南华大学

计 算 机 学 院

课程设计报告

（ 2020~2021 学年度 第一学期 ）

课程名称 机器学习

设计名称 服装分类

姓名 徐望成 学号 20184352135

专业 人工智能 班级 18 软智 01 班

地点 8-411 教师 熊东平

目录

一、课题设计背景

1.1、赛题名：Fashion-MNIST 分类练习

1.2 赛道：训练赛道

1.3 背景：

图像分类（image classification）是计算机视觉领域中最简单最基础的任务，学习研究图像分类是每个计算机视觉研究者的必经之路，图像分类网络也是很多更复杂任务（如目标检测、语义分割等）算法的基础。本练习赛旨在让选手们用图像分类任务来以赛代练、熟悉深度学习框架和比赛流程。

在图像分类学习中，MNIST 数据集常被用来作为入门教学数据集。但是，MNIST 数据集存在一些问题：首先，MNIST 数据集对于现在的卷积神经网络来说过于简单，SOTA 模型的分类精度达到了 99.84%，甚至传统机器学习方法也能达到 97%的精度，因此模型的精度在此达到了饱和，几乎没有提升的空间；再者，有些专家对 MNIST 数据集提出了质疑，比如谷歌的深度学习专家、Keras 的作者 François Chollet 曾表示：“MNIST 存在很多问题，但最重要的是，它真的不具有计算机视觉任务的代表性。”并补充道：“很多好点子（比如 batch norm）在 MNIST 上效果差，但相反的，一些差的方法可能在 MNIST 产生好效果，却不能迁移到真实计算机视觉任务中。”

总之，用 MNIST 数据集来学习计算机视觉既不够有难度，又不便学习到能运用到真实计算机视觉任务中的方法。因此，本练习赛采用和 MNIST 同等规模但更有难度的数据集 Fashion-MNIST（github 链接：https://github.com/zalandoresearch/fashion-mnist），Fashion-MNIST 由 60000 张训练集图像、10000 张测试集图像及对应的标签构成，每张图像是分辨率为 28x28 的灰度图像，包含 10 种分类：T 恤、裤子、套头衫、连衣裙、大衣、凉鞋、衬衫、运动鞋、包、短靴。

本练习赛的参赛者可以使用 Tensorflow、Keras、Pytorch、Paddlepaddle 等开源深度学习框架来进行模型的搭建、训练和预测。

1.4、任务：

本任务旨在构建一种机器学习算法模型，建立振动信号和“亚健康”状态之间的关系，通过一系列手段，使得模型具有更高的准确率、更好的鲁棒性和泛化性。

1.5、赛题网址

赛题网址为：https://www.datafountain.cn/competitions/490

二、设计方案概述

2.2、问题分析

本项目服装分类问题一共有 10 个类别，采用传统的机器学习算法比如支持向量机、线性模型，单独的这两类模型只能进行一对一分类，如果需要解决多分类问题，则需要进行组合，常见的组合有“一对一”（One VS One）、“一对多”（One VS Rest）、“多对多”（Many VS Many）。如果采用一对一组合，则需要有 $C_{10}^{2}=\frac{10\times(10-1)}{2}=45$ 个分类器；如果采用一对多组合，则需要有 10 个分类器。可以看到如果类别越多，需要使用越多的分类器，因而训练时间也会大大的加长。

如果采用主观贝叶斯网络进行分析，则需要计算 28x28=784 个类概率值，在进行合成的时候，尽管我们可以通过采用取对数的方法，，但是当 n=784，可以近似的看作 $n\to\infty$，计算机非常容易出现数据下溢。

2.1、解决方案

项目采用深度学习框架解决服装分类学习任务问题，输入数据为 28x28 的图像，我们可以转换成一个维度为 784 的向量。输出数据一共需要有 10 类：T 恤、裤子、套头衫、连衣裙、大衣、凉鞋、衬衫、运动鞋、包、短靴，我们可以采用 0~9 的编码方法，每一个数字与每

一个分类一一对应。输入数据与输出数据之间采用多层神经网络，进行训练、验证、测试。

三、具体实现

3.1、环境配置

1）操作系统：Windows 10

2）程序语言：python 3.7.8

3）学习框架：tensorflow 2.4.1

3.2、主程序

```
import tensorflow as tf
from keras.datasets import fashion_mnist
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt

def show(images,labels):
    class_names = ['T-shirt', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
    for i in range(20):
        plt.subplot(5, 4, i + 1)
        plt.xticks([])
        plt.yticks([])
        plt.grid(False)
        plt.imshow(images[i], cmap=plt.cm.binary)
        plt.xlabel(class_names[labels[i]])
    plt.show()


(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

```
# 展示训练集前 20 张图片和标签
show(train_images,train_labels)
## 将 train_images 与 test_images 这些值除以 255 缩小至 0-1 之间，以便将其送到神经网络中。
train_images = train_images / 255.0
test_images = test_images / 255.0
plt.figure(figsize=(10, 10))
# 为了验证数据格式是否正确、是否已准备好构建和训练网络，此处显示训练集中前 20 个图像，并在每个图像下方显示类名称。



# 构建神经网络
# 网络第一层 keras.layers.Flatten ：将图像格式从二维数组（28x28 像素）转换成一维数组（28x28=784 像素）。
# 网络第二层 keras.layers.Dense ：全连接层，128 个神经元，激活函数采用线性整流函数 ReLU。
# 网络第三层 keras.layers.Dense ：返回一个长度为 10 的 logits 数组（线性输出）。
model = keras.Sequential(
      [keras.layers.Flatten(input_shape=(28, 28)),
       keras.layers.Dense(256, activation='relu'),
       keras.layers.Dense(64, activation='relu'),
       keras.layers.Dense(10)])

# 编译模型
# 优化器：决定模型如何根据其看到的数据和自身的损失函数进行更新。
# 损失函数：用于测量模型在训练期间的准确率，希望最小化此函数，以便将模型“引导”到正确的方向上。
# 准确率：被正确分类的图像的比率。
model.compile(optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                    metrics=['accuracy'])

# 将训练数据送至模型，使用训练集的全部数据对模型进行 20 次训练。
model.fit(train_images, train_labels, epochs=20)

# 在测试集上评估
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
print('Test accuracy:', test_acc)
print('Test loss:', test_loss)

# 经过训练后，使用它对一些图像进行预测。模型具有线性输出，即 logits 。此处再附加一个 softmax 层，将 logits 转换成更容易理解的概率。
probability_model = tf.keras.Sequential([model, tf.keras.layers.Softmax()])
predictions = probability_model.predict(test_images)
```

```
# 展示测试集前 20 张图片和真实标签
show(test_images,test_labels)
# 展示测试集前 20 张图片的预测标签
pre_labels = [np.argmax(prediction) for prediction in predictions]
show(test_images,pre_labels)
```

3.3、生成提交文件代码

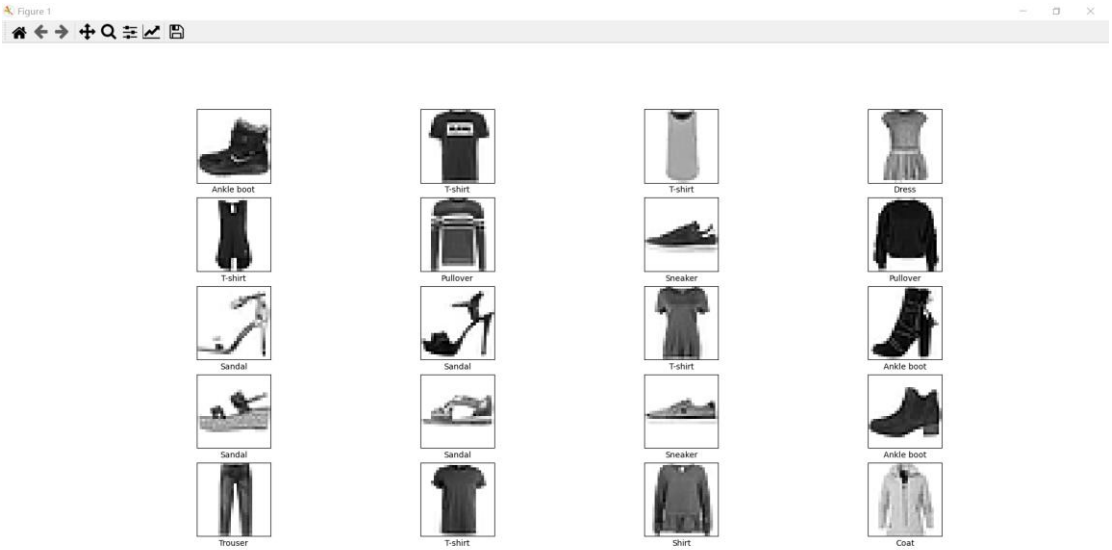
```
file = open("fashion-mnist_test_data.csv", 'r')
file.readline()
new_image = np.loadtxt(file, delimiter=',', skiprows=0)
file.close()

new_image = np.array(new_image)
new_image = np.delete(new_image, 0, axis=1)
new_image = np.reshape(new_image, test_images.shape)

predictions = probability_model.predict(new_image)
with open("result.csv", "w") as f:
    for i, prediction in enumerate(predictions):
        content = str(i) + ".jpg," + str(np.argmax(prediction)) + "\n"
        f.write(content)
```

四、结果及分析

4.1、训练数据前 20 个图像



4.2、测试集前 20 个数据和真实标签

4.3、测试集前 20 个数据和预测标签

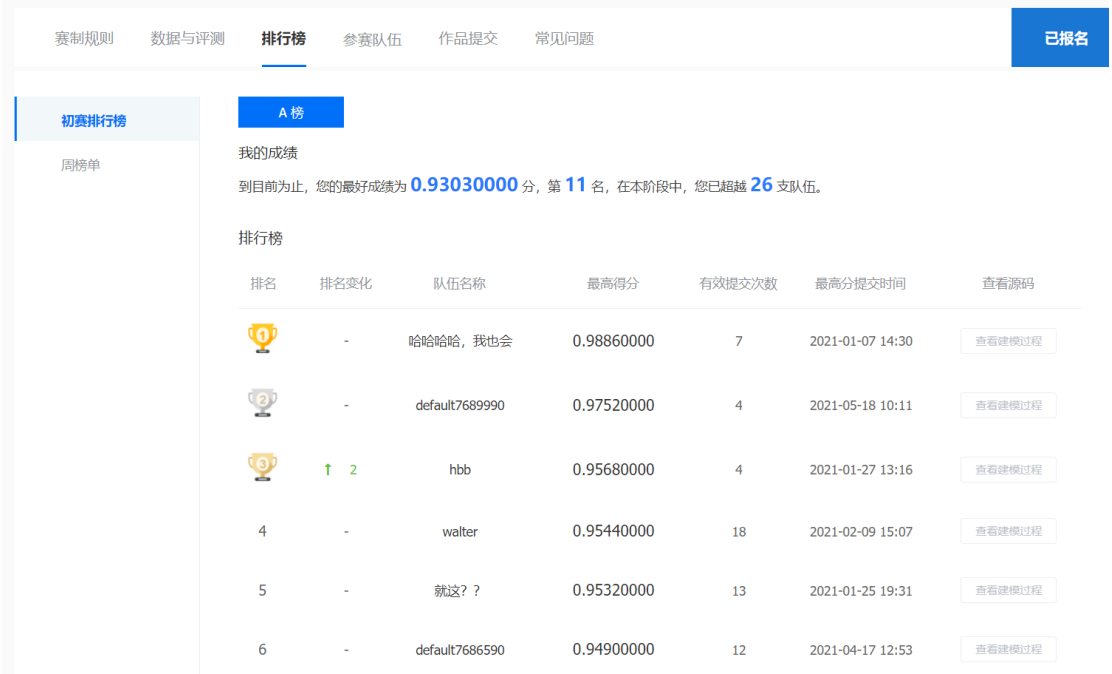
4.4、程序运行片段

```
1875/1875 [==============================] - 5s 2ms/step - loss: 0.6224 - accuracy: 0.7812
Epoch 2/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3713 - accuracy: 0.8634
Epoch 3/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3265 - accuracy: 0.8793
Epoch 4/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3027 - accuracy: 0.8858
Epoch 5/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2808 - accuracy: 0.8936
Epoch 6/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2703 - accuracy: 0.8973
Epoch 7/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2564 - accuracy: 0.9015
Epoch 8/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2480 - accuracy: 0.9061
Epoch 9/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2332 - accuracy: 0.9103
Epoch 10/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2242 - accuracy: 0.9157
Epoch 11/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2229 - accuracy: 0.9154
Epoch 12/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2123 - accuracy: 0.9188
Epoch 13/20
```

```
1875/1875 [==============================] - 4s 2ms/step - loss: 0.2020 - accuracy: 0.9241
Epoch 14/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1958 - accuracy: 0.9239
Epoch 15/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1915 - accuracy: 0.9256
Epoch 16/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1833 - accuracy: 0.9313
Epoch 17/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1799 - accuracy: 0.9311
Epoch 18/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1712 - accuracy: 0.9348
Epoch 19/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1705 - accuracy: 0.9345
Epoch 20/20
1875/1875 [==============================] - 4s 2ms/step - loss: 0.1628 - accuracy: 0.9382
313/313 - 1s - loss: 0.3888 - accuracy: 0.8825
Test accuracy: 0.8824999928474426
Test loss: 0.38875091075897217
```

可以看到只有第四行第一例的数据是预测错误的，训练准确率为 93.8%，测试准确率为 88.2%。

4.5、比赛提交结果



五、总结

在本次课程设计的过程中，作者遇到自己了之前没有碰到过的问题，但是最后还是通过自己的努力将其解决了。刚开始是选择赛题的问题，作者看到了阿里云强化学习版的超级马里奥赛题，作者打算选择这个赛题的，但是这个赛题需要使用阿里云提供的 GPU，如果使用 GPU 则需要排长队，一时半会难以得到合适的环境。所以作者后面选择了服装分类赛题，这个赛题可以在本地运行。

选择好赛题之后，便需要选择一个合适的模型来训练分类器，作者最后觉得神经网络是比较适合本任务。

总的来说，作者发现自身对人工智能方面的知识还比较的欠缺，虽然学到了机器学习、深度学习、人工智能的主要算法，但是缺乏实战经验，未来作者也会填补这一方面的空缺，争取早日成为行业精英。