Age Estimation with Autoencoder and CNN

Shihchia Chen, #50207079

Jiawei Zhao, #50207069

12/3/2017

Abstract

- Employed Autoencoder for feature extraction before feed the data into the neural network.
- Applied CNN for age estimation with two convolutional layer, two max pooling layer, and one fully connected layer and an output layer.
- Convolution Layer -> Max Pooling Layer -> Convolution Layer -> Max Pooling Layer -> Fully Connected Layer -> Output Layer
- Dataset: IMDB-Wiki facial images with age and gender labels

Core Code Snippet I
Autoencoder

```python
def encodeData(data):
    # Construct model
    encoder_op = encoder(X)
    decoder_op = decoder(encoder_op)

    # Prediction
    y_pred = decoder_op
    # Targets (Labels) are the input data.
    y_true = X

    cost = tf.reduce_mean(tf.pow(y_true - y_pred, 2))
    optimizer = tf.train.AdamOptimizer(learning_rate).minimize(cost)

    with tf.Session() as sess:
        init = tf.global_variables_initializer()
        sess.run(init)
        total_batch = int(len(data)/batch_size)

        for epoch in range(training_epochs):
            for i in range(total_batch):
                batch_xs = get_next_batch(data, batch_size, i)  # max(x) = 1, min(x) = 0
                _, c = sess.run([optimizer, cost], feed_dict={X: batch_xs})
            if epoch % display_step == 0:
                print("Epoch:", '%04d' % (epoch+1),
                      "cost=", "{:.9f}".format(c))

        print("Optimization Finished!")

        encode_decode = sess.run(
            y_pred, feed_dict={X: data})

        return encode_decode
```

```python
def encoder(x):
    layer_1 = tf.nn.sigmoid(tf.add(tf.matmul(x, weights['encoder_h1']),
                                   biases['encoder_b1']))
    layer_2 = tf.nn.sigmoid(tf.add(tf.matmul(layer_1, weights['encoder_h2']),
                                   biases['encoder_b2']))
    layer_3 = tf.nn.sigmoid(tf.add(tf.matmul(layer_2, weights['encoder_h3']),
                                   biases['encoder_b3']))
    layer_4 = tf.nn.sigmoid(tf.add(tf.matmul(layer_3, weights['encoder_h4']),
                                   biases['encoder_b4']))
    layer_5 = tf.add(tf.matmul(layer_4, weights['encoder_h5']),
                                    biases['encoder_b5'])
    return layer_5

def decoder(x):
    layer_1 = tf.nn.sigmoid(tf.add(tf.matmul(x, weights['decoder_h1']),
                                   biases['decoder_b1']))
    layer_2 = tf.nn.sigmoid(tf.add(tf.matmul(layer_1, weights['decoder_h2']),
                                   biases['decoder_b2']))
    layer_3 = tf.nn.sigmoid(tf.add(tf.matmul(layer_2, weights['decoder_h3']),
                                   biases['decoder_b3']))
    layer_4 = tf.nn.sigmoid(tf.add(tf.matmul(layer_3, weights['decoder_h4']),
                                   biases['decoder_b4']))
    layer_5 = tf.nn.sigmoid(tf.add(tf.matmul(layer_4, weights['decoder_h5']),
                                   biases['decoder_b5']))
    return layer_5
```

Core Code Snippet II
Convolutional Neural Network

```
def conv_neural_network(x):
    weights = {
                'W_conv1': tf.Variable(tf.random_normal([5,5,1,32])),
                'W_conv2': tf.Variable(tf.random_normal([5,5,32,64])),
                'W_fc': tf.Variable(tf.random_normal([25*25*64,1024])),  #this 1
                'output': tf.Variable(tf.random_normal([1024,n_classes]))
              }

    biases = {
                'b_conv1': tf.Variable(tf.random_normal([32])),
                'b_conv2': tf.Variable(tf.random_normal([64])),
                'b_fc': tf.Variable(tf.random_normal([1024])),
                'output': tf.Variable(tf.random_normal([n_classes]))
              }

    x = tf.reshape(x, [-1,100,100,1])

    conv1 = tf.nn.relu(conv2d(x, weights['W_conv1'])+biases['b_conv1'])
    conv1 = maxpool2d(conv1)

    conv2 = tf.nn.relu(conv2d(conv1, weights['W_conv2'])+biases['b_conv2'])
    conv2 = maxpool2d(conv2)

    fc = tf.reshape(conv2, [-1, 25*25*64])
    fc = tf.nn.relu(tf.matmul(fc, weights['W_fc'])+biases['b_fc'])

    output = tf.matmul(fc, weights['output'])+biases['output']

    return output
```

Result and conclusion

- Output could be in a large range, from 0 to 99. So the accuracy would be awful for exact prediction. So we take an age range as correct prediction.
- We compare the prediction accuracy with the first project (without autoencoder) with the second project (with autoencoder)

Accuracy Table:

Range of age	[age-1, age+1]	[age-3, age+3]	[age-5, age+5]
Accuracy (No autoencoder)	76.55%	80.62%	95.48%
Accuracy (with autoencoder)	65.37%	69.77%	97.73%

Result and conclusion(cont.)

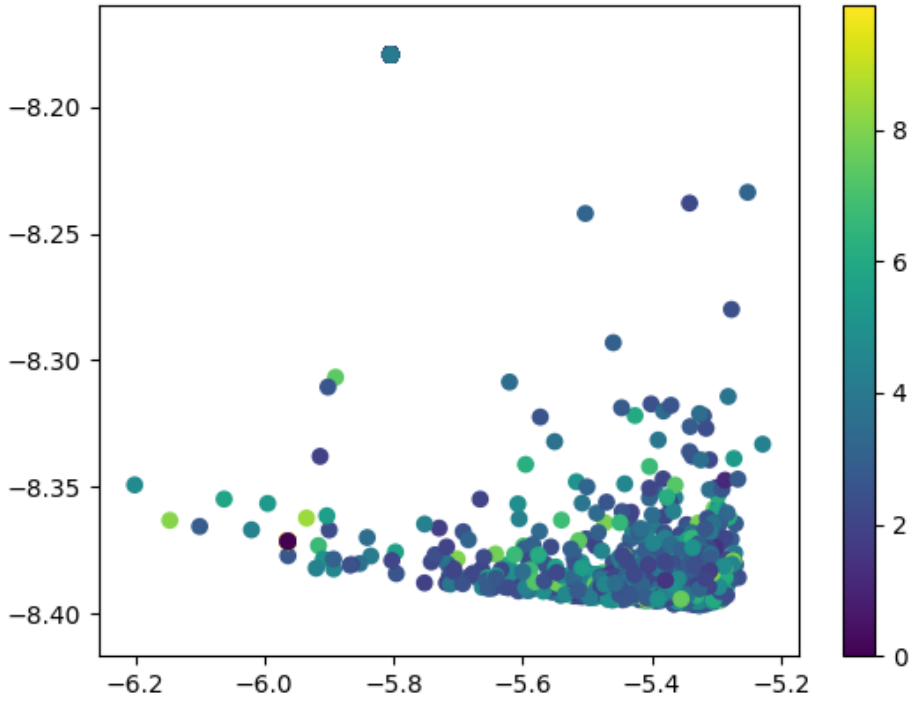

Figure1. Encoder Result

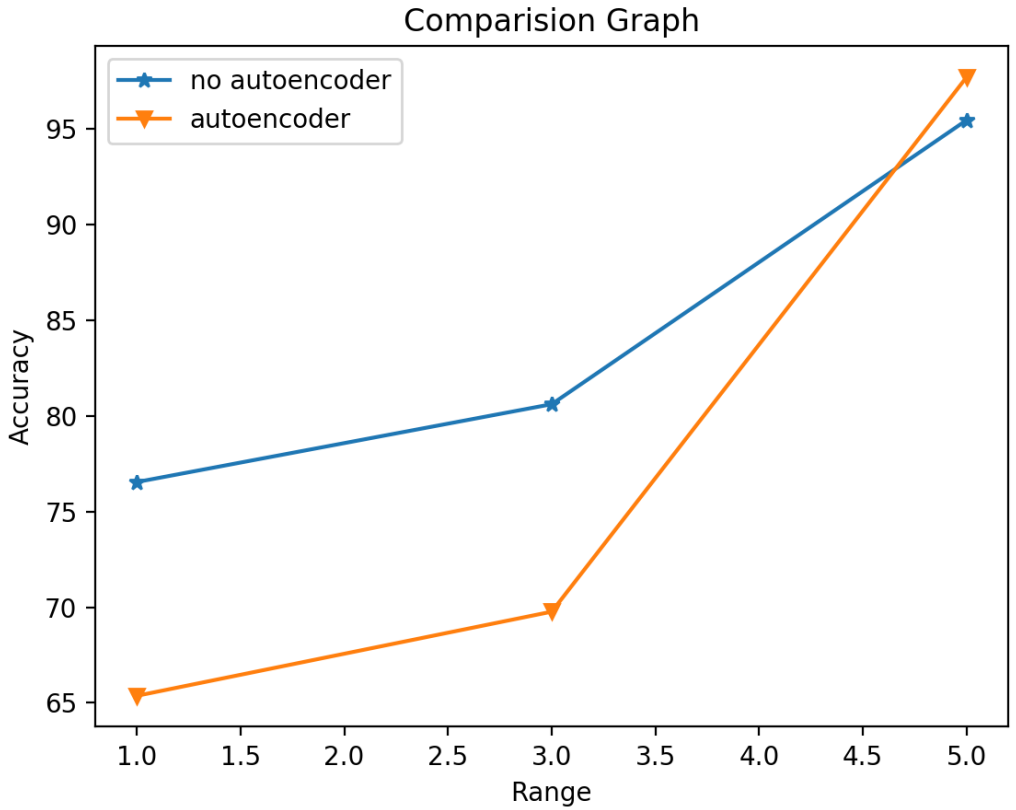

Figure2. Line Chart Result

Result and conclusion(cont.)

Advantage and disadvantage of Autoencoder:

- Pros:
  - Autoencoder network is handy and easy to build
  - Autoencoder is a great way for feature extraction
  - Autoencoder is an efficient network for dimention reduction
- Cons:
  - If the data isn't representative, then the autoencoder may obscure the information rather than clarify it
  - An autoencoder learns to capture as much information as possible rather than as relevant information as possible, which can lead to misleading results

In a nutshell, we think that it's good for us to employ autoencoder if our data is specific and doesn't have too much noise, and applying autoencoder is very likely to increase our neural network performance, on the other hand, if the data is big and general, or contains too much unnecessary information, it's not recommended to apply an autoencoder beforehand since that may result in loss of truly valuable information and aggregate unnecessary or even misleading information.