项目实践

- 数据准备
- VGG16的Tensorflow实现
  - 定义功能函数
  - 定义VGG16模型类
- VGG16模型复用
  - 微调
  - 载入权重
- 数据输入
- 模型重新训练与保存
- 预测

数据输入

```
def get_file(file_dir):
    images = []
    temp = []
    for root, sub_folders, files in os.walk(file_dir):
        for name in files:
            images.append(os.path.join(root, name))
        for name in sub_folders:
            temp.append(os.path.join(root, name))
            labels = []
    for one_folder in temp:
        n_img = len(os.listdir(one_folder))
        letter = one_folder.split('/')[-1]
        if letter == 'cat':
            labels = np.append(labels, n_img * [0])
        else:
            labels = np.append(labels, n_img * [1])

    # shuffle
    temp = np.array([images, labels])
    temp = temp.transpose()
    np.random.shuffle(temp)
    image_list = list(temp[:, 0])
    label_list = list(temp[:, 1])
    label_list = [int(float(i)) for i in label_list]

    return image_list, label_list
```

数据输入

```
from vgg_preprocess import preprocess_for_train

img_width = 224
img_height = 224

def get_batch(image_list, label_list,img_width,img_height,batch_size,capacity):#通过读取列表来载入批量图片及标签

    image = tf.cast(image_list,tf.string)
    label = tf.cast(label_list,tf.int32)
    input_queue = tf.train.slice_input_producer([image,label])
    label = input_queue[1]
    image_contents = tf.read_file(input_queue[0])

    image = tf.image.decode_jpeg(image_contents,channels=3)
    image = preprocess_for_train(image,224,224)
    image_batch,label_batch = tf.train.batch([image,label],batch_size=batch_size,num_threads=64,capacity=capacity)
    label_batch = tf.reshape(label_batch,[batch_size])

    return image_batch,label_batch
```

标签格式的重构

```
def onehot(labels):
    n_sample = len(labels)
    n_class = max(labels) + 1
    onehot_labels = np.zeros((n_sample, n_class))
    onehot_labels[np.arange(n_sample), labels] = 1
    return onehot_labels
```

项目实践

- 数据准备
- VGG16的Tensorflow实现
  - 定义功能函数
  - 定义VGG16模型类
- VGG16模型复用
  - 微调
  - 载入权重
- 数据输入
- 模型重新训练与保存
- 预测

模型重新训练与保存


```
import os
import tensorflow as tf
from time import time
import VGG16_model as model
import utils
```

```
startTime=time()
batch_size=32
capacity=256 #内存中存储的最大数据容量
means = [123.68, 116.779, 103.939] #VGG训练时图像预处理所减均值(RGB三通道)
```

模型重新训练与保存


```
xs, ys = utils.get_file("./data/train/")# 获取图像列表和标签列表
image_batch, label_batch = utils.get_batch(xs, ys, 224, 224, batch_size,capacity)# 通过读取列表来载入批量图片及标签

x = tf.placeholder(tf.float32, [None, 224, 224, 3])
y = tf.placeholder(tf.int32, [None, 2])# 对“猫”和“狗”两个类别进行判定

vgg = model.vgg16(x)
fc8_finetuining = vgg.probs # 即softmax(fc8)
loss_function = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=fc8_finetuining, labels=y)) #损失函数
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.001).minimize(loss_function) #优化器

sess = tf.Session()
sess.run(tf.global_variables_initializer())
vgg.load_weights('./vgg16/vgg16_weights.npz',sess)# 通过npz格式的文件获取VGG的相应权重参数，从而将权重注入即可实现复用
saver = tf.train.Saver()
```

微调（finetuining）

```
coord = tf.train.Coordinator() #使用协调器Coordinator来管理线程
threads = tf.train.start_queue_runners(coord=coord,sess=sess)
```

启动线程


```
epoch_start_time = time()

for i in range(1000):

    images, labels = sess.run([image_batch, label_batch])
    labels = utils.onehot(labels)#用one-hot形式对标签进行编码

    sess.run(optimizer, feed_dict={x: images, y: labels})
    loss = sess.run(loss_function, feed_dict={x: images, y: labels})
    print("Now the loss is %f "%loss)

    epoch_end_time = time()
    print('Current epoch takes: ', (epoch_end_time - epoch_start_time))
    epoch_start_time = epoch_end_time

    if (i+1) % 500 == 0:
        saver.save(sess, os.path.join("./model/", 'epoch_{:06d}.ckpt'.format(i)))
    print("----------Epoch %d is finished--------------"%i)
```

迭代训练

```
saver.save(sess,"./model/")
print("Optimization Finished!")
```

模型保存

```
duration =time()-startTime
print("Train Finished takes:","{:.2f}".format(duration))
```

```
coord.request_stop()#通知其他线程关闭
coord.join(threads)#join操作等待其他线程结束,其他所有线程关闭之后,这一函数才能返回
```

关闭线程

```
saver.save(sess,"./model/")
print("Optimization Finished!")

duration =time()-startTime
print("Train Finished takes:","{:.2f}".format(duration))

coord.request_stop()#通知其他线程关闭
coord.join(threads)#join操作等待其他线程结束,其他所有线程关闭之后,这一函数才能返回
```

```
Current epoch takes:  0.1919872760772705
----------Epoch 994 is finished--------------
Now the loss is 0.379803
Current epoch takes:  0.2115015983581543
----------Epoch 995 is finished--------------
Now the loss is 0.404790
Current epoch takes:  0.19910168647766113
----------Epoch 996 is finished--------------
Now the loss is 0.362488
Current epoch takes:  0.20191287994384766
----------Epoch 997 is finished--------------
Now the loss is 0.328630
Current epoch takes:  0.20640087127685547
----------Epoch 998 is finished--------------
Now the loss is 0.349333
Current epoch takes:  0.2050466537475586
----------Epoch 999 is finished--------------
Optimization Finished!
Train Finished takes: 221.29
```

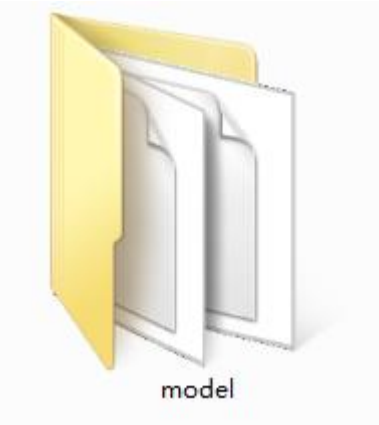
每一次迭代显示信息

训练过程结束

项目实践

- 数据准备
- VGG16的Tensorflow实现
  - 定义功能函数
  - 定义VGG16模型类
- VGG16模型复用
  - 微调
  - 载入权重
- 数据输入
- 模型重新训练与保存
- 预测

预测



- .data-00000-of-00001
- .index
- .meta
- checkpoint
- epoch_000200.ckpt.data-00000-of-00001
- epoch_000200.ckpt.index
- epoch_000200.ckpt.meta
- epoch_000400.ckpt.data-00000-of-00001
- epoch_000400.ckpt.index
- epoch_000400.ckpt.meta
- epoch_000499.ckpt.data-00000-of-00001
- epoch_000499.ckpt.index
- epoch_000499.ckpt.meta
- epoch_000600.ckpt.data-00000-of-00001
- epoch_000600.ckpt.index
- epoch_000600.ckpt.meta
- epoch_000800.ckpt.data-00000-of-00001
- epoch_000800.ckpt.index
- epoch_000800.ckpt.meta
- epoch_000999.ckpt.data-00000-of-00001
- epoch_000999.ckpt.index
- epoch_000999.ckpt.meta

预测

```
means = [123.68, 116.779, 103.939]
x = tf.placeholder(tf.float32, [None, 224, 224, 3])

sess = tf.Session()
vgg = model.vgg16(x)
fc8_finetuining = vgg.probs

saver = tf.train.Saver()
print("Model restoring...")
saver.restore(sess, './model/')#恢复最后保存的模型
#saver.restore(sess, './model/epoch_000800.ckpt')#或恢复指定检查点的模型
```

预测

```
filepath='./data/test/21.jpg' # 狗的图片
#filepath='./data/test/92.jpg' # 猫的图片
img = imread(filepath, mode='RGB')
img = imresize(img, (224, 224))
img = img.astype(np.float32)
for c in range(3):
    img[:,:,c] -= means[c]
prob = sess.run(fc8_finetuining, feed_dict={x: [img]})
max_index = np.argmax(prob)

if max_index == 0:
    print("This is a cat with possibility %.6f" % prob[:, 0])
else:
    print("This is a dog with possibility %.6f" % prob[:, 1])
```

```
Model restoring...
This is a dog with possibility 0.992637
```

预测结果输出

预测

```
#filepath='./data/test/21.jpg' # 狗的图片
filepath='./data/test/92.jpg' # 猫的图片
img = imread(filepath, mode='RGB')
img = imresize(img, (224, 224))
img = img.astype(np.float32)
for c in range(3):
    img[:,:,c] -= means[c]
prob = sess.run(fc8_finetuining, feed_dict={x: [img]})
max_index = np.argmax(prob)

if max_index == 0:
    print("This is a cat with possibility %.6f" % prob[:, 0])
else:
    print("This is a dog with possibility %.6f" % prob[:, 1])
```

```
Model restoring...
This is a cat with possibility 0.999549
```

预测结果输出

附录

脚本	函数
utils.py	get_file()
	get_batch()
	onehot()
VGG16_model.py **(注：右侧所列函数均在class vgg16中定义)**	__init__()
	saver()
	maxpool()
	conv()
	fc()
	convlayers()
	fc_layers()
	load_weights()
vgg_preprocess.py	（已在mooc平台上的第十二讲内容中通过富文本上传）