning Final Project

Ziqi Huang 521021910695

Fashion-MNIST clothing classification

LeNet

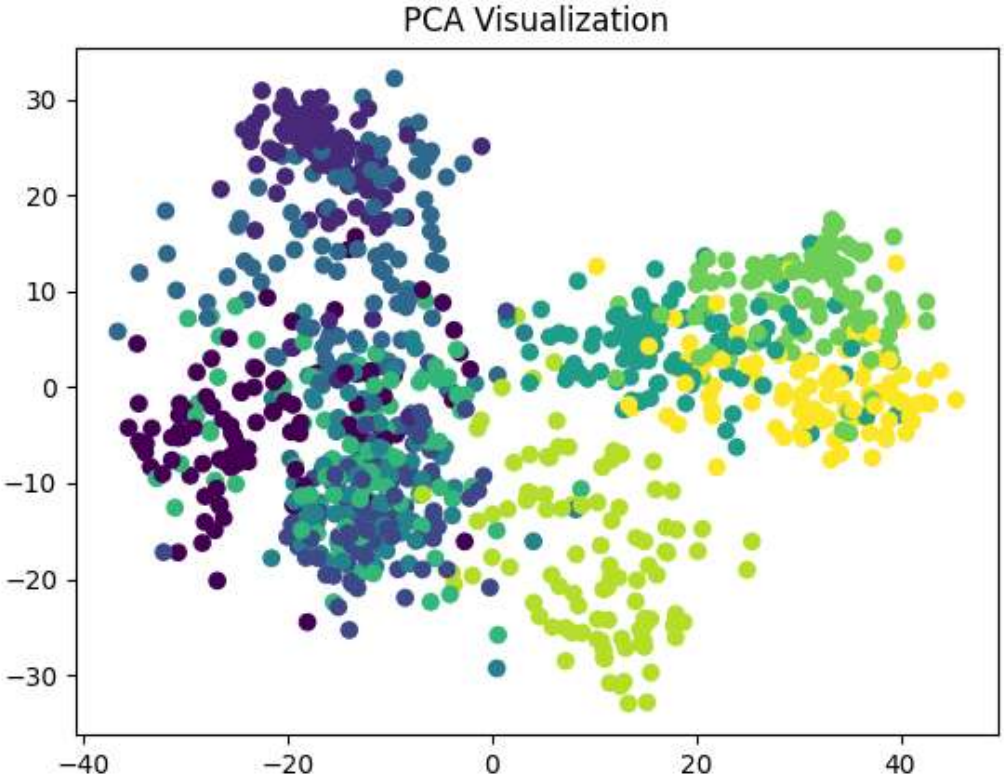
The output accuracy: 79%



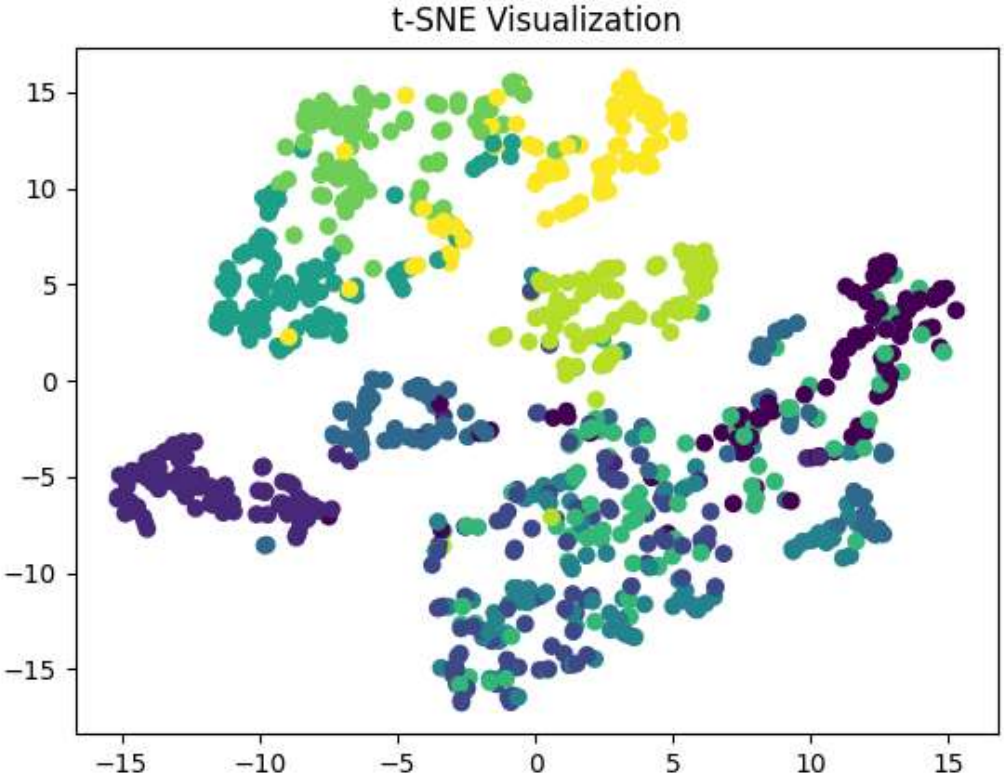
Output (the complete output image set is in the attachment)

Visualization

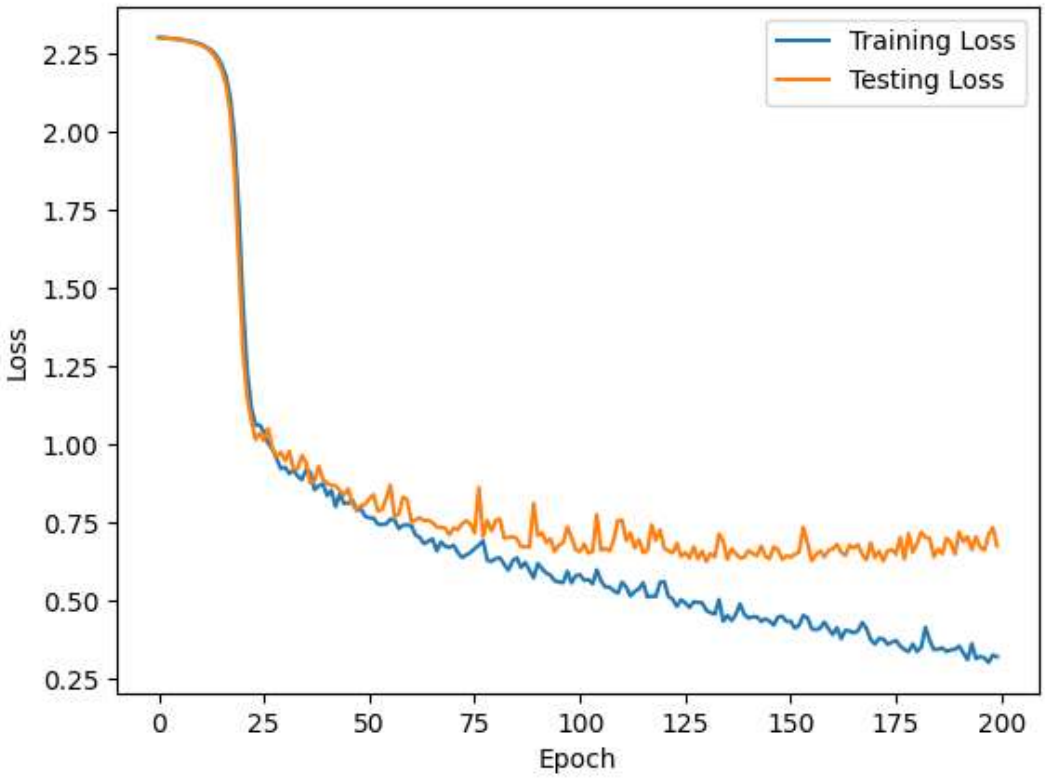
PCA

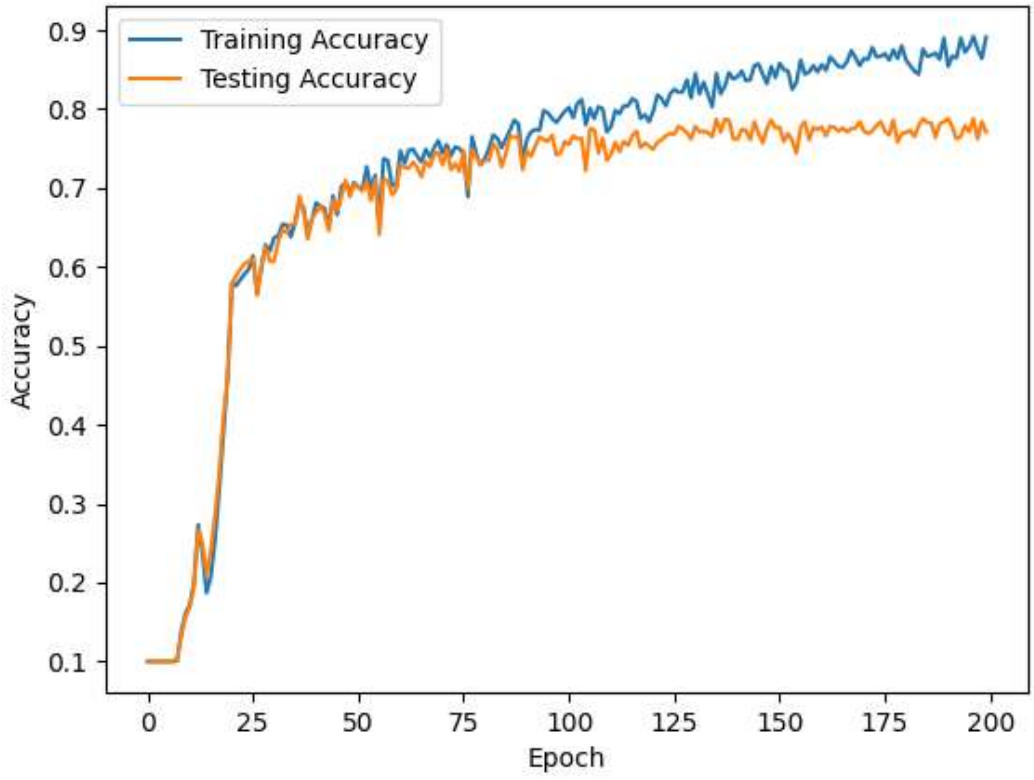

PCA Visualization
30
20
10
0
−10
−20
−30
−40
−20
0
20
40

t-SNE

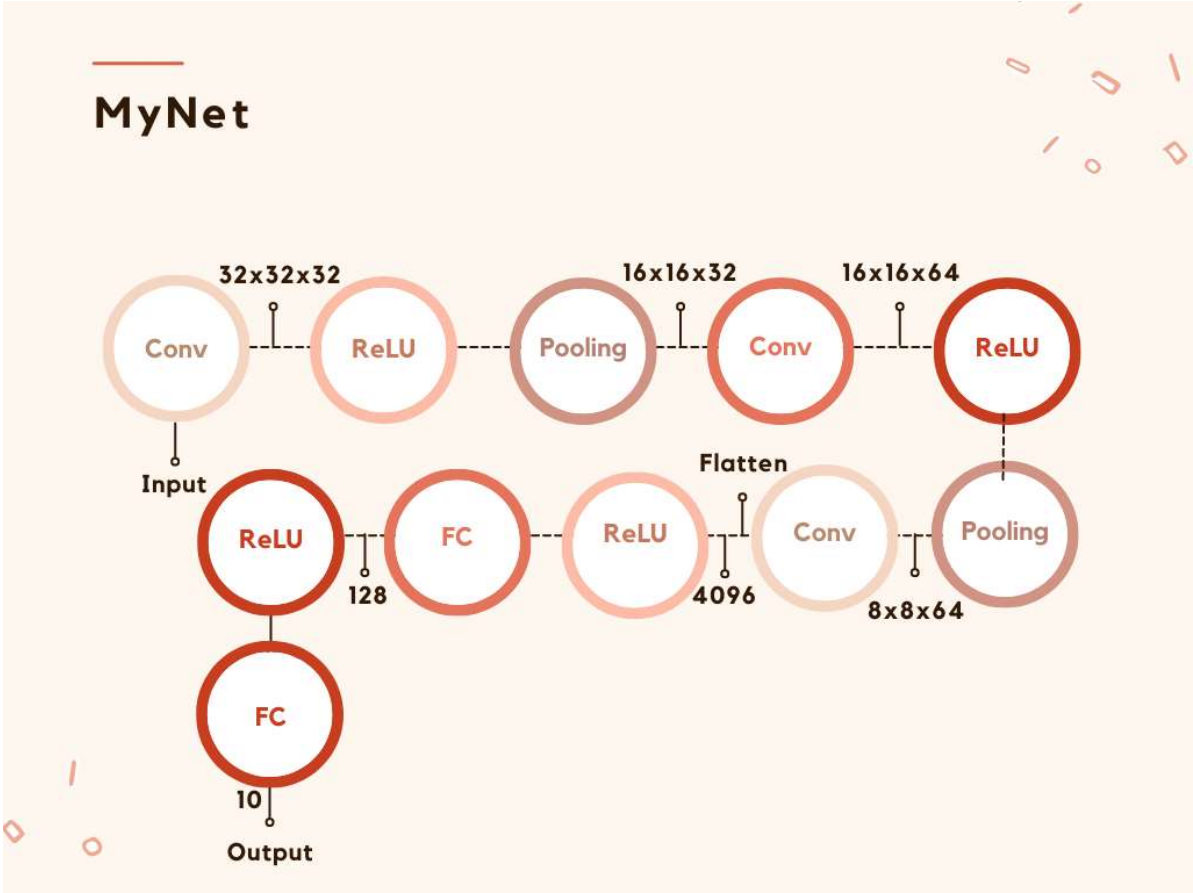

t-SNE Visualization
15
10
5
0
−5
−10
−15
−15
−10
−5
0
5
10
15

Loss Curve & Accuracy Curve





MyNet



The output accuracy: 84%

Modifications:

1. Increased the number of filters in the first convolutional layer from 6 to 32:
   The original LeNet architecture was designed for low-resolution grayscale images, such as the MNIST dataset. For higher-resolution images, such as the 32x32 grayscale images used in this example, it's generally a good idea to increase the number of filters in the first convolutional layer to capture more features in the input images. By increasing the number of filters to 32 in the first convolutional layer, the model is able to learn more complex features in the input images.
2. Added batch normalization after each convolutional layer:
   Batch normalization is a technique that standardizes the inputs to each layer in a neural network, which can help improve convergence and prevent overfitting. By adding batch normalization after each convolutional layer, the model is able to learn more efficiently and generalize better to new data.
3. Added a third convolutional layer with 64 filters:
   By adding a third convolutional layer with 64 filters, the model is able to capture even more complex features in the input images. This can help improve the model's accuracy on more challenging datasets.
4. Increased the number of units in the first fully connected layer from 120 to 128:
   The original LeNet architecture had a first fully connected layer with 120 units. However, since we increased the number of filters in theconvolutional layers, it's a good idea to also increase the number of units in the first fully connected layer to account for the increased number of features. By increasing the number of units to 128, the model is able to learn more complex representations of the input data, which can help improve its accuracy.

Overall, the modifications made to the original LeNet architecture were designed to improve the model's ability to learn complex features from higher-resolution grayscale images, while also improving convergence and preventing overfitting. By increasing the number of filters, adding batch normalization, and increasing the number of units in the first fully connected layer, the model is able to achieve good accuracy on the given dataset.

Hyperparameter specifications:

conv1: channels = 32, kernel size = 3, padding = 1, stride = 1

conv2: channels = 64, kernel size = 3, padding = 1, stride = 1

conv3: channels = 64, kernel size = 3, padding = 1, stride = 1 (followed with a flatten operation)

maxpooling: kernel size = 2

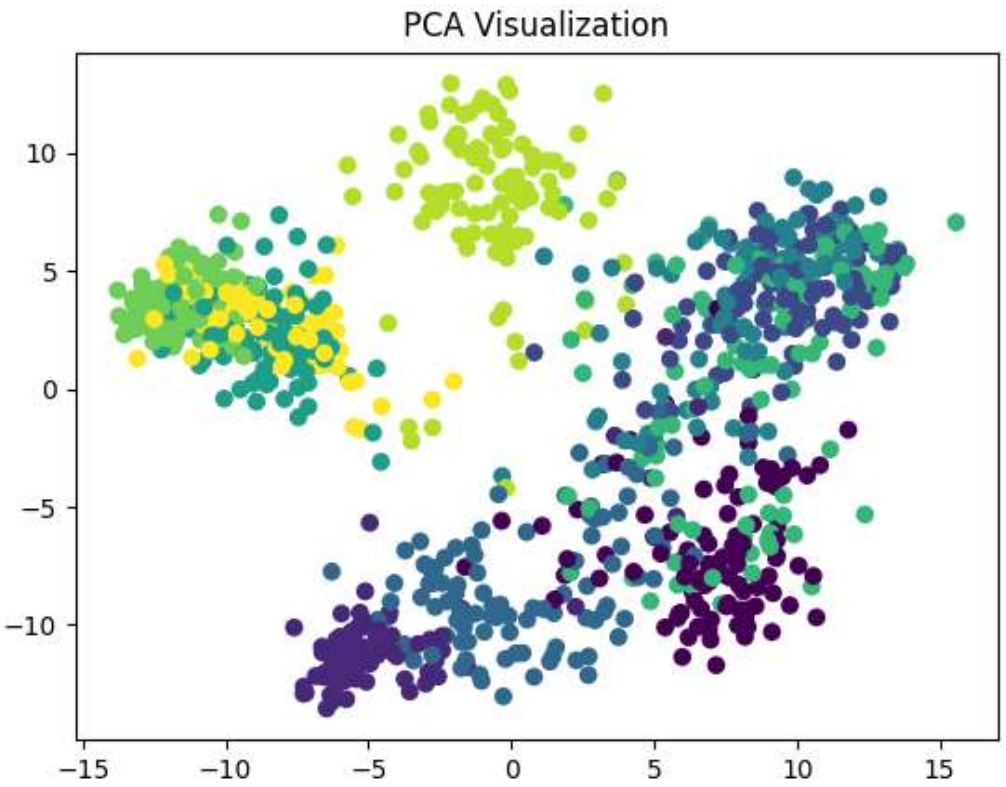
fc1: in_features = 4096, out_features = 128

fc2: in_features = 128, out_features = 10

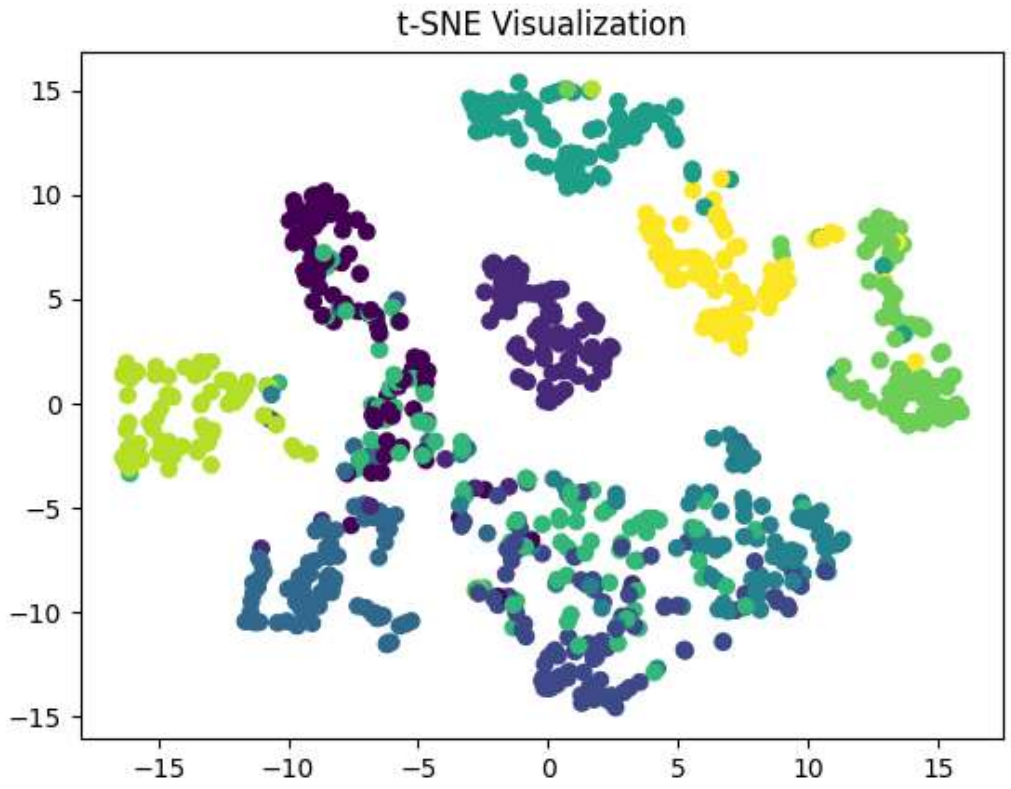
Output (the complete output image set is in the attachment)

Visualization

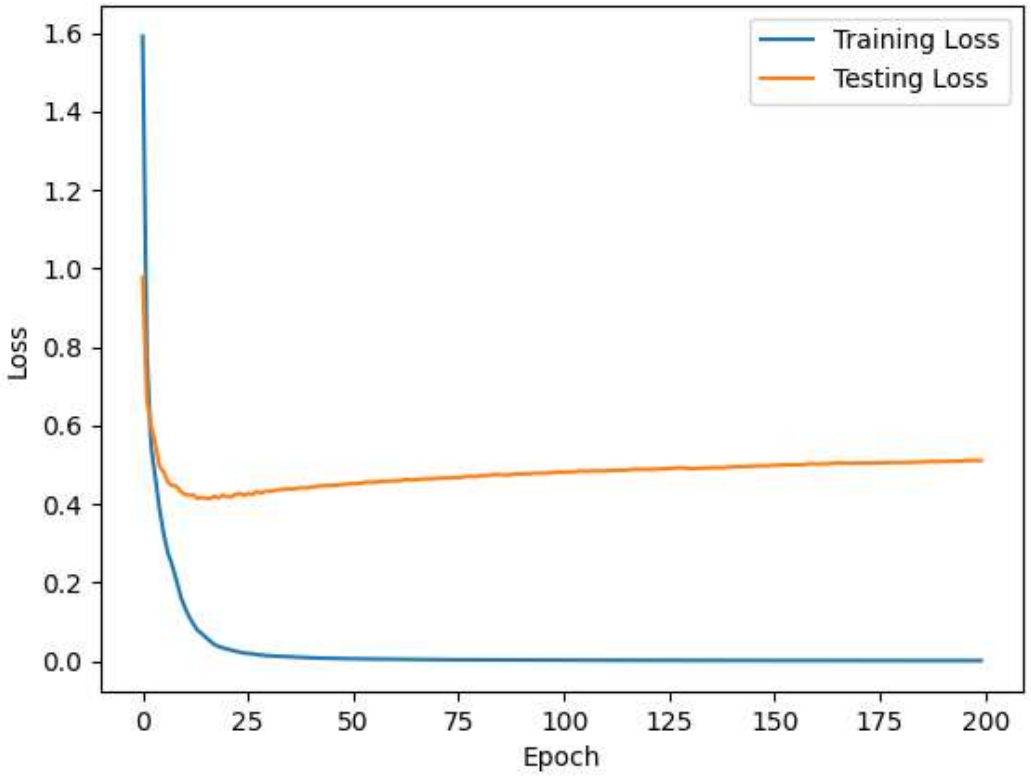
PCA

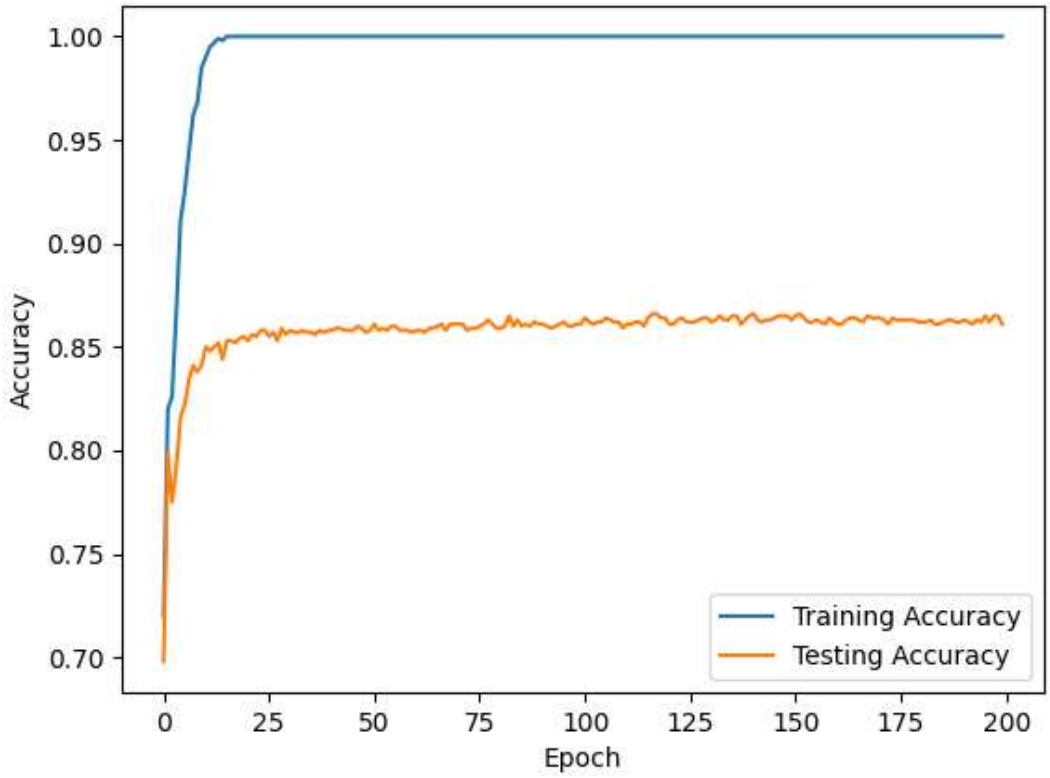


t-SNE



Loss Curve & Accuracy Curve





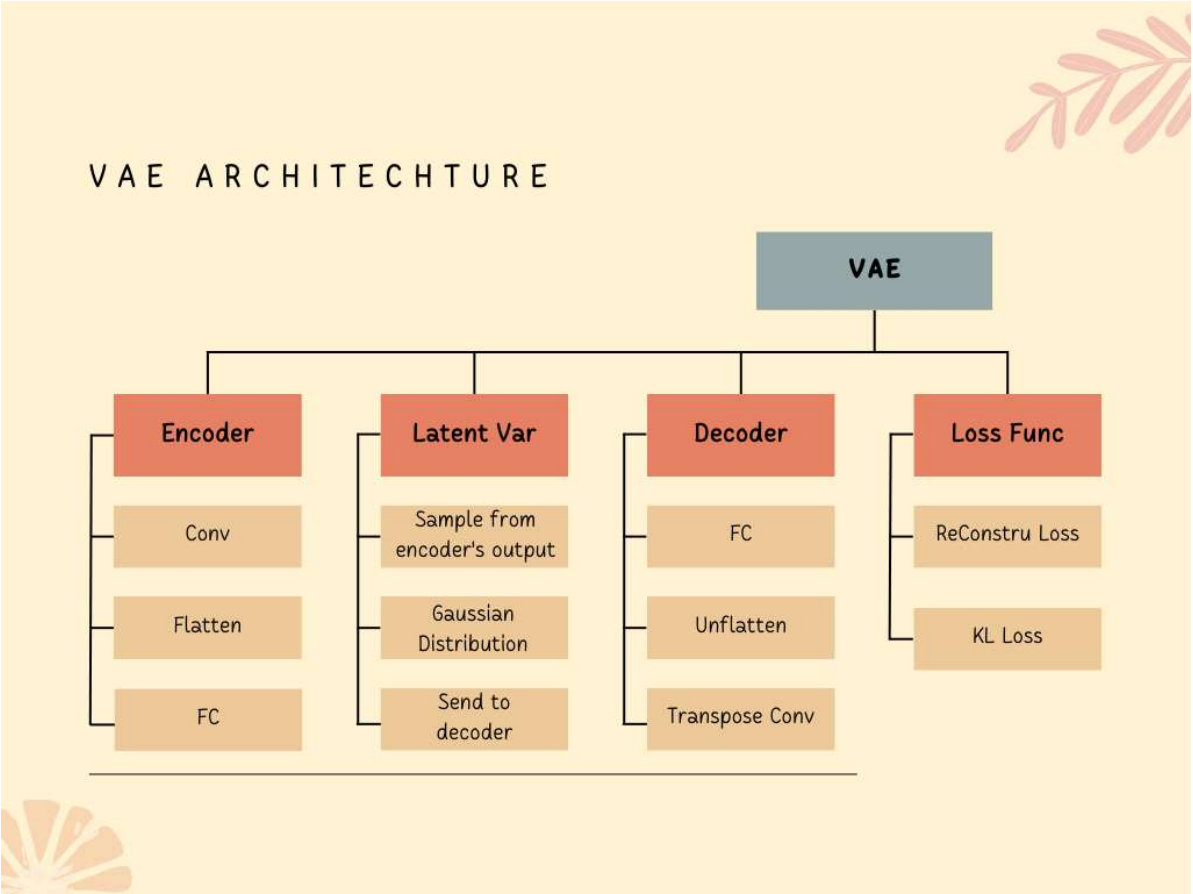
Some empirical conclusions

For the learning net part, the modified net seems to have a better performance than the original LeNet. This is the probably because in convolution layers the number of channels for the output is increased and the size of the kernel is smalled to capture more information. As a payment, the newly designed net has the speed dropped to process more data, leading to much more running time. Besides, the learning becomes faster per epoch, this is possibly due to the application of batch normalization, which has a function of improving training convergence and highering learning rate.

For the visualization part, According to the output, the visualization of PCA seems to possess a relatively good linearity, with the bounds among classes looking quite linear, while it is not the case for t-SNE, and t-SNE has a poor linearity performance. However, t-SNE shows a better capability of classification and it seems to distinguish different classes better, while PCA seems to have much more overlap between classes.

Image reconstruction

Arch



The VAE is constructed in the following way:

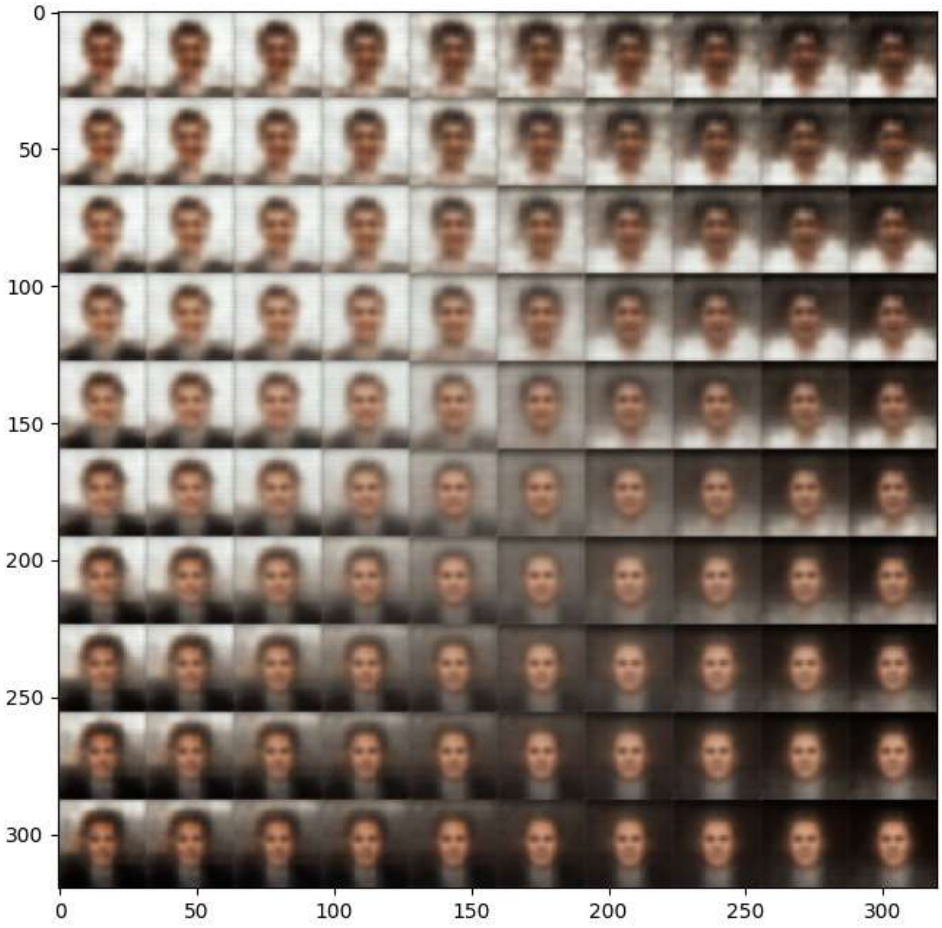
1. Encoder Network: containing 3 convolutional layers, 1 flatten layer and 1 FC layer.
2. Latent Variables: the latent variables are sampled from a Gaussian distribution with mean and standard deviation equal to the output of the previous fully connected layer.
3. Decoder Network: containing 3 transpose convolutional layers, 1 unflatten layer and 1 FC layer.
4. Loss Function: the loss function is made of two parts - the reconstruction loss is the binary cross-entropy between the input image and the output image; the KL-divergence loss is the KL-divergence between the distribution of the latent variables and a standard normal distribution.
5. Training: the VAE is trained by minimizing the sum of the reconstruction loss and the KL-divergence loss using the Adam optimizer.
6. Sampling: to generate a new image, sample from a Gaussian distribution with mean and standard deviation equal to the output of the encoder network, and pass the resulting latent variables through the decoder network to obtain the generated image.

Implementation

1. Applying pyTorch to construct the encoder network and the decoder network.
2. Using cross entropy to get the reconstruction loss
3. Get the KL loss and combine the two parts of the loss
4. Set the optimizer
5. Set the number of epochs (and some other parameters) and start training.

Output (the complete output image set is in the attachment)

the visualization of the reconstructed image



the visualization of generated images applying linear interpolation