机器学习课程报告：期末作业

一、实验简介

该实验任务是分成两部分，第一部分是使用机器学习进行人脸分类识别，并给出识别准确率，第二个任务是使用聚类或分类算法发现表情相似的脸图。对于第一个任务，本实验使用卷积神经网络（CNN）进行分类，这个任务相对简单，准确率也比较高，在测试集上达到了 100%。对于第二个任务，尝试使用了 Multinomial Naive Bayes（MNB）、Random Forest（RF）和 CNN 进行分类，准确率都不是非常高。使用 MNB 并使用五折交叉验证得到的评价准确率是 28.59%；使用 CNN 准确率基本在 25%左右；使用 RF 得到的准确率是 41%。

二、使用环境

本实验所使用环境如下：
Ubuntu16.04、16GB 内存、NIVIDA GTX1070（8GB 显存）、编程语言 Python 3.6、框架：TensorFlow 1.8.0 ，Keras 2.0.4 ， Sklearn 0.19.1、IDE 为 Pycharm2018。

三、数据集

本实验使用的数据集是 CMU 公开的人脸识别数据集（下载地址：http://www.cs.cmu.edu/afs/cs.cmu.edu/user/mitchell/ftp/faces.html），此数据集共有 20 个分类（即 20 个人），每人有 32 张脸图。对于每一张脸图，都提供了 3 种大小的版本，如下图所示，从左到右分别是 128*120, 64*60, 32*30 大小的脸图。


128*120


64*60


32*30

图片存储格式是.pgm，对于 128*120 的图片，其图片模式为 P2。对于后两种大小的图片，图片模式为 P5。P2 和 P5 文件读取的方式有所不同，P5 文件可以使

用 Pillow 方便读取，对于 P2 模式的文件需要读取数据存储像素点。本实验使用的是 128*120 大小的脸图，因为有 640 张，因此读取数据的维度为（640，120,128），并需要将图片名称转化为标签（0-19），代码如下：

```
def readData():
    """
    将 faces 文件中的数据读取出来，并获取标签信息
    :return: X，一个 3 维矩阵（图片数，LENGTH,WIDTH），y 标签，是一个长度为图片数的向量
    """
    X = []
    labels_index = {}        # dictionary mapping label name to numeric id
    y = []      # list of label ids
    for dir_name in sorted(os.listdir(BASE_DIR)):
        path = os.path.join(BASE_DIR, dir_name)
        if os.path.isdir(path):
            label_id = len(labels_index)    # 文件夹 id
            labels_index[dir_name] = label_id
            for fname in sorted(os.listdir(path)):
                if not fname.endswith("_2.pgm") and not fname.endswith("_4.pgm"):
                    fpath = os.path.join(path, fname)
                    data = readpgm(fpath)      # data[0]为数据，data[1]为 shape
                    # print(data[1])
                    if data[1]==(LENGTH, WIDTH):      # 图片(120, 128)
                        X.append(data[0].reshape(LENGTH, WIDTH))
                        y.append(label_id)
    return X, y
```

本数据集共有 4 种表情，分别为：angry，happy，neutral 和 sad，数据处理如上，这部分处理过的数据将用于第二个任务的相似表情识别。

四、人脸识别任务

1、模型介绍

本实验使用了 4 层卷积层和 2 层全连接层来构建网络，使用 ReLu 作为激活函数，其设置和输出维度如下：

输入维度	操作	步长	输出维度
(?, 120, 128, 1)	卷积	3*3	(?, 120, 128, 32)
(?, 120, 128, 32)	池化	2*2	(?, 60, 64, 32)
(?, 60, 64, 32)	卷积	3*3	(?, 60, 64, 32)
(?, 60, 64, 32)	池化	2*2	(?, 30, 32, 32)
(?, 30, 32, 32)	卷积	3*3	(?, 30, 32, 32)
(?, 30, 32, 32)	池化	2*2	(?, 15, 16, 32)
(?, 15, 16, 32)	卷积	3*3	(?, 15, 16, 32)
(?, 15, 16, 32)	池化	2*2	(?, 8, 8, 32)

其中，四元组第二个和第三个元素表示图片的长和宽，第四个元素表示输出维度。卷积采取补 0 策略，保留边界处的卷积结果。

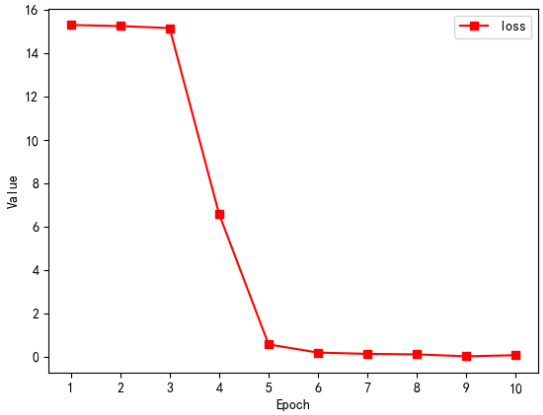
代码如下：

```
def cnn_model():
    model = Sequential()
    model.add(Convolution2D(
        input_shape = (LENGTH, WIDTH, 1),    # (?, 120, 128, 1)
        filters=32,
        kernel_size=3,
        strides=1,
        padding="same",
        data_format="channels_last"
    ))       # (?, 120, 128, 32)
    model.add(Activation('relu'))
    model.add(MaxPooling2D(
        pool_size=2,
        strides=2,
        data_format="channels_last"
    ))         # (?, 60, 64, 32)
    model.add(Convolution2D(32, 3, strides=1, padding='same', data_format='channels_last
'))    # (?, 60, 64, 32)
    model.add(Activation('relu'))
    model.add(MaxPooling2D(2, 2, data_format='channels_last'))      # (?, 30, 32, 32)
    model.add(Convolution2D(32, 3, strides=1, padding='same', data_format='channels_last
'))  # (?, 30, 32, 32)
    model.add(Activation('relu'))
    model.add(MaxPooling2D(2, 2, data_format='channels_last'))  # (?, 15, 16, 32)
    model.add(Convolution2D(32, 3, strides=1, padding='same', data_format='channels_last
'))  # (?, 15, 16, 32)
    model.add(Activation('relu'))
    model.add(MaxPooling2D(2, 2, data_format='channels_last'))  # (?, 8, 8, 32)
    model.add(Flatten())
    model.add(Dense(8*8*32, activation='relu'))
```

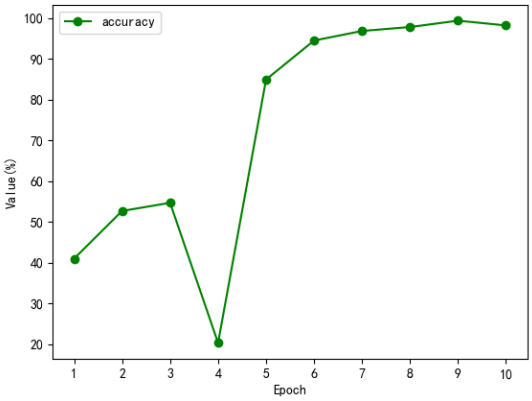
```
    model.add(Dropout(0.8))
    model.add(Dense(NUM_CLASSES, activation='softmax'))
    model.compile(optimizer=keras.optimizers.RMSprop(),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

2、实验结果分析

在原始使用 Adam 作为优化器时，准确率为 99.8%左右，而后将 Adam 换成了 RMSprop，算法的准确率达到了 100%，实验设置 epoches=10，batch_size=16。当优化器为 RMSprop 时，训练集上每个 epoches 后的损失值和准确率变化如下图所示：



损失值变化曲线



准确率变化曲线

五、相似表情发现任务

相似表情发现任务和人脸识别任务类似，只是分类数变成了四类，这里使用三种分类方法，Multinomial Naive Bayes、Random Forest 和 CNN 方法。

1、Multinomial Naive Bayes 方法

使用 Multinomial Naive Bayes 进行分类，并使用 5 折交叉验证，训练集和测试集的比例为 4:1，代码如下：

```
param_range = np.logspace(-6, -1, 5)
```

```
train_accuracy, validation_accuracy = validation_curve(MultinomialNB(), X_data, y_data, cv=5, param_name="alpha", param_range=param_range, scoring="accuracy")
```

算法在训练集上的准确率为 33.09%，在测试集上的准确率为 28.59%。

2、Random Forest 方法

使用随机森林进行分类的代码如下：

```
clf = RandomForestClassifier(n_estimators=10, max_depth=1, random_state=33)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(classification_report(y_pred, y_test, target_names=["angry", "happy", "neutral", "sad"]))
```

其混淆矩阵输出如下：

	precision	recall	f1-score	support
angry	0.03	0.50	0.05	2
happy	0.55	0.22	0.31	74
neutral	0.30	0.30	0.30	33
sad	0.10	0.16	0.12	19
avg / total	0.41	0.23	0.28	128

可以看出，在 128 张预测的图片中，有 2 张被预测为“angry”，有 74 张被预测为“happy”，有 33 张被预测为“neutral”，有 19 张被预测为“sad”，最终的准确率为 41%，召回率为 23%，F 值为 28%。

3、CNN 方法

使用 CNN 方法，若直接重用人脸识别任务的代码，发现预测的准确率非常低，通过调参、网络层数的增删，可以将准确率提升到 28.9%，但是这个预测准确率仍然比较低。因此，尝试通过图片增强方式来扩展训练数据。图片增强方式，就是将图片进行翻转、平移等操作。图片增强的代码如下：

```
def data_augment(X_train, y_train):
    X = np.reshape(X_train, (-1, X_train.shape[1], X_train.shape[2], 1))
    y = np.reshape(y_train,(y_train.shape[0],1))
    for i in range(X.shape[0]):
        x = X[i,:]
        # print(x.shape)
        x = x.reshape((1,) + x.shape) #datagen.flow 要求 rank 为 4
        # print(x.shape)
```

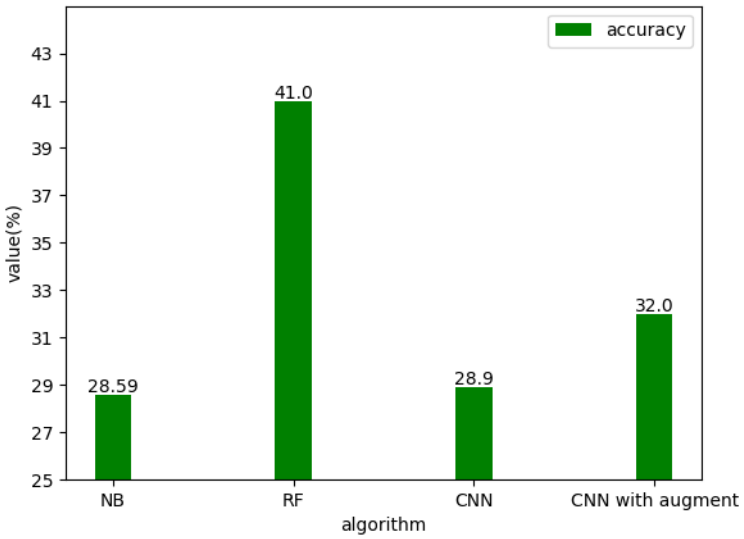
```
        datagen.fit(x)
        prefix = y[i][0]
        print(prefix)
        counter = 0
        for batch in datagen.flow(x, batch_size=4 , save_to_dir='pic', save_prefix=prefix,
save_format='jpg'):
            counter += 1
            if counter > 10:      # 每一张原始图片，生成 10 张变形图片
                break  # 否则生成器会退出循环
```

增强后的图片如下所示：

图片增强的时候只针对训练集数据进行增强，使用这些训练集训练模型，最后使用未增强的测试集进行测试，最终准确率为 32.0%。

4、实验结果对比

实验对比结果图如下，使用 Multinomial Naive Bayes 结果可以达到 28.59%，使用 Random Forest 结果可以达到 41.0%，使用 CNN 可以达到 28.9%，使用 CNN 并使用图片增强后的准确率可以达到 32%。对于表情相似度发现的任务，因为脸部表情属于面部微表情，可能只是嘴角做出上扬的动作，表情就发生了变化，因此其分类比较困难，准确率比较低。另外，训练数据也比较少。



实验结果对比图

六、总结

本次实验主要做了人脸分类识别和发现表情相似的脸图两个任务。对于人脸识别的任务，本实验使用卷积神经网络（CNN）进行分类，这个任务相对简单，在测试集上达到了 100%。对于第二个任务，使用了 Multinomial Naive Bayes（MNB）、Random Forest（RF）和 CNN 进行分类并使用了图片增强技术扩展数据集，准确率都不是非常高。针对相似表情发现的任务，在今后可以尝试通过一些算法先将前后景分开，再针对前景进行训练，或者扣去局部面部表情，使用这些局部面部表情进行训练，效果可能会有所提升。

本次实验加强了对使用 CNN 进行图片分类的学习，但是就算法性能的优化技巧等，仍然非常缺乏，这也是在今后需要继续提升的地方。