智能计算系统实验

实验指导书

安徽工业大学计算机科学与技术学院

实验　三	基于 TensorFlow 实现风格迁移
实验　四	基于 YOLOv3 实现目标检测
专业班级	智能 202
学　　号	209074458
姓　　名	张建才
指导教师	李雪

摘要:

1: 内含有主要代码。

2: 各个函数的数学表达形式及其实际意义。

3: TensorFlow 模型的使用。

4: TensorFlow 的函数说明。

5: 模型使用的方法。

实验 3: 基于 TensorFlow 实现风格迁移

实验目的

1) 神经网络的模块化实现;

2) 使用 Tensor Flow 实现风格迁移;

3) 使用内容损失函数和风格损失函数。

背景知识

1) TensorFlow 函数: tf.keras.applications.VGG19 部分参数说明。

include_top: whether to include the 3 fully-connected layers at the top of the network.;
weights: one of None (random initialization), 'ImageNet' (pre-training on ImageNet), or the path to the weights file to be loaded.;

2) 在 keras 中,要想获取层的输出的各种信息,可以先获取层对象,再通过层对象的属性 output 或者 output_shape 获取层输出的其他特性。`def get_layer(self, name=None, index=None):`返回值:层实例(因为每个层都是一个类,所以返回的层本质上是一个类)。

3) 内容损失函数:

$$L_i = \frac{1}{2 * M * N} \sum_{ij} (X_{ij} - P_{ij})^2$$

X 是噪声图片的特征矩阵，P 是内容图片的特征矩阵。M 是 P 的长*宽，N 是信道数

4) 风格损失函数：

$$L_i = \frac{1}{4 * M^2 * N^2} \sum_{ij} (G_{ij} - A_{ij})^2$$

M 是特征矩阵的长*宽，N 是特征矩阵的信道数。G 为噪音图像特征的 Gram 矩阵，A 为风格图片特征的 GRAM 矩阵。

5) 最终用于训练的损失函数为内容损失和风格损失的加权和：

$$L_{total} = \alpha L_{content} + \beta L_{style}$$

6) 当训练开始时，我们根据内容图片和噪声，生成一张噪声图片。并将噪声图片喂给网络，计算 loss，再根据 loss 调整噪声图片。将调整后的图片喂给网络，重新计算 loss，再调整，再计算…直到达到指定迭代次数，此时，噪声图片已兼具内容图片的内容和风格图片的风格，进行保存即可。

实验环境

使用运算硬件：CPU

软件环境：TensorFlow==2.3.1；python==3.7；

使用数据集：imagenet 数据集

实验步骤

1) 我们使用 VGG 中的一些层的输出来表示图片的内容特征和风格特征；分离内容特征层和风格特征层的输出，方便后续计算。

2) 将内容图片输入网络。

3) 计算内容损失。

4) 将风格图片输入网络。

5) 计算风格损失。

6) 最终用于训练的损失函数为内容损失和风格损失的加权和。

7) 提取特征层：

```
# 内容特征层字典 Dict[层名,加权系数]
self.content_layers = content_layers
# 风格特征层
self.style_layers = style_layers
# 提取需要用到的所有vgg层
layers = list(self.content_layers.keys()) + list(self.style_layers.keys())
# 创建layer_name到output索引的映射
self.outputs_index_map = dict(zip(layers, range(len(layers))))
# 创建并初始化vgg网络
self.vgg = get_vgg19_model(layers)
```

8) 迭代函数：

```
# 求loss
with tf.GradientTape() as tape:
    noise_outputs = model(noise_image)
    loss = total_loss(noise_outputs)
# 求梯度
grad = tape.gradient(loss, noise_image)
# 梯度下降，更新噪声图片
optimizer.apply_gradients([(grad, noise_image)])
return loss
```

实验思考

1) 使用 Adma 优化器。

2) TensorFlow 内含有比较多的函数和参数。在直接使用时，像使用 VGG 时，我们可以将 VGG 模型的参数固定，因为我们不需要训练，使用就行。

实验二：基于 YOLOv3 实现目标检测

实验目的

1） 使用 YOLOv3。

2） 理解目标检测的方法。

3） 检测加上分类。

背景介绍

1） YOLOv3:包括划分单元格来做检测、使用 Leaky ReLU[27] 作为激活 函数、使用 Batch Normalization[16] 避免过拟合、使用多尺度训练等，通过修改主干网络来提升精度与性能。

2） YOLOv3 做目标检测时，首先用卷积神经网络提取特征，然后用非极大值抑制（NMS） 来筛选候选框。

3） 借鉴 Resnet 思想，引入残差（Rsidual）结构。

实验环境

硬件环境：CPU

软件环境：python==3.5.2；TensorFlow==2.2.0； N

实验步骤

1） Download YOLOv3 weights from YOLO website.
2） Convert the Darknet YOLO model to a Keras model.
3） Run YOLO detection.

4） 然后通过 convert.py 脚本构建模型，并将权重转成 Keras 版

本。

5） 对于每一种 class，分别进行非极大值抑制：

```
for c in range(num_classes):
    class_boxes = tf.boolean_mask(boxes, mask[:, c])
    class_box_scores = tf.boolean_mask(box_scores[:, c], mask[:, c])
    nms_index = tf.image.non_max_suppression(
        class_boxes, class_box_scores, max_boxes_tensor, iou_threshold=iou_threshold)
    class_boxes = K.gather(class_boxes, nms_index)
    class_box_scores = K.gather(class_box_scores, nms_index)
    classes = K.ones_like(class_box_scores, 'int32') * c
    boxes_.append(class_boxes)
    scores_.append(class_box_scores)
    classes_.append(classes)
boxes_ = K.concatenate(boxes_, axis=0)
scores_ = K.concatenate(scores_, axis=0)
classes_ = K.concatenate(classes_, axis=0)
```

实验思考

1） YOLO 系列算法可以算是颇负盛名。目标检测相关的算法有不少，而 YOLO 因其识别速度快而出名，常被用于实时目标检测场景中。

2） 使用默认定位点。如果使用自己的定位点，则可能需要进行一些更改。

3） 推理结果与 Darknet 不完全相同，但差异很小。