视频分类之 UCF-101 上的 CNN 方法详解

Code at Github: *https://github.com/sujiongming/UCF-101_video_classification*

视频分类包括人类行为识别和通用的多标签视频分类等研究内容。用的多标签视频分类以 2016 年谷歌发布的 youtube-8M 数据集为代表，其中很多视频属于多个类别，并且在类别上不限于人类行为。人类行为识别主要研究分类视频中包含的人类行动，一般一个视频中只包含一类人类行为，UCF101 数据集就是其中的典型代表。UCF-101（2012）包含 13,320 个视频（共 27 个小时），101 个人类行为类别，如运动、乐器和人物交互等。[1] 国内外研究人员在 UCF-101 数据集上进行了深入研究，目前最好的准确率已经达到 95%以上。

UCF-101 上的 CNN 方法一般作为其他分类方法的比较基准，也是最简单和自然的视频分类方法[2]。CNN 方法是将这些视频帧视为一张张静态图像，应用 CNN 识别每一帧，然后对预测结果进行平均处理来作为该视频的最终结果。然而，这个方法使用的是不完整的视频信息，因此使得分类器可能容易发生混乱而导致准确度不高。本文 CNN 方法在测试集上的最终准确度为 top1：63.0%，top5：87.5%。

UCF-101 中 6 类行为的样本图像帧

一、 基本过程和思想

基本思想是将数据集中视频及分类标签转换为图像（视频帧）和其对应的分类标签，再采用 CNN 网络对图像进行训练学习和测试，将视频分类问题转化为图形分类问题。具体步骤包括：

（1） 对每个视频(训练和测试视频)以一定的 FPS 截出视频帧（jpegs）保存为训练集和测试集，将对图像的分类性能作为所对应视频的分类性能：train set 有 1, 788, 425 帧图像，test set 有 697, 865 帧图像

（2） 选择一个预先训练好的 CNN 网络架构和初始权重，迁移学习至 UCF-101，如 inception v3 with pre-trained on ImageNet

（3） 用 train set 对 CNN 网络部分层进行重新训练，获得模型

（4） 训练完成后载入模型对 test set 内所有的视频帧进行检查验证，得出全测试集上的 top1 准确率和 top5 准确率输出

二、 运行环境简介

（1） 服务器硬件环境：40 核至强 cpu，GeForce GTX 1080 8G 显存 X2，128G 内存，512G SSD，3TB 机械硬盘

（2） 服务器软件环境：安装 ubuntu16.04、conda（含 python2.7），CUDA，cudnn、tensorflow GPU，keras 等所需 python 包，SSH 服务，screen 包，vim 工具，ffmpeg 包

（3） 客户机：window7 64 位，pycharm，xshell，xftp

（4） 使用模式：客户机远程 SSH 连接服务器进行操作

三、 运行过程和结果

（5） 准备数据（UCF 提供了三种训练/测试划分方案，本实例采用 1#划分方案）

i. 用 xftp 软件将开源项目程序和 UCF101 数据包上传至服务器谋和目录下

ii. 将 UCF101 数据包放在开源项目文件夹的 data 目录下，运行命令“unrar e UCF101.rar”

iii. 运行命令`python 1_move_files.py`

iv. 运行命令`python 2_extract_files.py`

（6） 训练 CNN，输出测试集性能

i. 运行 python CNN_train_UCF101.py，命令行输出训练过程信息，在测试子集上的准确率和 top5 准确率，系统函数默认 k=5。

ii. 训练集：到 41 epoch 时训练自动停止，最好的结果在 29 epoch 时出现，val_loss 最小为 1.19，采用该模型评价测试集。模型名称为：inception.0.29-1.19.hdf5，需要修改 CNN_evaluate_testset.py 文件加载该名称模型。

iii. 运行 python CNN_evaluate_testset.py，命令行输出测试集结果 。

iv. 测试集:loss：1.33，accuracy：0.63，top5 accuracy：0.875

v. 运行 python CNN_validate_images.py，命令行输出随机选择的 5 张图像的分类信息和结果。

需要注意的是，你的运行结果可能与本文有所差异，因此研究者可能需要不断修改调整模型超参，多次运行，找出最好的一组模型参数，取最好的准确度结果。

四、 将 inception-v3 网络迁移学习至 UCF-101

假设有数据集 D，不同于数据集 ImageNet，D 有 1024 个输入特征，200 个输出类别。

```
from keras.applications.inception_v3 import InceptionV3
from keras.preprocessing import image
from keras.models import Model
from keras.layers import Dense, GlobalAveragePooling2D
from keras import backend as K
# create the base pre-trained model
base_model = InceptionV3(weights='imagenet', include_top=False)
# include_top=False 因为我们想要重新在数据集 D 上训练 top level，我们移除了最后三层，暴露的是 mixed10 层的输出。
#模型最后 4 层的 layer.name, layer.input_shape, layer.output_shape
('mixed10', [(None, 8, 8, 320), (None, 8, 8, 768), (None, 8, 8, 768), (None, 8, 8, 192)
('avg_pool', (None, 8, 8, 2048), (None, 1, 1, 2048))
('flatten', (None, 1, 1, 2048), (None, 2048))
('predictions', (None, 2048), (None, 1000))
#因此，需要加三层
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)#加入 1024 个特征层
```

```
predictions = Dense(200, activation='softmax')(x)#输出层，加入数据集类别
model = Model(input=base_model.input, output=predictions)
for layer in base_model.layers:#固定 inceptionv3 模型参数
    layer.trainable = False
#编译模型
model.compile(optimizer='rmsprop',
loss='categorical_crossentropy')
#模型训练新加的层
model.fit_generator(
    train_generator,
    steps_per_epoch=100,
    validation_data=validation_generator,
    validation_steps=10,
    epochs=nb_epoch,
    callbacks=callbacks)
# 训练 the top 2 inception blocks，因此固定前 172 层，训练后面的层
for layer in model.layers[:172]: layer.trainable = False
for layer in model.layers[172:]: layer.trainable = True
model.compile(
    optimizer=SGD(lr=0.0001, momentum=0.9),
    loss='categorical_crossentropy',
    metrics=['accuracy', 'top_k_categorical_accuracy'])
model.fit_generator(
        train_generator,
        steps_per_epoch=100,
        validation_data=validation_generator,
        validation_steps=10,
        epochs=nb_epoch,
        callbacks=callbacks)
```

这样就在新的数据集上通过迁移学习训练好一个新的网络了。

需要注意的是

（1） InceptionV3 模型的输入为 299X299X3，因此数据集上的图片大小得修改为该大小格式：

```
train_generator = train_datagen.flow_from_directory(
    './data/train/',
    target_size=(299, 299),
    batch_size=32,
    classes=data.classes,
    class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
    './data/test/',
```

```
target_size=(299, 299),
batch_size=32,
classes=data.classes,
class_mode='categorical')

image_arr = process_image(image, (299, 299, 3)) #CNN_validate_images
```

（2） 验证测试数据集时

```
results = model.evaluate_generator(generator=test_generator,
steps=test_data_num // batch_size) #参数 steps 为数据生成的批次，一般为数据总数除以每批产生的数据个数
```

（3） 参数设置

```
#每个 epoch 后存入 val_loss 最小的模型
checkpointer = ModelCheckpoint(
filepath='./data/checkpoints/inception.{epoch:03d}-{val_loss:.2f}.hdf5',
verbose=1,
save_best_only=True)
# patience: number of epochs with no improvement after which training will be stopped. 10 个 epoch 模型性能没有改进后训练停止
early_stopper = EarlyStopping(patience=10)
```

其他代码和使用说明可以详见 github 项目 UCF-101_video_classification。

五、 附录

（7） Screen 工具：最大的好处在于客户端命令行窗口关闭后，服务器端程序可以继续运行如抽取特征、训练深度网络等耗时的任务

i. 一般发行版是不带这个软件的，需要自行安装，ubuntu 下面就直接 sudo apt-get install screen 或者下载安装 dpkg -i screen_4.3.1-2build1_amd64.deb

ii. 使用过程要点

1. screen -S name 启动一个名字为 name 的 screen
2. 输入命令，执行任务
3. ctrl + a + d（先按下 ctrl 和 a，再按下 d）保存断开当前的 screen，当时在当前 screen 运行的程序不会停止，回到前一个 screen
4. screen -ls 是列出所有的 screen
5. screen -r name 或者 id，就可以回到某个 screen，继续查看程序运行情况

（8） 训练时常用 ubuntu 命令

i. 监控 GPU 使用情况：nvidia-smi –l

ii. 监控 CPU 使用情况：top 按 1

iii. 内存使用情况：free –h

iv. 磁盘使用情况：df –h

v. 查看分区：fdisk -l

（9） Keras 简介

i. 常用 Model 属性

1. model.layers:组成模型的各个层
2. model.inputs：模型的输入张量列表
3. model.outputs：模型的输出张量列表

ii. 模型方法

1. model.compile(self, optimizer, loss, metrics=none,)
   a) optimizer 优化器
   b) loss 损失函数
   c) metrics:列表，包含评估模型在训练和测试时的性能指标
2. model.fit()：训练模型
3. model.fit_generator(self, generator, steps_per_epoch, validation_data, validation_steps,)
   a) generator:生成器函数，输出应该为如（inputs，targets）或者（inputs，targets，sample_weights）的 tuple
   b) steps_per_epoch:整数，当生成器返回 steps_per_epoch 次数据时一个 epoch 结束，执行下一个 epoch
   c) epoches：数据迭代的轮数
   d) validation_data:可以有三种形式，生成验证集的生成器，一个如（inputs，targets）或者（inputs，targets，sample_weights）的 tuple
   e) validation_steps:生成多少批验证数据，一般等于验证集数据数除以 batch_size
   f) workers:最大进程数
   g) max_q_size:生成器队列的最大容量
   h) initial_epoch:从该参数指定的 epoch 开始训练，在继续之前的训练时有用
4. model.evaluate()/model.evaluate_generator()：评估模型，计算 loss 值

5. model.predict()/model.predict_generator：预测分类结果

[1] Khurram Soomro, Zamir A R, Shah M. UCF101: A dataset of 101 human actions classes from videos in the wild[C]. CRCV-TR-12-01, 2012,

[2] Karpathy A, Toderici G, Shetty S*, et al.* Large-scale Video Classification with Convolutional Neural Networks[C]. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, 1725–1732.