6000B project2

Lin Jianxia 20459661

Description:

Using the labeled training data to build a deep model to predict different classes of flowers according to the pictures.

I design two model to fit this data,VGGNet-16 and Simple CNN model.

The deep model: VGGNet-16

Feature extraction:

```
def add_cnn_layers(name,input,shape,stride):
    with tf.variable_scope(name):
        conv_weights = tf.get_variable('weight',shape,initializer=tf.truncated_normal_initializer(stddev=0.1))
        conv_biases = tf.get_variable('bias', [shape[3]], initializer=tf.constant_initializer(0.0))
        conv = tf.nn.conv2d(input, conv_weights, strides=stride, padding='SAME')
        relu = tf.nn.relu(tf.nn.bias_add(conv, conv_biases))
    return relu

def add_pooling_layers(name,input):
    with tf.name_scope(name):
        pool = tf.nn.max_pool(input, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='VALID')
        return pool
```

```
#conv1
conv1_1 = add_cnn_layers('conv1.1',x,[3,3,3,64],[1,1,1,1])
conv1_2 = add_cnn_layers('conv1.2',conv1_1,[3,3,64,64],[1,1,1,1])
pool1 = add_pooling_layers('pool1',conv1_2)

#conv2
conv2_1 = add_cnn_layers('conv2.1',pool1,[3,3,64,128],[1,1,1,1])
conv2_2 = add_cnn_layers('conv2.2',conv2_1,[3,3,128,128],[1,1,1,1])
pool2 = add_pooling_layers('pool2',conv2_2)

#conv3
conv3_1 = add_cnn_layers('conv3.1',pool2,[3,3,128,256],[1,1,1,1])
conv3_2 = add_cnn_layers('conv3.2',conv3_1,[3,3,256,256],[1,1,1,1])
conv3_3 = add_cnn_layers('conv3.3',conv3_2,[3,3,256,256],[1,1,1,1])
pool3 = add_pooling_layers('pool3',conv3_3)

#conv4
conv4_1 = add_cnn_layers('conv4.1',pool3,[3,3,256,512],[1,1,1,1])
conv4_2 = add_cnn_layers('conv4.2',conv4_1,[3,3,512,512],[1,1,1,1])
conv4_3 = add_cnn_layers('conv4.3',conv4_2,[3,3,512,512],[1,1,1,1])
pool4 = add_pooling_layers('pool4',conv4_3)

#conv5
conv5_1 = add_cnn_layers('conv5.1',pool4,[3,3,512,512],[1,1,1,1])
conv5_2 = add_cnn_layers('conv5.2',conv5_1,[3,3,512,512],[1,1,1,1])
conv5_3 = add_cnn_layers('conv5.3',conv5_2,[3,3,512,512],[1,1,1,1])
pool5 = add_pooling_layers('pool5',conv5_3)
```

By using 13 convolution layers and 5 max pooling layers to do feature extraction, and all the convolution matrix is 3×3.

Label prediction：

The use 3 full connection layers to predict the label of each picture.

```
def ful_con_layers(name,input,input_size,output_size):
    with tf.variable_scope(name):
        weights = tf.get_variable('weight', [input_size, output_size],initializer=tf.truncated_normal_initializer(stddev=0.1))
        biases = tf.get_variable('bias', [output_size], initializer=tf.constant_initializer(0.1))
        f = tf.nn.relu(tf.matmul(input, weights) + biases)
        # f = tf.nn.dropout(f, 0.5)
        return f,weights

def ful_con_layer(name,input,input_size,output_size):
    with tf.variable_scope(name):
        weights = tf.get_variable('weight', [input_size, output_size],initializer=tf.truncated_normal_initializer(stddev=0.1))
        biases = tf.get_variable('bias', [output_size], initializer=tf.constant_initializer(0.1))
        f = tf.matmul(input, weights) + biases
        # if train: f = tf.nn.dropout(f, 0.5)
        return f,weights
```

```
nodes = 7*7*512
input_data = tf.reshape(pool5,[-1,nodes])
ful1,weights1 = ful_con_layers('ful1',input_data,nodes,1024)
ful2,weights2 = ful_con_layers('ful2',ful1,1024,1024)
ful3,weights3 = ful_con_layer('ful3',ful2,1024,5)
```

Loss function: Cross-entropy

```
loss=tf.nn.sparse_softmax_cross_entropy_with_logits(logits=ful3, labels=y)+0.001*(tf.nn.l2_loss(weights1)+tf.nn.l2_loss(weights2))
```

Mini—batch:

Using mini-batch to update the parameter and calculate the training accuracy and test accuracy.

```
def minibatches(inputs=None, targets=None, batch_size=None, shuffle=False):
    assert len(inputs) == len(targets)
    if shuffle:
        indices = np.arange(len(inputs))
        np.random.shuffle(indices)
    for start_idx in range(0, len(inputs) - batch_size + 1, batch_size):
        if shuffle:
            excerpt = indices[start_idx:start_idx + batch_size]
        else:
            excerpt = slice(start_idx, start_idx + batch_size)
        yield inputs[excerpt], targets[excerpt]
```

Data processing:

Randomly choose 80% raw data as training data and 20% as test data.

```
num_example=data.shape[0]
arr=np.arange(num_example)
np.random.shuffle(arr)
data=data[arr]
label=label[arr]

ratio=0.8
s=np.int(num_example*ratio)
x_train=data[:s]
y_train=label[:s]
x_val=data[s:]
y_val=label[s:]
```

Set each image as a $224 \times 224 \times 3$ matrix.

```
width = 224
high = 224
color = 3
data, label = read_img(datapath,width,high)
```

```
def read_img(path,width,high):
    flower_cat = [path + x for x in os.listdir(path) if os.path.isdir(path + x)]
    del flower_cat[1]
    imgs = []
    labels = []
    for index, folder in enumerate(flower_cat):
        print index
        print folder
        for pic in glob.glob(folder + '/*.jpg'):
            img = io.imread(pic)
            img = transform.resize(img,(width,high))
            imgs.append(img)
            labels.append(index)
    print len(labels)
    return np.asarray(imgs,np.float),np.asarray(labels,np.int32)
```

It costs so much time to calculate, so I did not get the accuracy of this VGG-Net model accuracy.

The deep model: Simple CNN

Feature extraction:

```
relu1 = add_cnn_layers('conv1',x,[5,5,3,32],[1,1,1,1])
pool1 = add_pooling_layers('pool1',relu1)
relu2 = add_cnn_layers('conv2',pool1,[5,5,32,64],[1,1,1,1])
pool2 = add_pooling_layers('pool2',relu2)
relu3 = add_cnn_layers('conv3',pool2,[5,5,64,128],[1,1,1,1])
pool3 = add_pooling_layers('pool3',relu3)
relu4 = add_cnn_layers('conv4',pool3,[3,3,128,128],[1,1,1,1])
pool4 = add_pooling_layers('pool4',relu4)
relu5 = add_cnn_layers('conv5',pool4,[3,3,128,128],[1,1,1,1])
pool5 = add_pooling_layers('pool5',relu5)
```

By using 5 convolution layers and 5 max pooling layers to do feature extraction, and there 3 the convolution matrix are 5×5, 2 the convolution matrix are 3×3.

Other processing is similar to VGG-Net, and the test accuracy of this simple CNN classifier is 64%.