Flowers

相关论文

resnet50
inceptionv3
xception
inceptionres

数据集

数据集是在Kaggle上下载的数据集.

训练集的相关信息都存储在`json`文件里,数据集相当大,训练集有`70W`张图片.

```
def get_data():
    with open("test.json",'r') as f:
        test_dict = json.load(f)
        test_images = test_dict["images"]
    with open("train.json",'r') as f:
        train_dict = json.load(f)
        annotations = train_dict["annotations"]

    total_classes = 997 # test_dict['images']
    total_test = len(test_images)
    test_image_names = []

    for index in range(total_test):
        img_name = "test/"+test_images[index]["file_name"]
        test_image_names.append(img_name)
    test_image_names = np.array(test_image_names)
    # print("-----------------train----------------------")
    total_train = len(annotations)
    train_image_names = []
    train_image_labels = np.ones((total_train,))

    for index in range(total_train):
        img_name = "train/category_"+str(annotations[index]
['category_id']+1)+"/"+str(annotations[index]["image_id"])+".jpg"
        train_image_names.append(img_name)
        train_image_labels[index] = annotations[index]["category_id"]
    train_image_names = np.array(train_image_names)
    # release memory
    del annotations
    del test_images
    # train_test_split
    train_x,valid_x,train_y,valid_y =
```

```
    train_test_split(train_image_names,train_image_labels,test_size=0.2, random_state=42)

    return train_x,valid_x,train_y,valid_y,test_image_names
```

预处理

计算了一下数据集的均值和标准差,打算用于训练(最后没有尝试),使用`AWS P2.xlarge`计算一代需要5个小时,时间太长,就没有计算到最后结果.

最后使用迁移学习,使用了各个模型在`imagenet`上的训练权值.

搭建模型

我使用了`Resnet50`,`Xception`,`InceptionV3`,`InceptionRes`四个模型进行迁移学习.

微调各个模型的输出层,避免对卷积层造成太大的更新,等输出层收敛了,在微调一下卷积层.

最后结合各个模型来导出特征向量,结合特征向量在搭建一个分类层,训练至收敛.

Resnet50

添加了`dropout`需要训练更多代才能收敛.

```
# ResModel
Input = KL.Input(shape=(224,224,3))
# pre-processing
process_input = KL.Lambda(res_preprocess_input)(Input)
resnet_50 = resnet50.ResNet50(include_top=False,
                              weights='imagenet', input_tensor=process_input, pooling='avg')
# trainable
for layer in resnet_50.layers:
    layer.trainable=False


# output_shape: batch_size * 2048
output = resnet_50.output
# dropout
# output = KL.Dropout(0.5)(output)
# classfier
output = KL.Dense(CLASS_COUNT,activation='softmax')(output) #
,kernel_regularizer=regularizers.l2(0.001)
# model
res_model = KM.Model(resnet_50.input,output)

train_generator = MXGenerator((train_x,train_y),len(train_x),
                              des_size=(224,224),means=None,stds=None,
                              is_directory=True,batch_size=batch_size,shuffle=True,seed=0)

valid_generator = MXGenerator((valid_x,valid_y),len(valid_x),
                              des_size=(224,224),means=None,stds=None,
```

```
                                    is_directory=True,batch_size=batch_size,shuffle=True,seed=0)
adam = Adam(lr=0.001)
#
res_model.compile(optimizer=adam,loss='categorical_crossentropy',metrics=['accuracy'])

res_model.fit_generator(train_generator,len(train_x) // batch_size,
                        epochs=1, verbose=1,validation_data=valid_generator,
                        validation_steps=(len(valid_x) // batch_size))

# 放开所有层
for layer in resnet_50.layers:
    layer.trainable=True

res_model.compile(optimizer=Adam(lr=0.001),loss='categorical_crossentropy',metrics=
['accuracy'])

res_model.fit_generator(train_generator,len(train_x) // batch_size,
                        epochs=epochs - 5, verbose=1,validation_data=valid_generator,
                        validation_steps=(len(valid_x) // batch_size))

res_model.save_weights("res_model.h5")
```

其他几个模型也都是类似的处理,除了`inception_resnet_v2`需要调节一下输出层以外,因为它的`avg pooling`之后的输出为`(batch,1536)`,

```
conv_7b_bn (BatchNormalization) (None, 5, 5, 1536)   4608        conv_7b[0][0]
__________________________________________________________________________________________________
conv_7b_ac (Activation)         (None, 5, 5, 1536)   0           conv_7b_bn[0][0]
__________________________________________________________________________________________________
global_average_pooling2d_4 (Glo (None, 1536)         0           conv_7b_ac[0][0]
==================================================================================================
Total params: 54,336,736
Trainable params: 54,276,192
Non-trainable params: 60,544
```

,所以添加一层全连接层让输出为`(batch,2048)`与其他几个模型保持一致.

```
#output: (None, 1536)
output = inception_res.output
# dropout
output = KL.Dropout(0.5)(output)
# Dense
output = KL.Dense(2048,activation='relu')(output)
output = KL.Dropout(0.5)(output)
# classfier
output = KL.Dense(CLASS_COUNT,activation='softmax')(output)
# model
inc_res_model = KM.Model(inception_res.input,output,name='inc_res')
```

导出特征向量

通过几个微调后的模型分别进行特征向量的导出,其他模型也是相同的处理.

```
train_generator = MXGenerator((train_x,train_y),len(train_x),
                              des_size=(299,299),means=None,stds=None,
                              is_directory=True,batch_size=batch_size,shuffle=True,seed=0)

```

```
valid_generator = MXGenerator((valid_x,valid_y),len(valid_x),
                              des_size=(299,299),means=None,stds=None,
                              is_directory=True,batch_size=batch_size,shuffle=True,seed=0)
# test_generator
test_generator = MXGenerator((test_images,None),len(test_images),
                              des_size=(299,299),means=None,stds=None,
                              is_directory=True,batch_size=batch_size,shuffle=False,seed=0)
#

xcep_feature_model = KM.Model(xcep_model.input,xcep_model.layers[-2].output)

xcep_train_feature_vector =
xcep_feature_model.predict_generator(train_generator,steps=len(train_x) // batch_size)
xcep_valid_feature_vector =
xcep_feature_model.predict_generator(valid_generator,steps=len(valid_x) // batch_size)
xcep_test_feature_vector = xcep_feature_model.predict_generator(test_generator,
                                                                steps=len(test_images) //
batch_size)

# save to h5
xception_h5 = h5py.File("xception.h5",'w')
xception_h5.create_dataset('trian_x',
                           xcep_train_feature_vector.shape,data=xcep_train_feature_vector)
xception_h5.create_dataset('trian_label',
                           trian_y.shape,data=trian_y)
xception_h5.create_dataset('valid_x',
                           xcep_valid_feature_vector.shape,data=xcep_valid_feature_vector)
xception_h5.create_dataset('valid_label',
                           valid_y.shape,data=valid_y)
xception_h5.create_dataset('test',
                           xcep_test_feature_vector.shape,data=xcep_test_feature_vector)
```

结合特征向量以及构建新模型

分别导出了四个特征向量之后,就可以`concatenate`了,并训练与新模型.

```
X_train = []
y_trian = []
X_valid = []
y_valid = []
test = []

for file_name in ['xception.h5','inc_res.h5','res_.h5','inc_.h5']:
    with h5py.File(file_name,'r') as f:
        X_train.append(f['train_x'])
        X_valid.append(f['valid_x'])
        y_train.append(f['y_train'])
        y_valid.append(f['y_valid'])
        test.append(f['test'])

X_train = np.concatenate(X_train, axis=1)
X_valid = np.concatenate(X_valid,axis=1)
test = np.concatenate(test,axis=1)
y_train = np.concatenate(y_train)
```

```
y_valid = np.concatenate(y_valid)

# build the network
merge_input = KL.Input(shape=X_train.shape[1:])
merge_dropout = KL.Dropout(0.5)(merge_input)
merge_classfier = KL.Dense(CLASS_COUNT,activation='softmax')(merge_dropout)
merge_model(inputs=[merge_input],outputs=[merge_classfier])

merge_model.compile(optimizer=Adam(lr=0.001),loss='categorical_crossentropy',metrics=
['accuracy'])

datagen = ImageGenerator()

merge_train_generator = datagen.flow((X_train,y_train))
merge_valid_generator = datagen.flow((X_valid,y_valid))

merge_model.fit_generator(train_generator,len(train_x) // batch_size,
                    epochs=epochs, verbose=1,validation_data=valid_generator,
                    validation_steps=(len(valid_x) // batch_size)
```

导出预测结果

使用融合的模型进行预测最后结果,并生成`predict.csv`文件.

```
df = pd.read_csv("kaggle_sample_submission.csv")

images = np.zeros((224,224,3),dtype=np.int)

for i in range(len(test_images)):
    img = cv2.imread(test_images[i])
    img = cv2.cvtColor(img,cv2.BGRA2RGB)
    img = cv2.resize(img,(224,224))
    images[i] = img

# test_generator
test_generator = MXGenerator((test_images,None),len(test_images),
                             des_size=(299,299),means=None,stds=None,
                             is_directory=True,batch_size=batch_size,shuffle=False,seed=0)

y_pred = res_model.predict_generator(test_generator)

for i, fname in enumerate(test_generator.filenames):
    index = int(fname[fname.rfind('/')+1:fname.rfind('.')])
    df.set_value(index-1, 'predict', y_pred[i])

```