Machine Learning Engineer Nanodegree

Capstone Proposal

顾飞 优达学城
2019年03月20日

Proposal

- AWS
- Dogs vs. Cats 项目背景
- 项目描述
- 数据集
- 解决方案
- 模型选择
- 结果评估
- 项目实施

AWS

由于此项目要求的计算量较大，也许你能够在本地 CPU 上训练。训练过程可能持续几个小时，取决于迭代次数、神经网络的大小和其他因素。

一种更快速的方法是在 GPU (图形处理器)上进行训练。虽然 CPU 可以比 GPU 计算更复杂的运算，但是吞吐量较小。因此在训练神经网络时，GPU 比 CPU 的速度要快很多。GPU 在同一时间会执行更多的计算，当我们处理大数据时这很重要。

你也可以自己购买 NVIDIA GPU，但是并非像将 GPU 插入机器那么简单。即使你知道如何安装 GPU，但是正确地安装 cuDNN 还是有一定的挑战。因此，我们不建议这么使用 GPU。

我们建议使用 Amazon EC2 实例，使用 Amazon p3.2xlarge 云服务器来完成该项目。

PS：如果你不想自己从头开始训练模型，重复造轮子。也可以使用预训练模型，提高训练准确度。

Domain Background

Dogs vs. Cats Redux: Kernels Edition

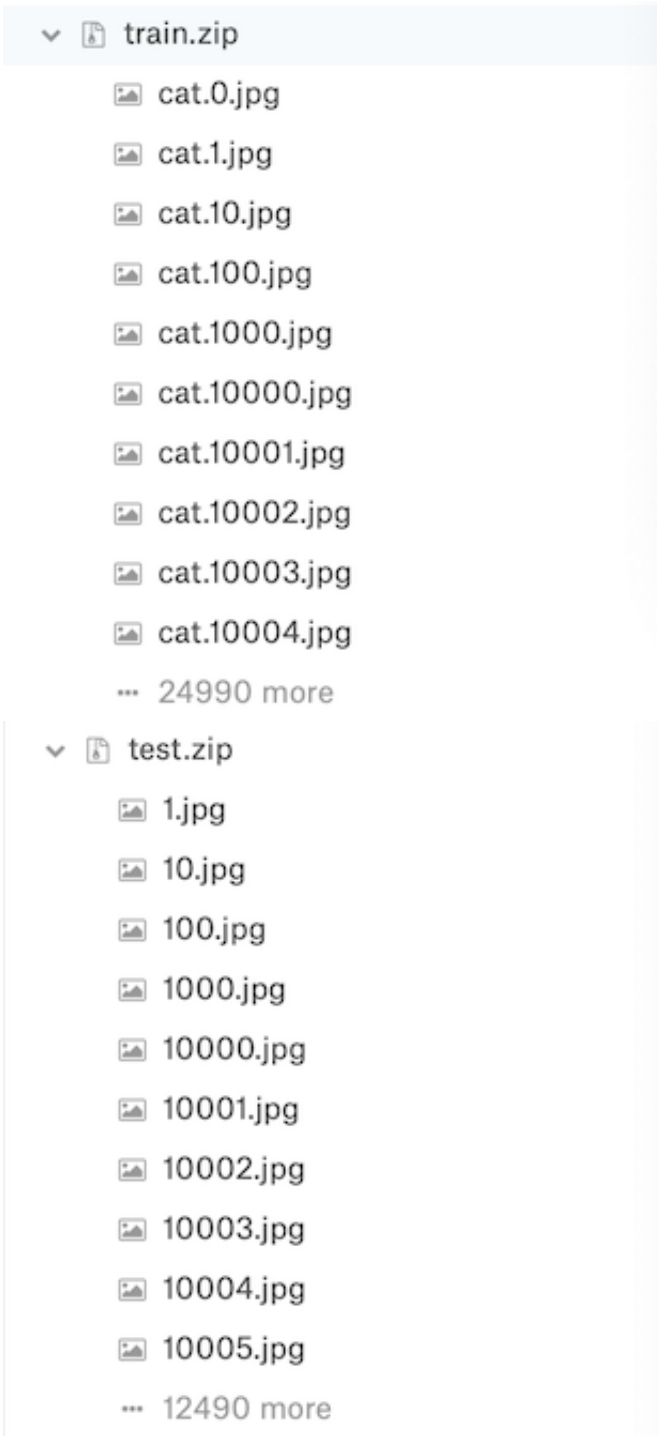
项目来源于 kaggle 在 2013 年组织的一场比赛：识别猫与狗。目前 Leaderboard 上展示了 1314 支队伍的成绩，排名第一的 score 是 0.03302，Top2% 的成绩是 0.04357。本项目的最低要求是 kaggle Public Leaderboard 前 10%，即 0.06149。

Problem Statement

本文会通过 Keras 搭建一个深度卷积神经网络（Convolutional Neural Network, CNN）来识别一张图片是猫还是狗，融合三种模型（ResNet50、InceptionV3 和 Xception）。本项目需要对测试样本进行分类，然后基于 CNN 的模型进行训练，并将结果上传至 kaggle 进行评分。

Datasets and Inputs

此数据集可以从 kaggle 上下载。Dogs vs. Cats Redux: Kernels Edition



下载 kaggle 猫狗数据集解压后分为 3 个文件 train.zip、 test.zip 和 sample_submission.csv。

train 训练集包含了 25000 张猫狗的图片，猫狗各一半，每张图片包含图片本身和图片名。命名规则根据 “type.num.jpg” 方式命名。

test 测试集包含了 12500 张猫狗的图片，没有标定是猫还是狗，每张图片命名规则根据 “num.jpg”，需要注意的是测试集编号从 1 开始，而训练集的编号从 0 开始。

sample_submission.csv 需要将最终测试集的测试结果写入.csv 文件中，上传至 kaggle 进行打分。

Solution Statement

为了提高模型的表现，本项目决定使用预训练网络，最终选择了ResNet50, Xception, Inception V3 这三个模型，每个模型导出的时间都挺长。 这三个模型都是在 ImageNet 上面预训练过的，所以每一个模型都可以说是身经百战。我们可以将多个不同的网络输出的特征向量先保存下来，后续即使是在普通笔记本上也能轻松训练。

有了三个特征向量后，我们需要将这三条向量进行合并成一条特征向量。

Benchmark Model

模型构建时，设置 dropout = 0.5，激活函数：Sigmoid，优化器为 Adadelta。

Evaluation Metrics

对数损失（Log loss）亦被称为逻辑回归损失（Logistic regression loss）或交叉熵损失（Cross-entropy loss）。 交叉熵是常用的评价方式之一，它实际上刻画的是两个概率分布之间的距离，是分类问题中使用广泛的一种损失函数。 本文实际上是二分类问题， 因此可以采用 logloss 损失函数作为评价指标， 计算公式如下：

$$LogLoss = -\frac{1}{n}\sum_{i=1}^{n}[y_i log(\hat{y}_i) + (1-y_i) log(1-\hat{y}_i)]$$

其中：

- n 是测试集中图片数量
- $\hat{y}$ 是图片预测为狗的概率
- y_i 如果图像是狗，则为1，如果是猫，则为0
- $\log()$ 是自然（ 基数^e）对数

采用交叉熵作为损失函数可以有效的解决梯度消失和梯度爆炸的问题。 交叉熵损失越小，代表模型的性能越好。上述评估指标可用于评估该项目的解决方案以及基准模型。

Project Design

数据处理：

```
train_list = os.listdir('train')

train_cat = filter(lambda x:x[:3] == 'cat', train_list)
train_dog = filter(lambda x:x[:3] == 'dog', train_list)

def mkdir(path):
    folder = os.path.exists(path)
    if folder:
        shutil.rmtree(path)
    os.mkdir(path)

train_symlink_path = 'train-symlink'
test_symlink_path = 'test-symlink'
```

```
mkdir(train_symlink_path)

os.mkdir(train_symlink_path + '/cat')
for filename in train_cat:
    os.symlink('../../train/' + filename, train_symlink_path + '/cat/' +
filename)

os.mkdir(train_symlink_path + '/dog')
for filename in train_dog:
    os.symlink('../../train/' + filename, train_symlink_path + '/dog/' +
filename)

mkdir(test_symlink_path)
os.symlink('../test/', test_symlink_path + '/test')
```

导出特征向量：

```
def write_gap(MODEL, image_size, lambda_func=None):
    input_tensor = Input(shape=(image_size[0], image_size[1], 3))
    x = input_tensor
    if lambda_func:
        x = Lambda(lambda_func)(x)

    base_model = MODEL( include_top=False, weights='imagenet', input_tensor=x)
    model = Model(base_model.input, GlobalAveragePooling2D()
(base_model.output))

    gen = ImageDataGenerator()
    train_generator = gen.flow_from_directory("train-symlink", image_size,
shuffle=False,
                                              batch_size=32)
    test_generator = gen.flow_from_directory("test-symlink", image_size,
shuffle=False,
                                             batch_size=32, class_mode=None)

    train = model.predict_generator(train_generator,
math.ceil(train_generator.samples*1.0/train_generator.batch_size), verbose=1)
    test = model.predict_generator(test_generator,
math.ceil(test_generator.samples*1.0/test_generator.batch_size), verbose=1)

    if MODEL==ResNet50:
        model_name = "gap_ResNet50.h5"
    elif MODEL==Xception:
        model_name = "gap_Xception.h5"
    elif MODEL==InceptionV3:
        model_name = "gap_InceptionV3n.h5"
    with h5py.File(model_name) as h:
        h.create_dataset("train", data=train)
```

```
        h.create_dataset("test", data=test)
        h.create_dataset("label", data=train_generator.classes)

write_gap(ResNet50, (224, 224))
write_gap(Xception, (299, 299), xception.preprocess_input)
write_gap(InceptionV3, (299, 299), inception_v3.preprocess_input)
```

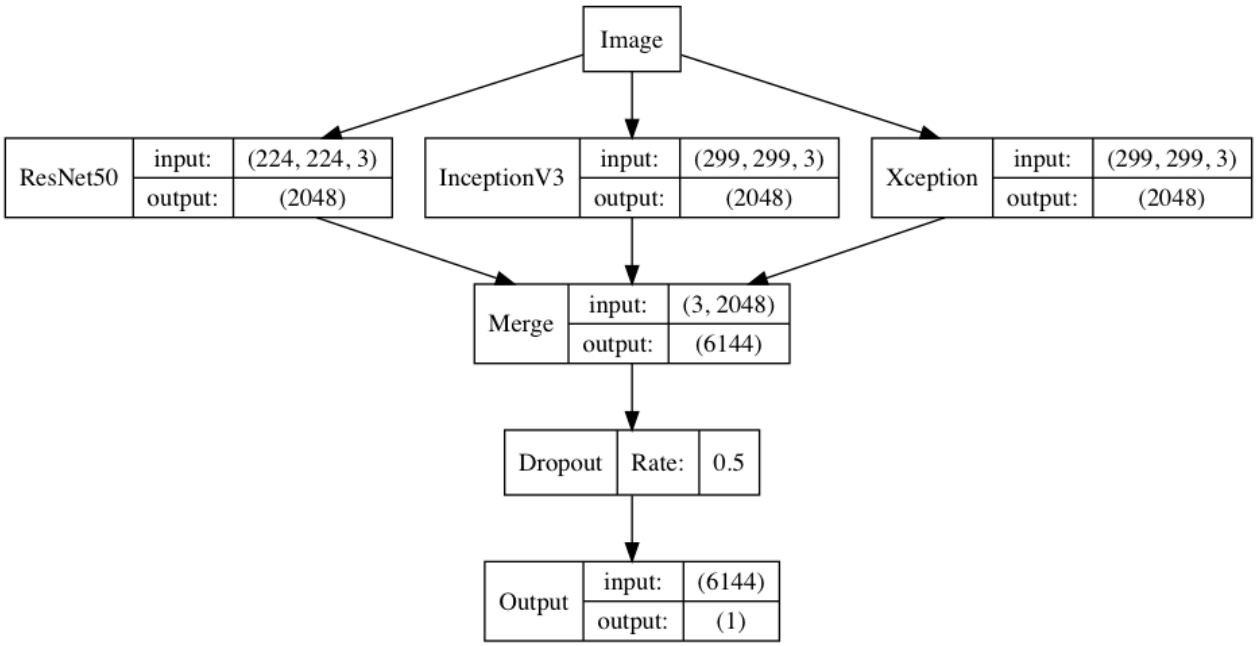
合并特征向量：

```
np.random.seed(2019)

X_train = []
X_test = []

for filename in ["gap_ResNet50.h5", "gap_Xception.h5", "gap_InceptionV3.h5"]:
    with h5py.File(filename, 'r') as h:
        X_train.append(np.array(h['train']))
        X_test.append(np.array(h['test']))
        y_train = np.array(h['label'])

X_train = np.concatenate(X_train, axis=1)
X_test = np.concatenate(X_test, axis=1)

X_train, y_train = shuffle(X_train, y_train)
```

模型构建：

```
input_tensor = Input(X_train.shape[1:])
x = Dropout(0.5)(input_tensor)
x = Dense(1, activation='sigmoid')(x)
model = Model(input_tensor, x)

model.compile(optimizer='adadelta',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

迁移学习的神经网络结果如下图：



模型训练：

```
fit = model.fit(X_train, y_train, batch_size=128, epochs=8,
validation_split=0.2, verbose=1)
print(fit.history)
```

预测测试集：

```
y_pred = model.predict(X_test, verbose=2)
y_pred = y_pred.clip(min=0.005, max=0.995)
```

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

gen = ImageDataGenerator()
test_generator = gen.flow_from_directory("test-symlink", (224, 224),
shuffle=False,
                                         batch_size=32, class_mode=None)

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

df.to_csv('pred.csv', index=None)

df.head(10)
```