媒体计算方法课程

实验总结报告

实验名称: 媒体信号变换

学　　号: 19053026

姓　　名: 张壹帆

日　　期: 2020/01/06

一、实验目的

1 实验目标

根据指导书中的实验介绍，本实验以图像分类为例，将实验目标概述为以下三方面:

(1) 对图像的颜色直方图等全局特征进行提取;

(2) 基于 SIFT 和词袋模型对图像的局部特征进行提取，同时用基于 VGG 的网络进行深度特征的提取，将两种方式提取到的特征进行对比分析;

(3) 设计并实现基于 SIFT 特征提取的图像分类算法，和基于 VGG 网络进行分类的效果进行对比分析。

2 实验涉及到的学习内容

本实验主要是对传统方式和基于深度神经网络的模型对图像特征的表征能力进行对比，以图像分类为例，涉及到的学习内容主要有如下几点:

(1) 了解图像的全局特征（如颜色直方图）和提取方法;

(2) 了解传统方式对图像的表征能力：学习基于 SIFT 方法对局部不变特征的提取原理，学习实现此算法的具体实现方法;

(3) 了解基于深度神经网络的模型对图像的表征能力：学习深度学习相关的基础知识，掌握基于 VGG16 网络模型对图像深度特征的提取基本流程;

(4) 了解基于 SIFT 特征提取的图像分类基本流程，学习特征编码方式和 SVM 分类器的相关知识;

(5) 学习相关的深度学习框架来复现基于VGG的图像分类算法，学习特征的可视化相关的知识以更好进行对比分析。

二、实验具体完成情况

(1) 对实验目的的理解

本实验目的是以图像分类为例了解图像特征的表征能力，分别为实现对图像全局特征（颜色直方图）的提取，实现基于 SIFT 算法对图像局部不变特征的提取，实现利用 SVM 分类器对基于 SIFT 算法所提取的特征进行分类，实现基于 VGG16 对图像深度特征的提取和分类；最后，将基于 SVM 和 SIFT 算法的图像分类器效果和基于深度网络 VGG16 的图像分类器下效果进行对比分析，得出相关结论。

(2) 实验方案的设计

根据对实验的理解，将实验方案设计如表 1 所示。

表 1 实验方案设计

实现内容	方法 1	方法 2
全局特征提取	Opencv 提取颜色直方图	---
局部特征提取	基于 SIFT 算法	基于 VGG16 网络
图像分类	基于 SIFT 特征提取、特征聚类、SVM 分类器	基于 VGG16 网络模型

实验环境设置列出如下。

- 主要硬件环境：CPU 为 Intel(R) Core(TM) i7-6700 CPU @ 3.40GH，GPU 为 NVIDIA GeForce GTX 1080;
- 主要软件环境：操作系统为 Ubuntu18.04；IDE 为 VSCode；编

程语言为 Python; 深度学习计算框架为 Tensorflow2.0; 其他主要用到的软件库为 python-opencv、matplotlib、numpy 等。

- 数据集: 考虑到硬件环境和训练时间问题，采用 Caltech256 数据集的前 43 类作为本次实验的数据集。

2 具体技术途径

2.1 全局特征提取（颜色直方图）

(1) 基本原理

颜色特征是一种全局特征，描述了图像或图像区域所对应的景物的表面性质。一般颜色特征是基于像素点的特征，此时所有图像或图像区域的像素都有各自的贡献。由于颜色对图像或图像区域的方向、大小等变化不敏感，所以颜色特征不能很好地捕捉图像中对象的局部特征。另外，仅使用颜色特征查询时，如果数据库很大，常会将许多不需要的图像也检索出来。颜色直方图是最常用的表达颜色特征的方法，其优点是不受图像旋转和平移变化的影响，进一步借助归一化还可以不受图像尺度变化的影响，其缺点是没有表达出颜色空间分布的信息。

(2) 实现方法

Step1: 导入 opencv，利用 imread 函数读入原始图像;

Step2: 利用 split 函数划分图像颜色 RGB 通道;

Step3: 利用 calcHist 函数计算直方图;

Step4: 对每个通道分别统计并在直方图中画出;

Step5: 展示结果，将结果直方图保存。

2.2局部特征提取

2.2.1 基于 SIFT 的局部特征提取

(1)基本原理

SIFT 即尺度不变特征转换，SIFT 特征是基于物体上的一些局部外观的兴趣点，与图像的大小和旋转无关。SIFT 算法是用来提取图像的局部特征的算法，它的实质是在不同的尺度空间上查找关键点，并计算出关键点的方向。(注：查找到的关键点是一些十分突出，不会因光照，仿射变换和噪音等因素而变化的点，如角点、边缘点、暗区的亮点及亮区的暗点等。)

SIFT 算法的主要流程为：初始化操作，构建图像的金字塔；关键点搜索、定位、筛选；关键点方向赋值；关键点描述子的生成。

初始化操作，构建图像的金字塔：高斯金字塔构建过程中，一般首先将图像扩大一倍，在扩大的图像的基础之上构建高斯金字塔，然后对该尺寸下图像进行高斯模糊，几幅模糊之后的图像集合构成了一个八度，然后对该八度下的最模糊的一幅图像进行下采样的过程，长和宽分别缩短一倍，图像面积变为原来四分之一。

关键点搜索、定位、筛选：实质是为了寻找尺度空间的极值点，因此每一个采样点要和它所有的相邻点比较，看其是否比它的图像域和尺度域的相邻点大或者小。在离散空间下寻找的点不一定是符合的，所以需要定位，可以通过对尺度空间 DoG 函数进行曲线拟合寻找极值点来减小这种误差。在 DOG 函数中，它存在比较强的边缘效应，而当特征点在边缘的时候，这些点就会很不稳定，所以，我们应该把

这些边缘效应很强的点找出来，进行筛选。

关键点方向赋值：为了实现图像的旋转不变性，需要根据检测到的关键点的局部图像结构为特征点方向赋值。

关键点描述子的生成：这个描述子不但包括关键点，也包括关键点周围对其有贡献的像素点。

(2)实现方法：

Step1： 导入 opencv，利用 imread 函数读入训练集中的图像；

Step2： 利用 cutColor 函数将图像转换成灰度图：

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Step3： 利用 xfeatures2d 中的 SIFT 算法提取图像的局部特征：

```
sift = cv2.xfeatures2d.SIFT_create(200)
```

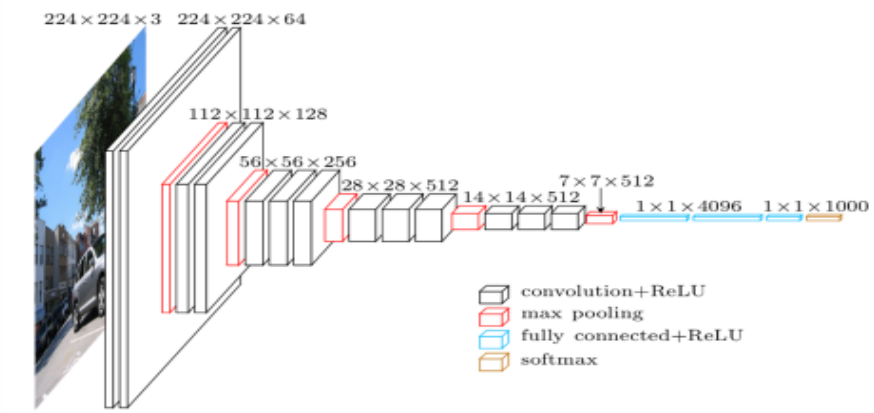
Step4： 对提取到的局部特征进行描述：

```
kp, des = sift.detectAndCompute(gray, None)
```

2.2.2 基于 VGG16 的特征提取

(1)基本原理

VGG-16 网络没有那么多超参数，是一种只需要专注于构建卷积层的简单网络。



首先用 3 × 3，步幅为 1 的过滤器构建卷积层，padding 参数为 same 卷积中的参数。然后用一个 2 × 2，步幅为 2 的过滤器构建最大池化层。进行第一个卷积之后得到 224×224×64 的特征图，接着还有一层 224×224×64，得到这样 2 个厚度为 64 的卷积层，意味着我们用 64 个过滤器进行了两次卷积。接下来创建一个池化层，池化层将输入图像压缩到 112×112×64。经过若干个卷积层，使用 129 个过滤器，再进行池化，可以推导出池化后的结果是这样（56×56×128）。接着再用 256 个相同的过滤器进行三次卷积操作，然后再池化，然后再卷积三次，再池化。如此进行几轮操作后，将最后得到的 7×7×512 的特征图进行全连接操作，得到 4096 个单元，然后进行 softmax 激活，输出从 1000 个对象中识别的结果。本实验中修改输出层，为 1024 个单元，所分类别为 43，VGG 中间层的结构如下图所示（VGG16 为 D 所对应）。

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					

(2) 实现方法

Step1: 导入训练集和测试机;

Step2: 利用 keras 接口导入 VGG16 网络模型:

```
from keras.applications.vgg16 import VGG16
```

Step3: 更改网络的输出层，在输出所提取的特征后添加一个全连接层层，并且利用 softmax 激活函数对 43 种类别进行分类，构建完整的适用于本实验图像分类的模型:

```
if base_model_name=="VGG16":
    base_model=VGG16(weights=weights,include_top=False)
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(43, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=predictions)
```

Step4: 构建模型，加载 ImageNet 上的预训练权重:

```
model=build_model("VGG16","imagenet")
```

Step5: 对模型进行训练，保存训练相关的信息和参数;

Step6: 对训练好的网络模型进行测试，同时编写可视化特征方法将 VGG16 网络输出的特征进行可视化。

2.3 图像分类器

2.3.1 基于 SIFT 特征提取、特征聚类、SVM 分类器

(1) 基本原理

基于 SIFT 特征提取的基本原理已经在 2.2.1 中进行说明。

特征聚类采用的是 K-Means 方法，K-means 聚类算法是一种迭代求解的聚类分析算法，其步骤是随机选取 K 个对象作为初始的聚类中心，然后计算每个对象与各个种子聚类中心之间的距离，把每个对象分配给距离它最近的聚类中心。聚类中心以及分配给它们的对象就代表一个聚类。每分配一个样本，聚类的聚类中心会根据聚类中现有的对象被重新计算。这个过程将不断重复直到满足某个终止条件。终止条件可以是没有（或最小数目）对象被重新分配给不同的聚类，没有（或最小数目）聚类中心再发生变化，误差平方和局部最小。

支持向量机（SVM）的基本模型是定义在特征空间上间隔最大的线性分类器。是一种二分类模型，当采用了核技巧后，支持向量机就可以用于非线性分类。

(2) 实现方法

Step1: 首先对 2.2.1 中 SIFT 算法提取到的特征进行 K-Means 聚类:

```
compactness, labels, centers = cv2.kmeans(features, wordCnt,None)
```

Step2: 读取训练集的图片，并且进行 SIFT 特征提取;

Step3: 创建 SVM 分类器，进行参数训练:

```
        trainData = np.float32(trainData)
        response = response.reshape(-1, 1)
        svm = cv2.ml.SVM_create()
        svm.setKernel(cv2.ml.SVM_LINEAR)
    svm.train(trainData,cv2.ml.ROW_SAMPLE,response)
```

Step4: 保存训练的参数:

svm.save("svm.clf")

2.3.2 基于 VGG16 网络构建图现象分类器

基本原理和实现方法已经在 2.2.2 中说明。其中输出层没有使用两个 FC-4096 全连接层和 FC-1000 分类，而是采用全局池化层替代全连接层，主要有两个优点:

一是 GAP 在特征图与最终的分类间转换更加简单自然;

二是不像 FC 层需要大量训练调优的参数，降低了空间参数会使模型更加健壮，抗过拟合效果更佳。

3、实验结果与分析

3.1 全局特征提取（颜色直方图）

使用 calcHist 函数对图像的颜色直方图进行提取，如图 1 所示。

图 1 测试图片原图

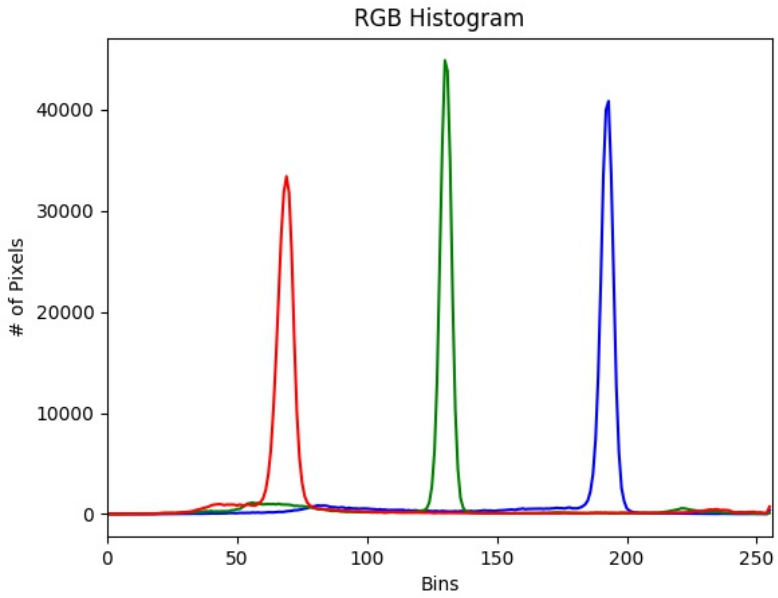


图 2 测试图片对应的颜色直方图

3.2 局部特征值提取

(1) 基于 SIFT 算法特征提取

SIFT 提取到的特征如图 3 所示。

图 3 绘制 SIFT 在图像上提取到的关键点

分析：Python-Opencv 中的 SIFT 对象会使用 DoG 检测关键点，

对关键点周围区域计算向量特征，检测并返回关键点和描述符，描述符是检测到的特征的局部区域图像列表。根据返回的关键点，对每个关键点画出圆圈和方向，即图 3 中的蓝色圆圈所标注。

(2) 基于 VGG16 算法的特征提取

选择测试图片，输入已经训练好的网络模型，那基于 VGG16 基本架构的分类网络模型过程中提取到的深度特征可视化如图 4 所示。

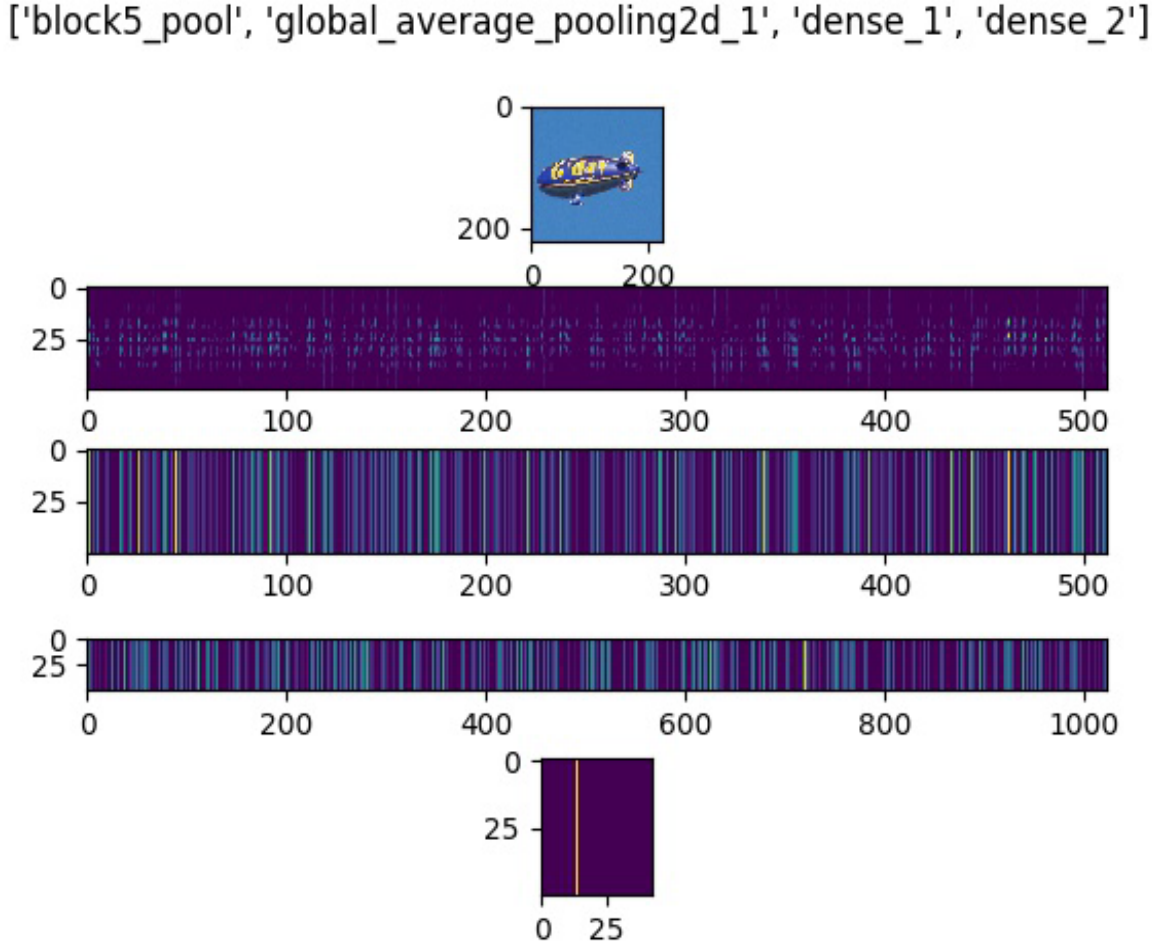


图 4 绘制 SIFT 在图像上提取到的关键点

分析：对 block5_pool 输出的特征，由于是三维的，所以进行特征拼接输出可视化特征图；其后面可视化所接的全局池化层 global_average_poloing2d 所提取的特征，因为是一维的，所以在列上

重复 50 次显示输出，同样全连接层 dense_1 和 dense_2 中可视化输出特征，可以看出 dense_2 的输出已经显示出 softmax 的分类效果。

3.3 图像分类器

数据集说明：对 43 类样本划分成训练集、验证集和测试集，其中在每类图像下随机抽取一张图片作为测试已经训练好的模型，主要用于观察对图像特征的提取过程；其余将训练集和验证集的比例设置为 8：2。基于 SIFT 算法、K-Means 聚类和 SVM 分类器的图像分类器和基于 VGG16 网络模型的图像分类器采用相同的训练集进行训练，相同的验证集进行准确率的评估。

(1) 基于 SIFT 算法、K-Means 聚类和 SVM 分类器

训练过程如图 5 所示，在验证集上的效果如图 6 所示。

```
(tf2.0) oem@cryan:/media/oem/HopeField/zyf_workspace/Media_Compute$ python sift_feature.py
Init training data of 019.boxing-glove...
Done

Init training data of 001.ak47...
Done

Init training data of 002.american-flag...
Done

Init training data of 003.backpack...
Done

Init training data of 004.baseball-bat...
Done
```

图 5 基于 SVM 分类器算法的训练过程

```
Classify on TestSet 039.chopsticks:
Accuracy: 13 / 21

Classify on TestSet 040.cockroach:
Accuracy: 14 / 29

Classify on TestSet 041.coffee-mug:
Accuracy: 8 / 22

Classify on TestSet 042.coffin:
Accuracy: 9 / 18

Classify on TestSet 043.coin:
Accuracy: 31 / 49

Total accuracy: 518 / 1047---0.49475
(tf2.0) oem@cryan:/media/oem/HopeField/zyf_workspace/Media_Compute$
```

图 6 SVM 分类器在验证集上的表现

最终在验证集上的准确率为 49.48%

(2) 基 VGG16 网络模型构建的分类器

基于 VGG16 网络架构实现的具体分类模型的结构和模型参数说明如图 7 所示。

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, None, None, 3)     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, None, None, 64)    1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, None, None, 64)    36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, None, None, 64)    0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, None, None, 128)   73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, None, None, 128)   147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, None, None, 128)   0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, None, None, 256)   295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, None, None, 256)   590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, None, None, 256)   590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, None, None, 256)   0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, None, None, 512)   1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, None, None, 512)   0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, None, None, 512)   0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              525312
_________________________________________________________________
dense_2 (Dense)              (None, 43)                44075
=================================================================
Total params: 15,284,075
Trainable params: 15,284,075
Non-trainable params: 0
```

图 7 基于 VGG 架构的图片分类网络模型

网络训练过程如图 8-10 所示，在验证集上的效果如图 10 所示。

```
Epoch 27/40
124/124 [==============================] - 46s 369ms/step - loss: 0.0062 - acc: 0.9985 - val_loss: 2.2674 - val_acc: 0.8329
Epoch 28/40
124/124 [==============================] - 45s 361ms/step - loss: 0.0077 - acc: 0.9977 - val_loss: 0.6952 - val_acc: 0.8262
Epoch 29/40
124/124 [==============================] - 45s 366ms/step - loss: 0.0077 - acc: 0.9985 - val_loss: 0.9170 - val_acc: 0.8138
Epoch 30/40
124/124 [==============================] - 45s 361ms/step - loss: 0.0110 - acc: 0.9975 - val_loss: 1.3356 - val_acc: 0.8252
Epoch 31/40
124/124 [==============================] - 45s 361ms/step - loss: 0.0038 - acc: 0.9990 - val_loss: 1.0542 - val_acc: 0.8319
Epoch 32/40
124/124 [==============================] - 45s 361ms/step - loss: 0.0079 - acc: 0.9980 - val_loss: 3.1941 - val_acc: 0.8262
Epoch 33/40
124/124 [==============================] - 45s 367ms/step - loss: 0.0028 - acc: 1.0000 - val_loss: 0.9497 - val_acc: 0.8300
Epoch 34/40
124/124 [==============================] - 45s 365ms/step - loss: 0.0050 - acc: 0.9982 - val_loss: 1.4224 - val_acc: 0.8271
Epoch 35/40
124/124 [==============================] - 45s 362ms/step - loss: 0.0056 - acc: 0.9980 - val_loss: 1.1216 - val_acc: 0.8309
Epoch 36/40
124/124 [==============================] - 45s 367ms/step - loss: 0.0045 - acc: 0.9987 - val_loss: 0.7644 - val_acc: 0.8281
Epoch 37/40
124/124 [==============================] - 45s 365ms/step - loss: 0.0025 - acc: 0.9995 - val_loss: 2.0575 - val_acc: 0.8319
Epoch 38/40
124/124 [==============================] - 45s 360ms/step - loss: 0.0036 - acc: 0.9990 - val_loss: 1.1145 - val_acc: 0.8319
Epoch 39/40
124/124 [==============================] - 45s 365ms/step - loss: 0.0020 - acc: 1.0000 - val_loss: 0.7757 - val_acc: 0.8290
Epoch 40/40
124/124 [==============================] - 45s 366ms/step - loss: 0.0025 - acc: 0.9997 - val_loss: 0.7262 - val_acc: 0.8300
```

图 8 网络参数的训练过程

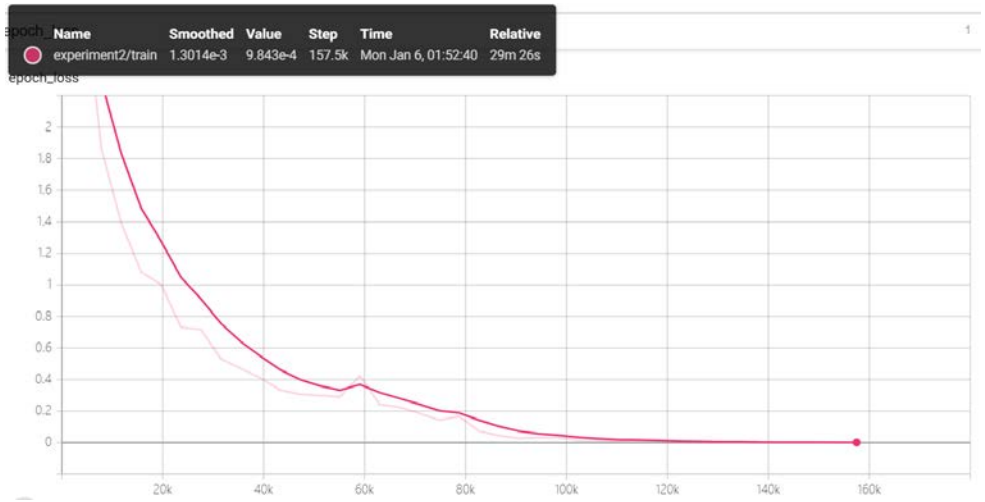


图 9 可视化测试集上损失率下降曲线

图 10 可视化训练集和验证集上准确率表现

最终在验证集上的准确率为 83.0%

分析: 对于相同的实验环境和数据集，基于深度神经网络 VGG16 的分类器表现更加优异，在验证集上的准确率比传统的基于 SIFT 特征提取、K-means 特征聚类和 SVM 分类器构建的图像分类器大概提高了 40.39%。因此，相较于传统的特征提取方法，深度神经网络能够捕捉到更为丰富精细的深度特征信息。

三、实验心得与体会

实验过程中主要的收获:

(1) 编程实现了基于 SIFT 特征提取算法、K-Means 聚类算法和 SVM 分类器的图像分类算法，熟悉了基于传统特征提取和机器学习方法构建图现象分类器的流程。

(2) 编程实现了基于 VGG16 网络模型的图像分类器。

(3) 对两种不同方式实现的分类器进行对比，了解基于深度神经网络的模型分类效果要远远优于传统特征提取的方法，说明深度神经网络能够捕捉到一些 SIFT 算法捕捉不到的精细的深度信息。

实验过程中的经验总结:

(1) 在编写实验运行文件的时候，要注意读入文件的路径;

(2) 实现深度神经网络提取特征的可视化需要注意很多问题，需要细致地对特征进行拼接或降维。

四、存在的主要问题和建议

对于基于 VGG16 网络的图像分类模型的超参数仍然有优化的空间，可以进一步提高其训练结果的性能。