一、运行环境说明

数据集来源：THE MNIST DATABASE of handwritten digits

（<http://yann.lecun.com/exdb/mnist/>）

CPU：AMD R5 4600H

GPU: NVIDIA GeForce GTX 1650

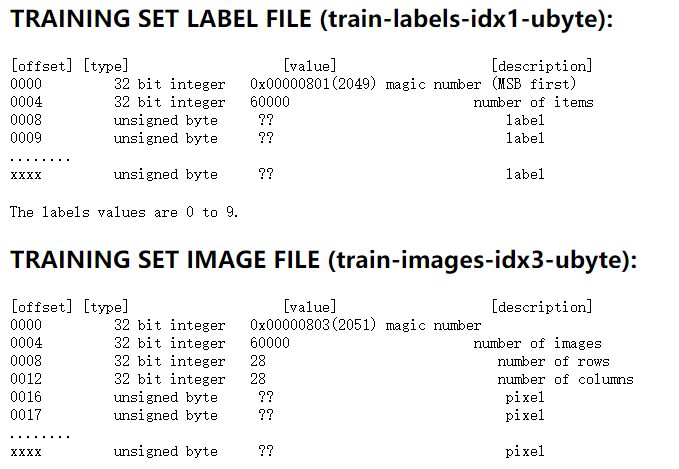
内存：16G DDR4 3200MHz

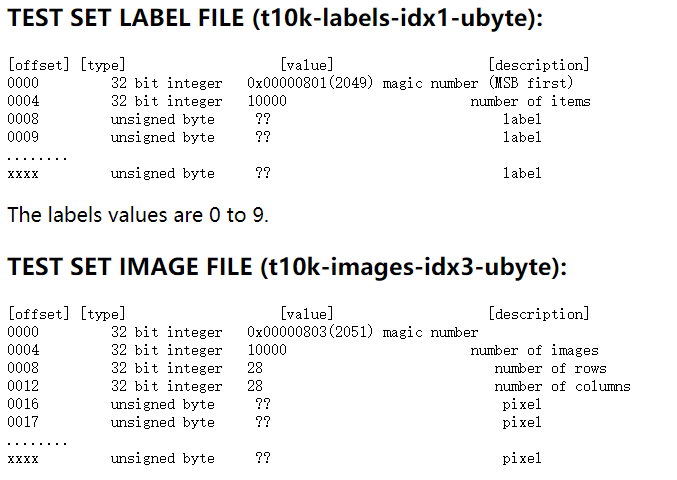
编译环境：pycharm+python3.8+tensorflow2.2.0

源代码已上传我的GitHub：https://github.com/HugoWang232608/minst-

二、数据集说明

Mnist训练集中共有60000张图片，测试集中共有10000张图片，图片大小为28×28，具体文件格式说明如下：





三、主要源代码说明

将数据集进行处理并展示部分数据和标签：

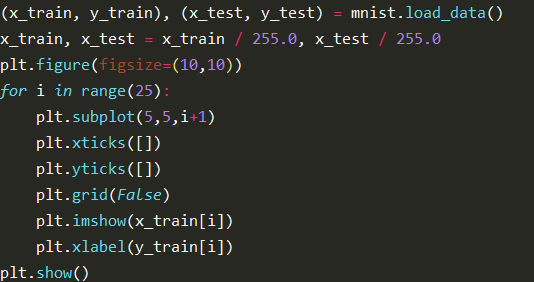


图 1 部分源代码

结果：

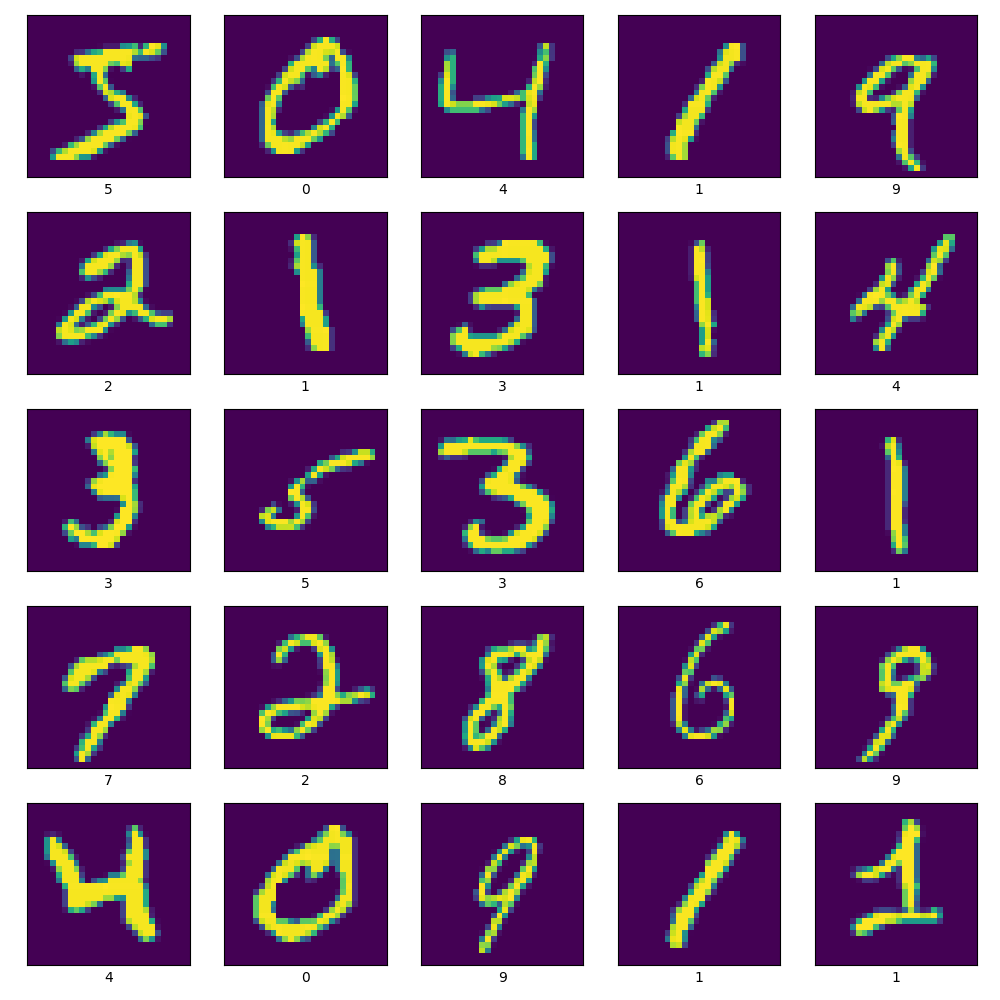


图 2 部分数据展示

神经网络模型：

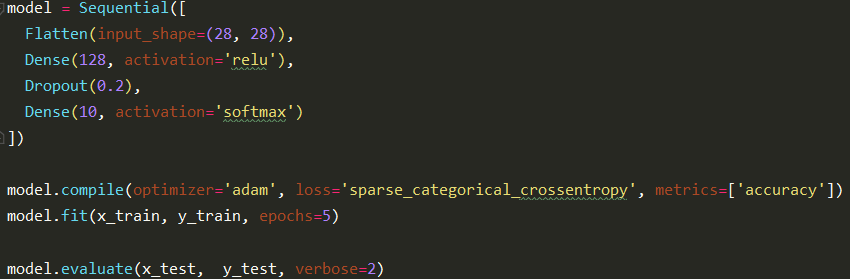


图 3 模型源代码

该网络的第一层 Flatten 将图像格式从二维数组（28 x 28 像素）转换成一维数组（28 x 28 = 784 像素）。将该层视为图像中未堆叠的像素行并将其排列起来。该层没有要学习的参数，它只会重新格式化数据。

展平像素后，网络会包括两个 Dense 层的序列。它们是密集连接或全连接神经层。第一个 Dense 层有 128 个节点（或神经元）。第二个（也是最后一个）层会返回一个长度为 10 的 logits 数组。每个节点都包含一个得分，用来表示当前图像属于 10 个数字中的哪一个。

设置Dropout是在每次神经网络的训练过程中，使得部分神经元工作而另外一部分神经元不工作。而测试的时候激活所有神经元，用所有的神经元进行测试。这样便可以有效的缓解过拟合，提高模型的准确率。

损失函数 - 用于测量模型在训练期间的准确率。最小化此函数，以便将模型“引导”到正确的方向上。

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。

指标 - 用于监控训练和测试步骤。

结果：准确率为97.86%

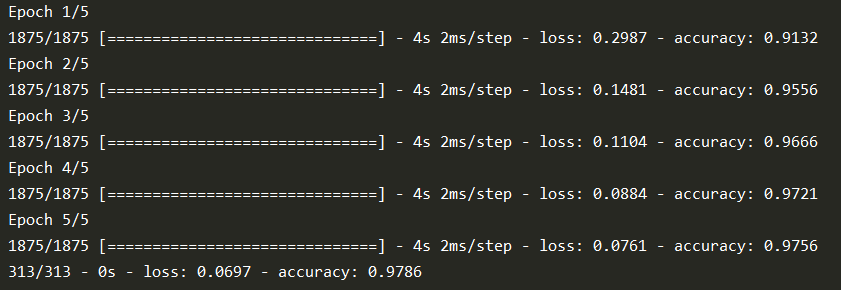
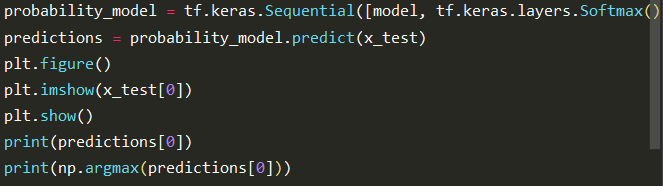
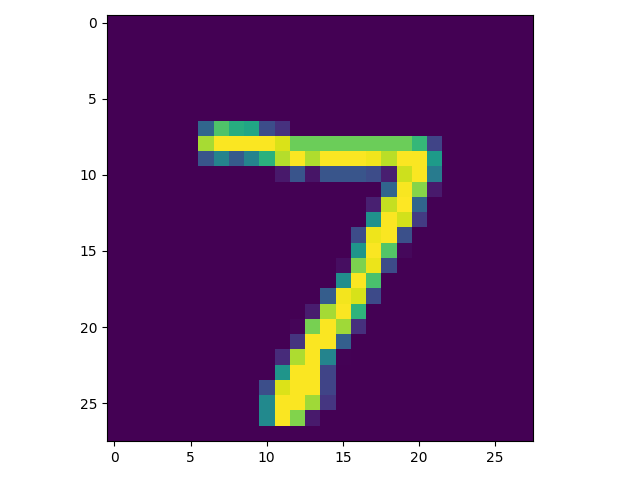


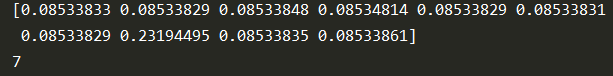
图 4 模型结果

用单张图片对模型进行测试：

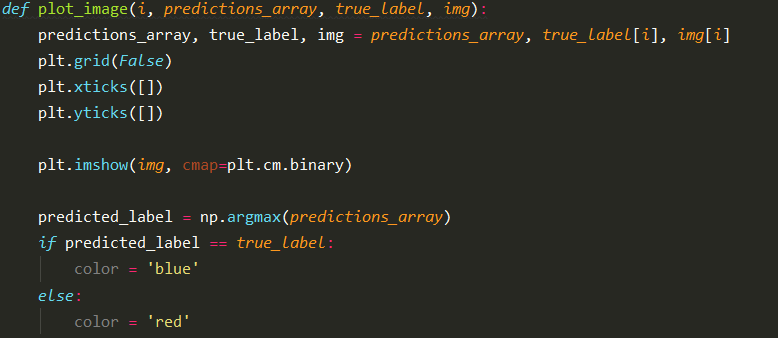


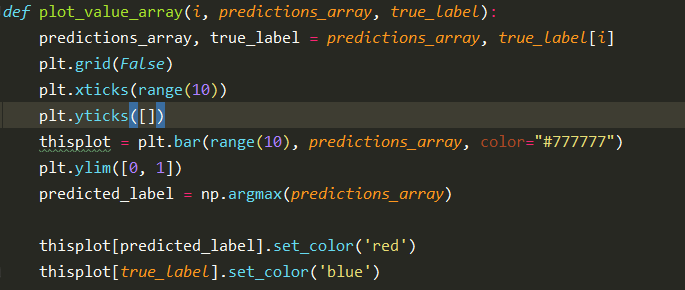
结果：

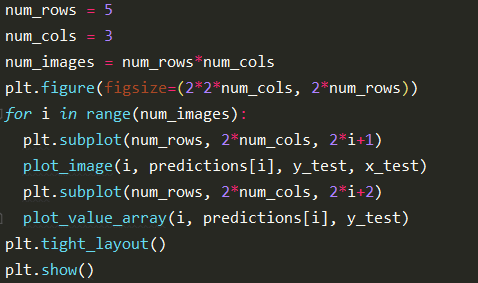




继续展示模型测试：







结果：

