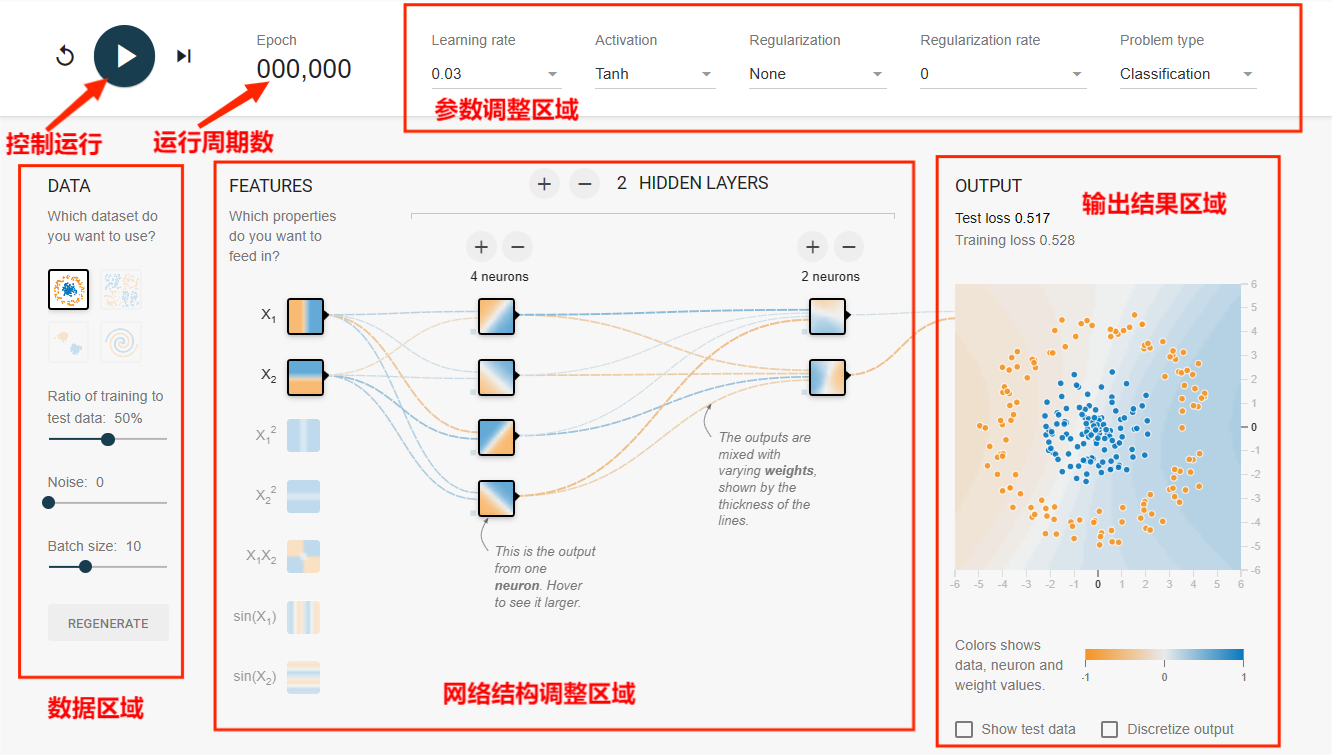
**一、tensorflow playground试用报告**

1、使用说明

（1）主界面



（2）功能详解

①控制运行

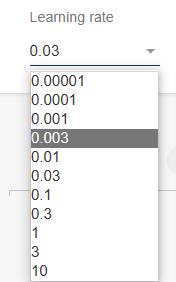
从左到右三个功能分别是：（a）重启；（b）运行；（c）一次运行一个周期

②运行周期数

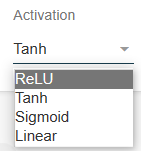
用于查看训练的周期数，周期增长的速度可以通过①来调整

③参数调整区域

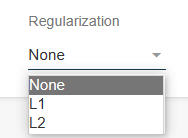
Learning rate：学习率（是一个超参数，在梯度下降算法中会用到；学习率是人为根据实际情况来设定）。值设定固定，如下所示：



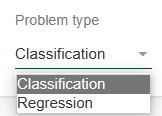
Activation：激活函数（默认为非线性函数Tanh；如果对于线性分类问题，这里可以不使用激活函数）。函数类型固定，如下所示：



Regularization：正则化（正则化是利用范数解决过拟合的问题）。 正则函数固定，如下所示：

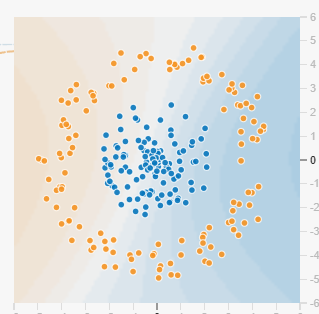


Problem type：问题类型，这里包含分类和回归两大问题，如下所示：



④数据区域

DATA：数据集类型（这里提供了四种数据集，默认选中第一种；被选中的数据也会显示在最右侧的OUTPUT中；在这个数据中，我们可以看到二维平面上有蓝色和黄色的小点；每一个小点代表一个样例例子；点的颜色代表样例的标签；因为只有两种颜色，所以这里是一个二分类问题；如果我们以判断某工厂零件是否合格为例子来说明，那么黄色就代表不合格零件，蓝色就代表合格零件）。



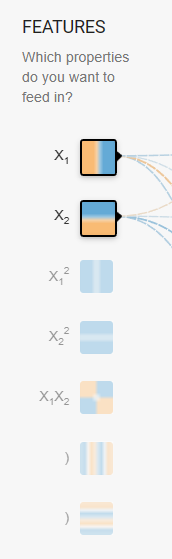
Ratio of training to test：数据用于测试的比例（直接对进度条进行操作即可调整）。

Noise：对数据中引入噪声。

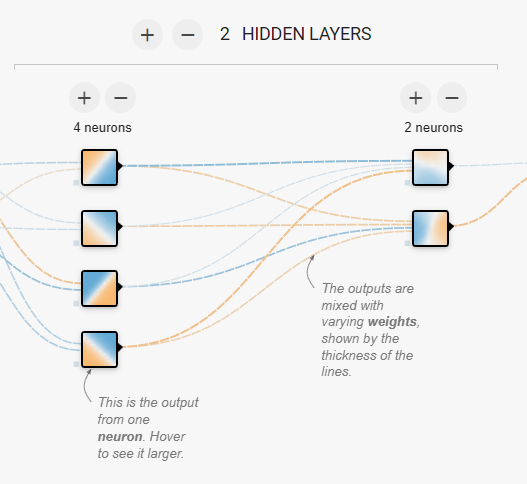
Batch size：调整batch size的大小，也就是一次训练使用的样本数量

⑤网络结构调整区域

FEATURES：特征提取层（为了将一个实际问题对应到空间中的点，首先需要提取数据的特征）。其中包括很多提取方法，具体如下所示：



HIDDEN LAYERS ：隐藏层（在输入和输出之间的神经网络称为隐藏层；一般神经网络的隐藏层越多这个神经网络越深）。我们可以调整隐藏层的数量，以及每个隐藏层的神经元数量，具体如下所示：



总的来说，直接通过点击各个图标即可选择Features的类型，对于隐藏层的操作，可以直接选择加减号即可获得想要的隐藏层层数以及每层的神经元个数。

⑥输出结果区域

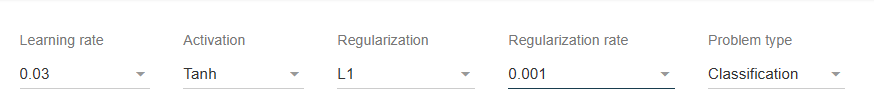
设置完上面的参数，点击运行即可观测到输出结果的变化。

如果选择的是分类问题，即可看到明显的边界变化以及loss在不断减小的情况，点击show test data可以显示未参与训练的test数据集的情况，点击Discretize output可以看到离散化后的结果。

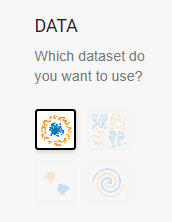
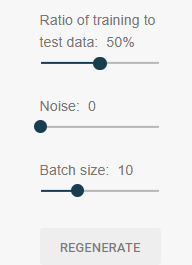
2、运行演示

（1）测试一

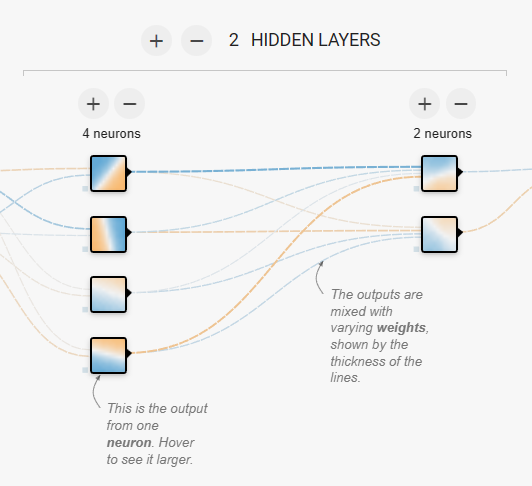
参数设置：学习率0.03，激活函数Tanh，正则项L1 比例0.001，问题类型Classification



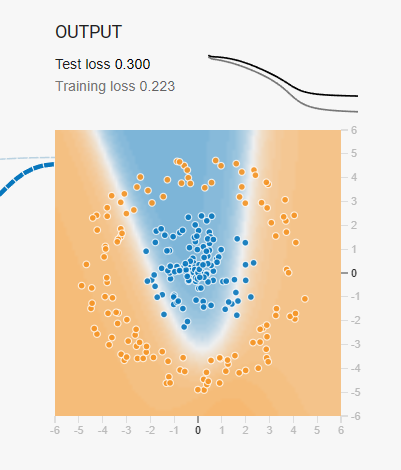
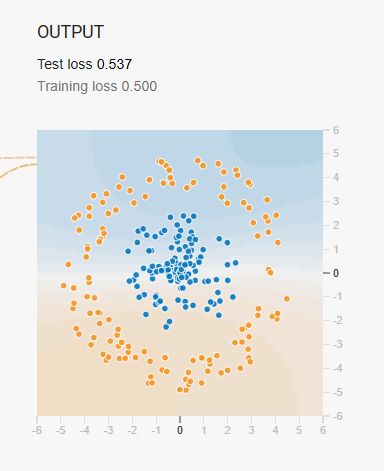
数据：选择左上第一个

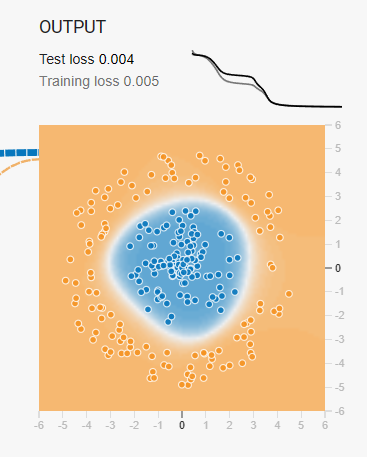
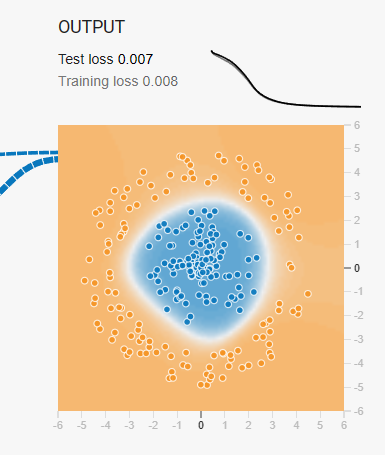
 

网络结构：选择含两个隐藏层，第一个隐藏层4个神经元，第二个隐藏层2个神经元

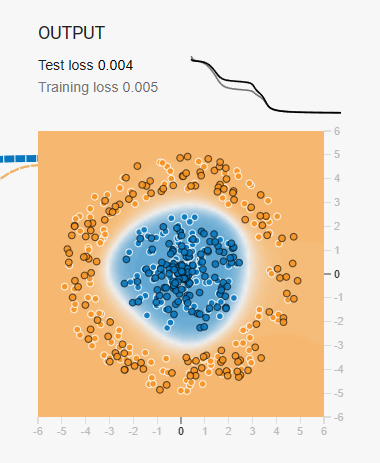


运行结果（间隔50个epoch）：从中我们可以看到，随着训练的不断进行，模型的分类效果越来越好，在第100个epoch时，基本上已经能够实现两种数据的分类。

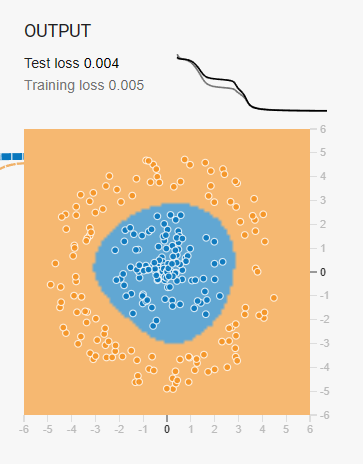




150个epoch的train\_data和test\_data对比：

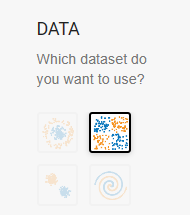


150个epoch的离散化结果：

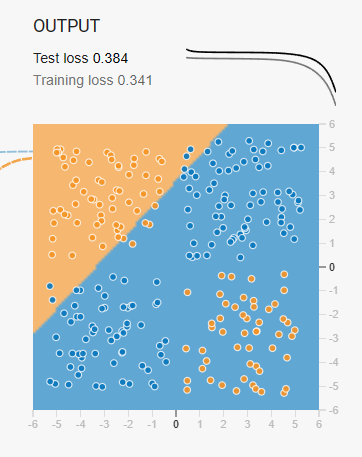
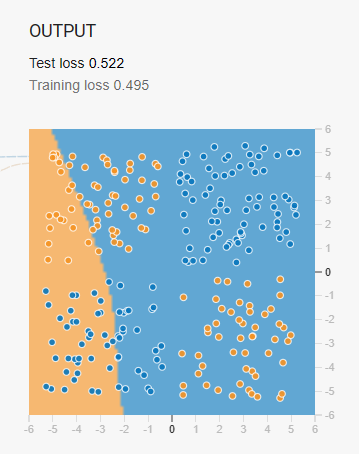


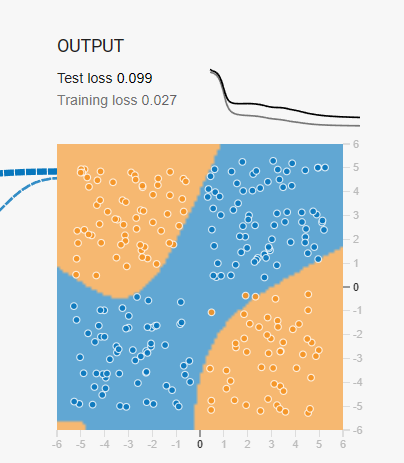
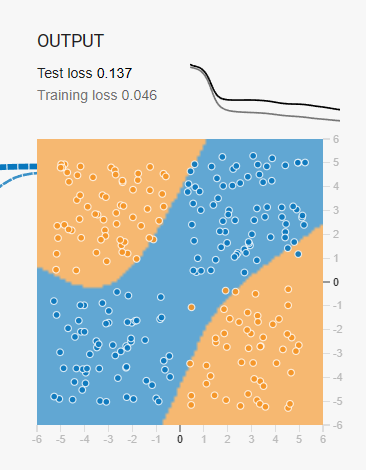
（2）测试二

此部分在测试一的基础上，将数据集换成第二个，其余不变：



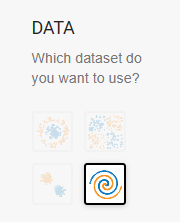
运行结果：从中，我们可以针对分布比较杂乱的数据，我们的模型分类效果也比较好，最终也是实现了两者的分类



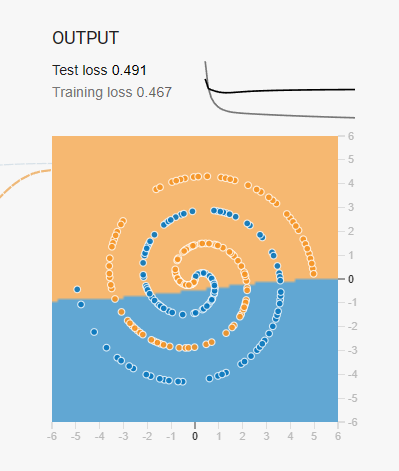
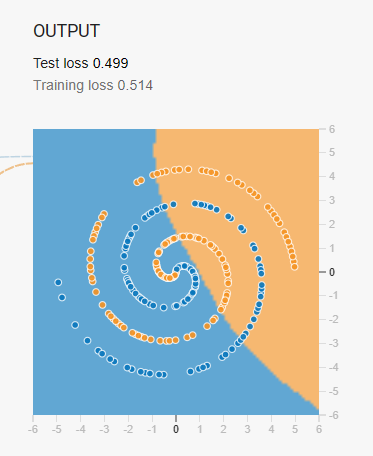


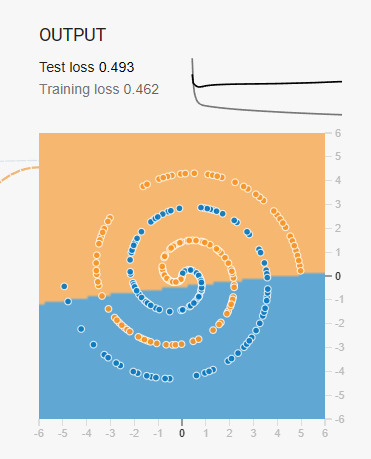
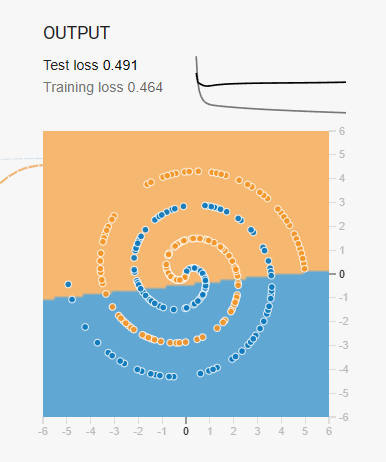
（3）测试三

此部分在测试一的基础上，进一步提升了数据分布的难度，选择了第四个数据集：



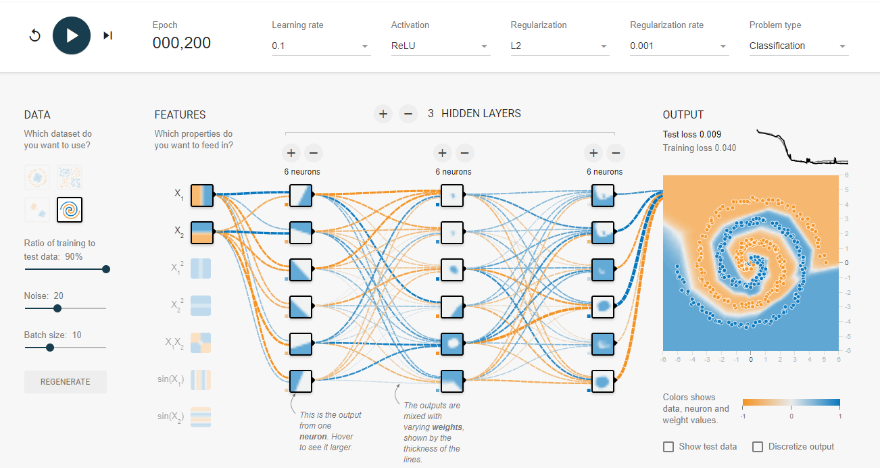
运行结果：从中我们可以看到，针对这种复杂的数据分类，我们同样的模型，最终收敛的效果明显不一样，在这个数据集中，模型明显学不到实际的数据分布



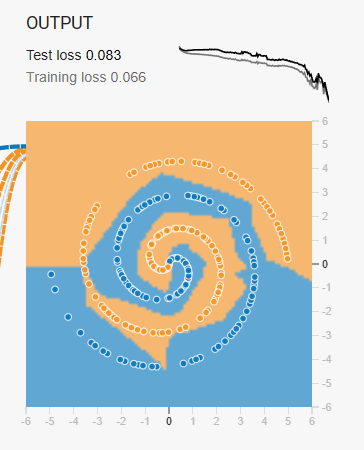
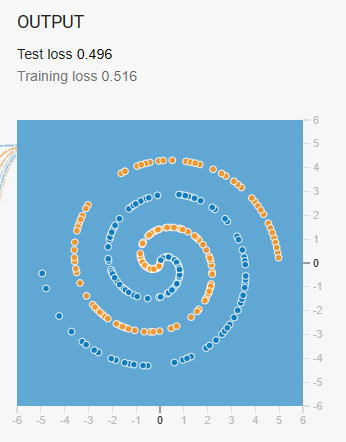


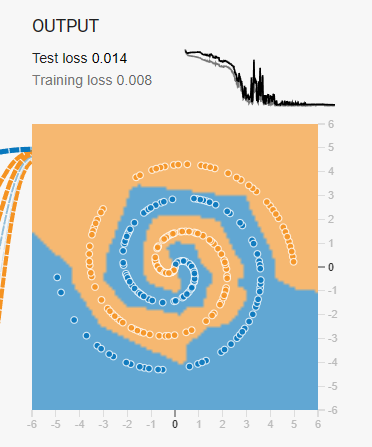
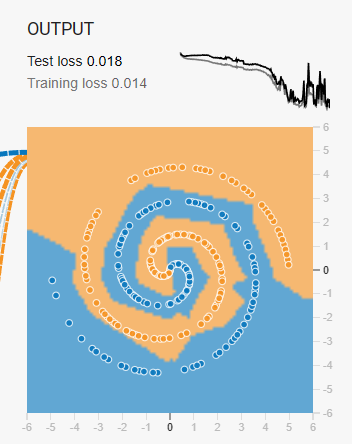
（4）测试四

这部分在测试三的数据集上，调整模型的结构与参数，试图学习其复杂的数据分布，具体来说，我在以下几个方面进行了调整：



加深了隐藏层的数量，以及神经元个数，使网络更加复杂；同时，调整了学习率与激活函数等超参数，最终运行结果如下：

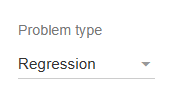




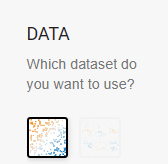
很明显的，我们此次的模型已经成功学习到了此复杂数据集的分布。

（5）测试五

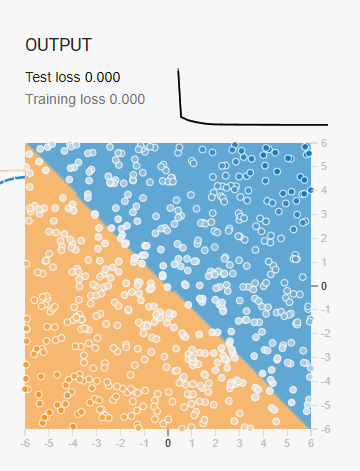
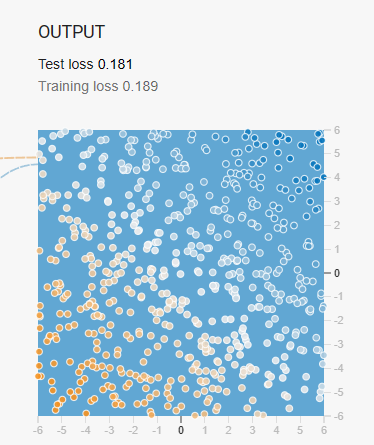
这部分，我将问题类型从分类问题切换到回归问题：



此时的数据集也发生了变换，这里我选择第一个数据集进行训练：

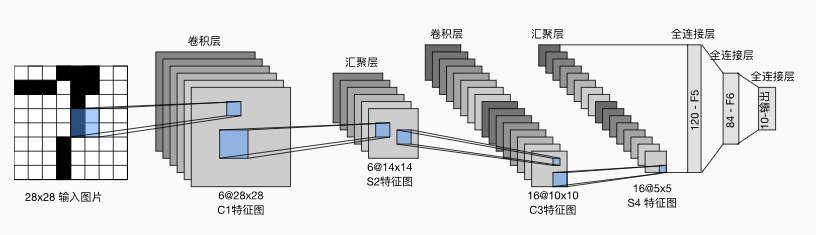


运行结果：从中可以看到，由于此数据集过于简单，我们的模型很快收敛了，并收敛效果很好。



1. LeNet5
2. 模型结构

（1）模型整体结构



（2）模型结构分析

①输入层

模型输入大小28\*28的手写体数字图片

②卷积层

用于提取图片的上下文关系特征，模型中包含两个卷积层，可以提取不同粒度下的图片关系

③池化层

对特征进行下采样，减小特征维度，一般采用在卷积层之后，因此该模型中拥有两个池化层

④全连接层

将提取的特征关系，压缩成一维向量再进一步提取特征

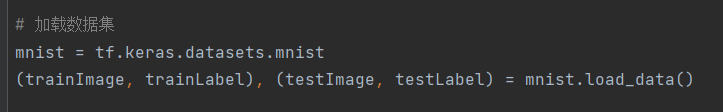
⑤输出层

本质上也是一个全连接层，因为模型是针对MNIST数据集的，这是一个十分类数据集，因此最终输出十个值

2、训练过程及结果

（1）训练数据

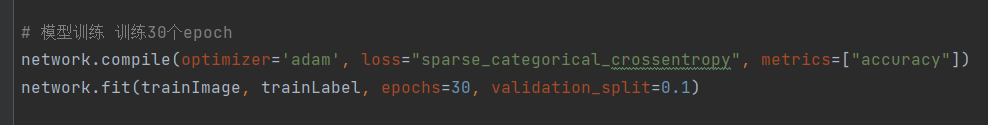
①读取数据集



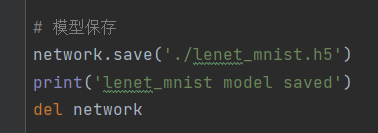
②定义模型



③训练模型

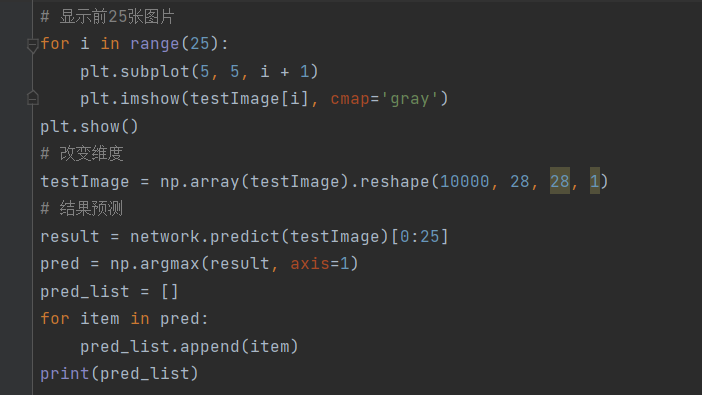


1. 保存训练模型



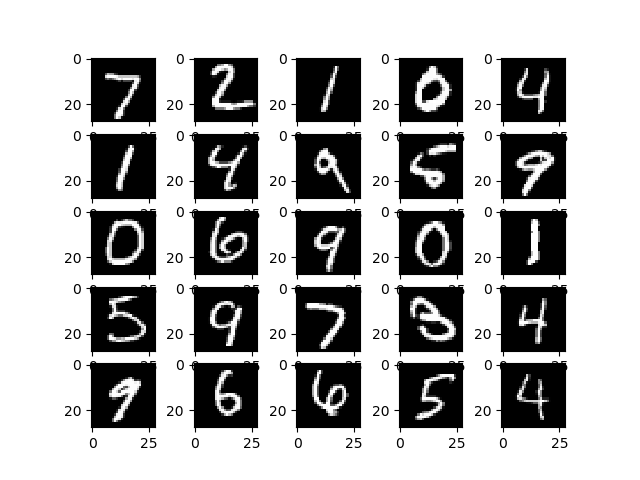
1. 测试数据

此处由于测试集过大，不便于展示，因此只选择了前25个样本进行测试，并且输出对应预测类别。



（4）结果展示

测试数据：



模型预测的类别：



分析：我们可以从左到右，从上到下，一一比对，可以发现每一个图片都是预测正确，能够说明我们的模型成功学习到了MNIST数据集的样本特征