# AutoEncoder 보고서

컴퓨터소프트웨어학부

2016025996

조성진

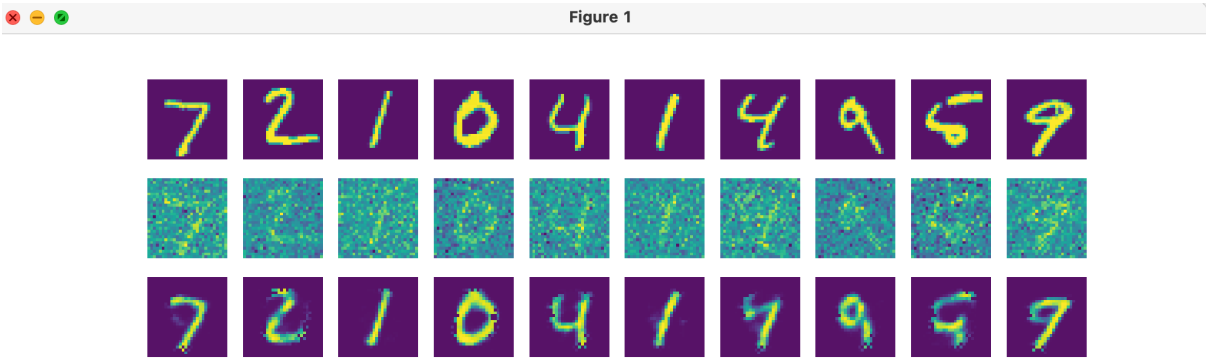
Assignment2는 autoencoder을 mnist dataset에 noise를 추가하고 DAE를 잘 수행하는지 확인하는 과제이다.

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("./mnist/data/", one_hot=True)
batch_size = 100
learning_rate = 0.01
epoch_num = 20
n_input = 28*28
n_hidden1 = 256
n_hidden2 = 128
noise_level = 0.6

X_noisy = tf.placeholder(tf.float32, [None, n_input])
Y = tf.placeholder(tf.float32, [None, n_input])
W_encode1 = tf.Variable(tf.random_uniform([n_input, n_hidden1], -1., 1.))
b_encode1 = tf.Variable(tf.random_uniform([n_hidden1], -1., 1.))
encoder_h1 = tf.nn.sigmoid(tf.add(tf.matmul(X_noisy, W_encode1), b_encode1))
W_encode2 = tf.Variable(tf.random_uniform([n_hidden1, n_hidden2], -1., 1.))
b_encode2 = tf.Variable(tf.random_uniform([n_hidden2], -1., 1.))
encoder_h2 = tf.nn.sigmoid(tf.add(tf.matmul(encoder_h1, W_encode2), b_encode2))
W_decode1 = tf.Variable(tf.random_uniform([n_hidden2, n_hidden1], -1., 1.))
b_decode1 = tf.Variable(tf.random_uniform([n_hidden1], -1., 1.))
decoder_h1 = tf.nn.sigmoid(tf.add(tf.matmul(encoder_h2, W_decode1), b_decode1))

W_decode2 = tf.Variable(tf.random_uniform([n_hidden1, n_input], -1., 1.))
b_decode2 = tf.Variable(tf.random_uniform([n_input], -1., 1.))
output = tf.nn.sigmoid(tf.add(tf.matmul(decoder_h1, W_decode2), b_decode2))


cost = tf.reduce_mean(tf.square(Y - output))
optimizer = tf.train.AdamOptimizer(learning_rate).minimize(cost)
```

우선 mnist dataset을 불러오고, hyperparameter값들을 batch size = 100, lr = 1e-2, epoch = 20 으로 설정하고 첫번째 hidden layer 의 노드 갯수를 256개, 두번째 hidden layer의 노드 갯수를 128개로 하고, 가우시안 노이즈를 만들때 노이즈 레벨을 0.6으로 했다. 층은 mnist classifier때 만든것처럼 하는데, 각 과정에 feature vector을 생성하는 encoding과정이 있고, 그걸 원래의 이미지로 복원해내는 decoding과정이 있다. Encoding에 2개의 layer과 decoding에 2개의 layer을 사용했다.

```
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    total_batch = int(mnist.train.num_examples/batch_size)
....
    for epoch in range(epoch_num):
        avg_cost = 0
        for i in range(total_batch):
            batch_xs, batch_ys = mnist.train.next_batch(batch_size)
            batch_x_noisy = batch_xs + noise_level * np.random.normal(loc=0.0, scale=1.0, size=batch_xs.shape)
            _, cost_val = sess.run([optimizer, cost], feed_dict={X_noisy: batch_x_noisy, Y:batch_xs})
            avg_cost += cost_val / total_batch
        print('Epoch:', '%d' % (epoch+1), 'cost:', '{:.9f}'.format(avg_cost))
....
    test_X = mnist.test.images[:10] + noise_level * np.random.normal(loc=0.0, scale=1.0, size=mnist.test.images[:10].shape)
    samples = sess.run(output, feed_dict={X_noisy: test_X})
    fig, ax = plt.subplots(3, 10, figsize=(10, 3))
....
    for i in range(10):
        ax[0][i].set_axis_off()
        ax[1][i].set_axis_off()
        ax[2][i].set_axis_off()
        ax[0][i].imshow(np.reshape(mnist.test.images[i], (28, 28)))
        ax[1][i].imshow(np.reshape(test_X[i], (28, 28)))
        ax[2][i].imshow(np.reshape(samples[i], (28, 28)))
........
    plt.show()
```

그리고 epoch수만큼 train하면서, random한 noise를 원본데이터에 추가해 인풋으로 주고 Y값은 원본 데이터로 주어 DAE의 학습을 시킨다. 마지막 부분은 ax[0]은 원본, ax[1]은 noise가 추가된 데이터, ax[2]는 복원된 데이터를 의미하며 결과는 다음과 같다.




```
Epoch: 1 cost: 0.065606020
Epoch: 2 cost: 0.048335081
Epoch: 3 cost: 0.043050555
Epoch: 4 cost: 0.040118103
Epoch: 5 cost: 0.038286377
Epoch: 6 cost: 0.036936735
Epoch: 7 cost: 0.035825843
Epoch: 8 cost: 0.035034601
Epoch: 9 cost: 0.034540876
Epoch: 10 cost: 0.034031185
Epoch: 11 cost: 0.033772338
Epoch: 12 cost: 0.033387455
Epoch: 13 cost: 0.033257668
Epoch: 14 cost: 0.032805803
Epoch: 15 cost: 0.032636979
Epoch: 16 cost: 0.032399505
Epoch: 17 cost: 0.032304833
Epoch: 18 cost: 0.032192334
Epoch: 19 cost: 0.032071325
Epoch: 20 cost: 0.031958794
```

잘 복원된 결과를 확인할 수 있었다.