1. **目录文件简述**

在根目录下主要包含四个文件，分别是文件夹data、文件夹saved\_weights\_biases，function.ipynb、train.ipynb、test.ipynb、Ipynb\_importer.py、download.ipynb。其中文件夹data为数据集，主要包含训练数据、训练数据标签、测试数据、测试数据标签，各种数据以.gz形式储存；文件夹saved\_weights\_biases用来保存训练过程中每一轮次更新后的权重和偏置，权重和偏置以.npz格式保存；function.ipynb为自定义函数集，主要包含在代码编写过程中各个模块的函数，共7个函数；train.ipynb为训练脚本，主要作用是针对训练数据与训练数据标签对全连接层进行训练，输出损失值并保存每一轮次的权重值；test.ipynb为测试脚本，主要作用是加载训练好的权重并针对测试数据与测试数据标签，计算正确率；Ipynb\_importer.py文件用于在jupyter notebook中可以调用其他ipynb文件中的自定义函数；download.ipynb文件用于打包当前目录下的所有文件。

1. **function.ipynb**
2. **函数load\_mnist\_images(filename)**

主要用于加载MNIST数据集中.gz文件的图像数据。输入参量filename为MNIST数据集的图像文件路径，输出为numpy格式的大小为(x,28,28)，根据训练集与测试集x为60000或10000。

1. **函数load\_mnist\_labels(filename)**

主要用于加载MNIST数据集中.gz文件的标签数据。输入参量filename为MNIST数据集的标签文件路径，输出为numpy格式的大小为(x,1)，根据训练集与测试集x为60000或10000。

1. **函数get\_mini\_batches(images, labels, batch\_size, shuffle=False)**

主要用于得到batch\_size大小的数据。输入参量images为图像数据，输入格式为(x,28,28)；参量labels为标签数据，输入格式(x,1)；参量batch\_size为单次批量处理数据大小，必须能够整除x；参量shuffle为是否打乱数据，可选择“True”或者“False”；输出为两个变量，分别为images\_batch与labels\_batch，输出格式分别为(x,batch\_size,28,28)、(x,batch\_size,1)。

1. **函数one\_hot(Input, batch\_size)**

主要根据标签数据得到标签数据的的独热码。输入参数Input为标签数据，输入格式为(batch\_size,1)，参数batch\_size为单次批量处理数据大小，输出为标签数据的独热码，输出格式为(batch\_size,10)。

1. **函数relu(Input)**

主要用于对输入层与隐藏层的输出数据的激活。激活函数Relu的公式为

(1.1)

1. **函数softmax(Input)**

主要用于对输出层的输出数据的激活。激活函数softmax的公式为

(1.2)

针对MNIST数据集，分类数量为10类，即K=10，。

1. **函数correct\_num(predict\_labels, true\_labels, batch\_size)**

主要用于计算该轮次预测正确的样本个数。输入参数predict\_labels为预测标签，输入格式为(batch\_size,1)；参数true\_labels为真实标签，输入格式为独热码，大小为(batch\_size,10)，参数batch\_size为单次批量处理数据大小。输出为标量，即该轮次预测正确的样本个数。

1. **train.ipynb**
2. **参数定义**

主要定义四个变量，分别是学习率、训练轮数、单批数量以及是否对数据进行打乱，具体相关参数取值如图3.1。



图3.1 训练参数变量定义与取值

1. **训练数据准备**

训练数据的准备主要分为了定义训练图像、训练标签的路径，加载训练图像，加载训练标签，获得batch大小的数据以及batch的数量，共4部分。其中加载训练图像、加载训练标签采用function.ipynb文件中函数load\_mnist\_images、函数load\_mnist\_labels完成，具体函数作用见上文第二章。获得batch大小的数据以及batch的数量采用function.ipynb文件中函数get\_mini\_batches完成。训练数据的格式为(1200,50,28,28)，训练标签的格式为(1200,50,1)。训练数据准备的相关代码见图3.2所示。

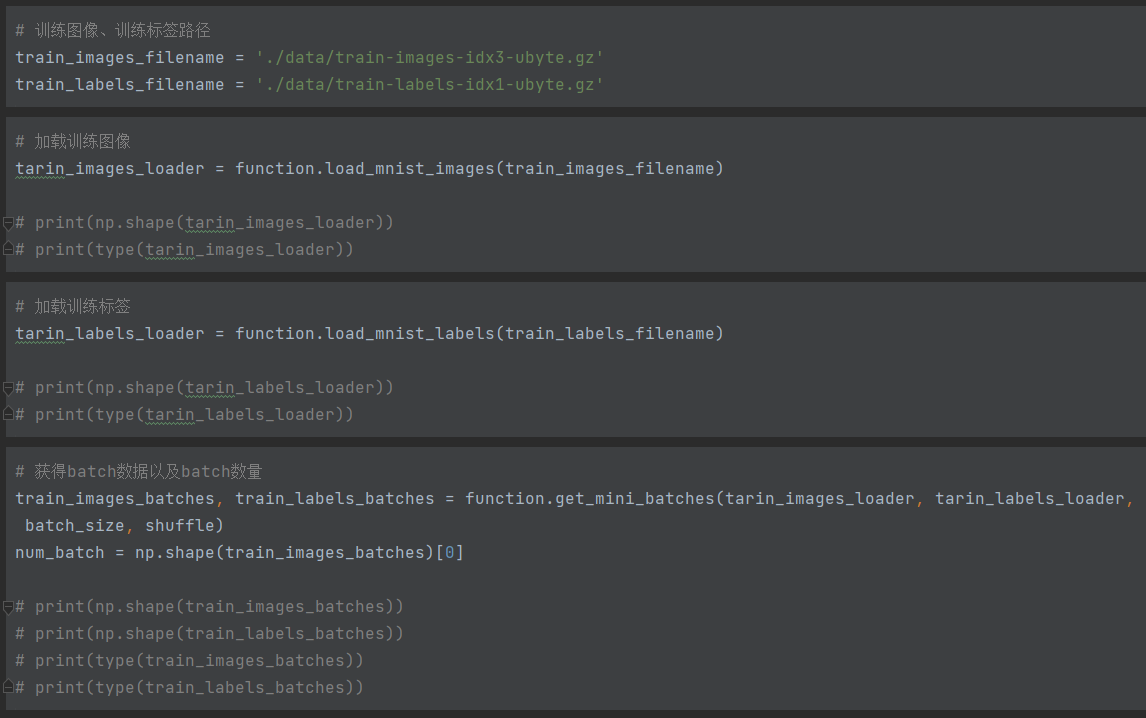


图3.2 训练数据准备

1. **权重与偏置的初始化**

权重初始化主要采用均值为0，标准差为0.01的高斯分布，偏置主要初始化为0。本次实验的网络层数为4，输入与输出的关系分别为、、、。因此需要根据上述所描述的各层输入输出大小分别初始化各层的权重与偏置。权重与偏置初始化相关代码见图3.3，网络结构见图3.4。

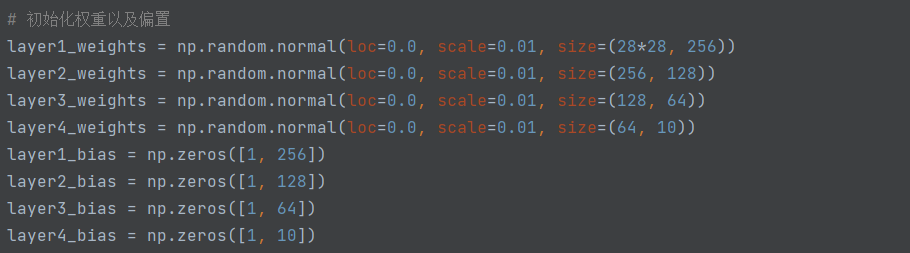


图3.3 权重与偏置初始化

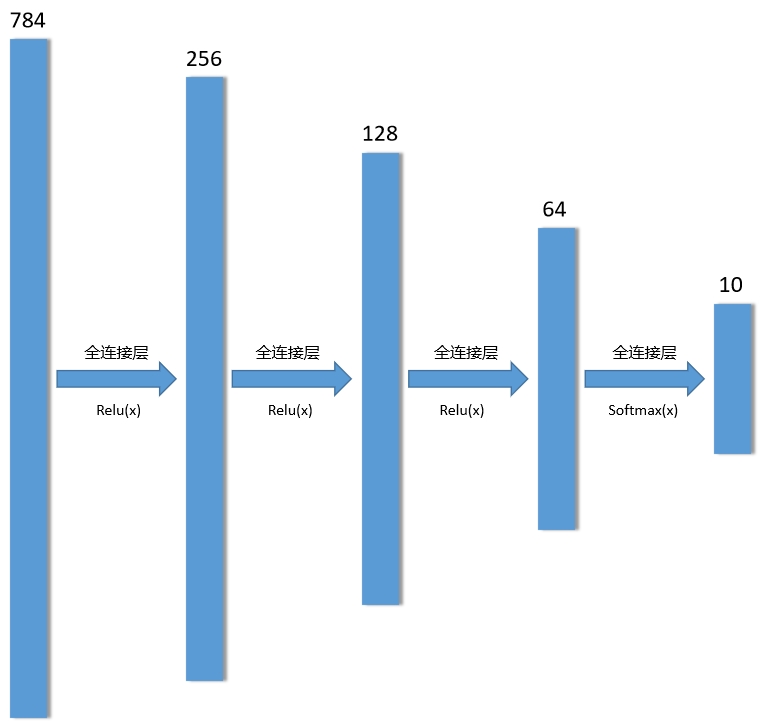


图3.4 网络结构

1. **训练过程**
2. **训练数据预处理**

在训练过程中，训练数据预处理主要分为获得单批训练图像数据与标签数据、展平训练图像数据、将标签数据转换成独热码格式三个步骤。其中单批训练图像数据与标签数据可以利用函数get\_mini\_batches输出的数据取该轮数的维度作为单批训练集；展平训练图像数据可以利用np.reshape函数将其展平格式为(50,784)的数据；将标签数据转换成独热码格式可以利用函数one\_hot，获得大小为(50,10)的独热码。训练数据预处理相关代码见图3.5所示。

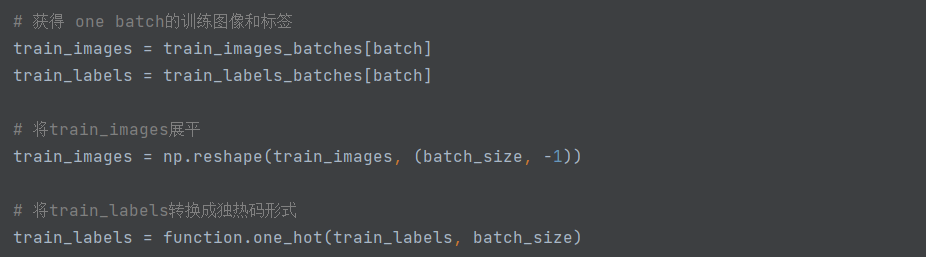


图3.5 训练数据预处理

1. **前向传播**

在训练过程中，前向传播在全连接层中体现为矩阵的运算，主要为矩阵的乘法与加法，前向传播的具体计算过程如下列公式。

(3.1)

(3.2)

(3.3)

(3.4)

(3.5)

(3.6)

(3.7)

(3.8)

根据上述公式，各个变量矩阵大小见表3.1。

表3.1 变量与变量矩阵大小

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **变量** | **矩阵大小** | **变量** | **矩阵大小** | **变量** | **矩阵大小** | **变量** | **矩阵大小** |
|  |  |  |  |  |  |  | 256 |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

前向传播相关代码如图3.6所示。

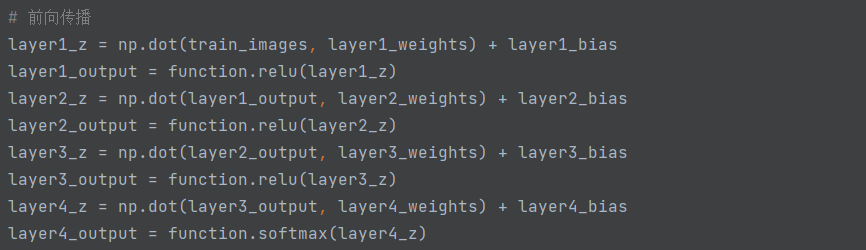


图3.6 前向传播

1. **计算损失**

在分类问题中主要采用交叉熵损失函数，因此本次实验的损失韩式也采用CE。交叉熵损失函数具有对数函数的形式，避免了梯度消失的问题，这使得模型的训练更加稳定。交叉熵损失函数公式如下：

(3.9)

其中a为网络输出标签，y为真实标签。计算损失相关代码如图3.7所示。

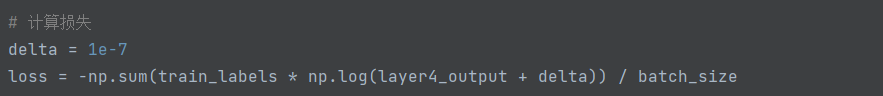


图3.7 计算损失

1. **反向传播**

在训练过程中，反向传播体现为线性函数、激活函数与损失函数的求导，根据链式法则，反向传播的具体计算过程如下列公式。

(3.10)

(3.11)

(3.12)

(3.13)

(3.14)

(3.15)

(3.16)

(3.17)

(3.18)

(3.19)

(3.20)

(3.21)

(3.22)

(3.23)

(3.24)

根据上述公式，各个变量矩阵大小见表3.2。

表3.2 变量与变量矩阵大小

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **变量** | **矩阵大小** | **变量** | **矩阵大小** | **变量** | **矩阵大小** | **变量** | **矩阵大小** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

反向传播相关代码如图3.8所示。

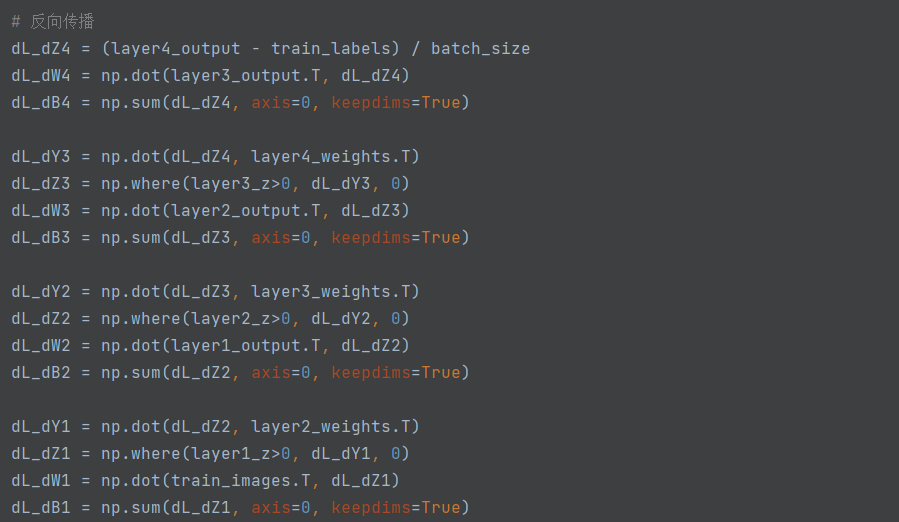


图3.8 反向传播

1. **参数更新**

权重与偏置的参数更新可根据随机梯度下降完成，利用（4）中反向传播的权重与偏置梯度完成参数更新。参数更新的具体计算过程如下列公式。

(3.25)

(3.26)

(3.27)

(3.28)

(3.29)

(3.30)

(3.31)

(3.32)

参数更新相关代码如图3.9所示。

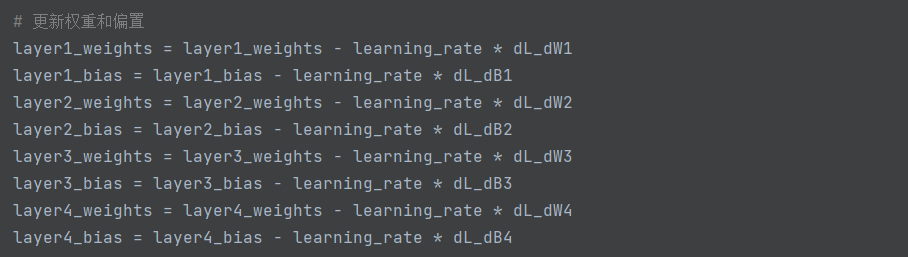


图3.9 参数更新

1. **其他**

完成网络的训练过程中，利用np.savez保存当轮次的权重和偏置，利用print函数打印每一轮的损失，相关代码如图3.10所示。

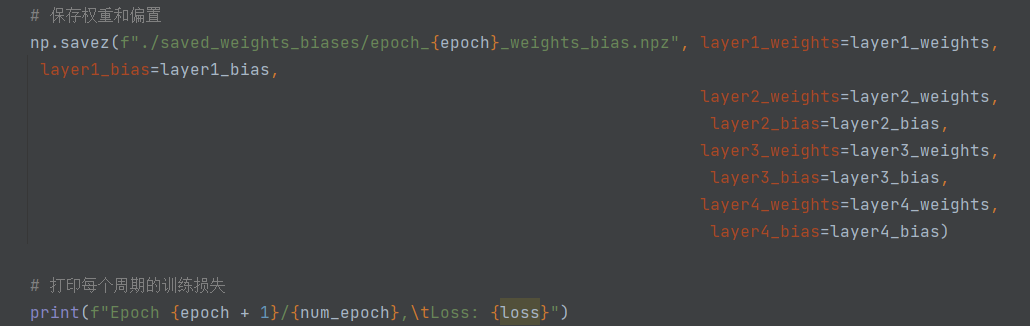


图3.10 保存参数与打印输出

1. **test.ipynb**
2. **参数定义**

主要定义两个变量，分别是单批数量、是否对数据进行打乱，具体相关参数取值如图4.1。



图4.1 测试参数变量定义与取值

1. **测试数据准备**

测试数据的准备主要分为了定义测试图像、测试标签的路径，加载测试图像，加载测试标签，获得batch大小的数据以及batch的数量，共4部分。其中加载测试图像、加载测试标签采用function.ipynb文件中函数load\_mnist\_images、函数load\_mnist\_labels完成，具体函数作用见上文第二章。获得batch大小的数据以及batch的数量采用function.ipynb文件中函数get\_mini\_batches完成。测试数据的格式为(200,50,28,28)，训练标签的格式为(200,50,1)。测试数据准备的相关代码见图4.2所示。

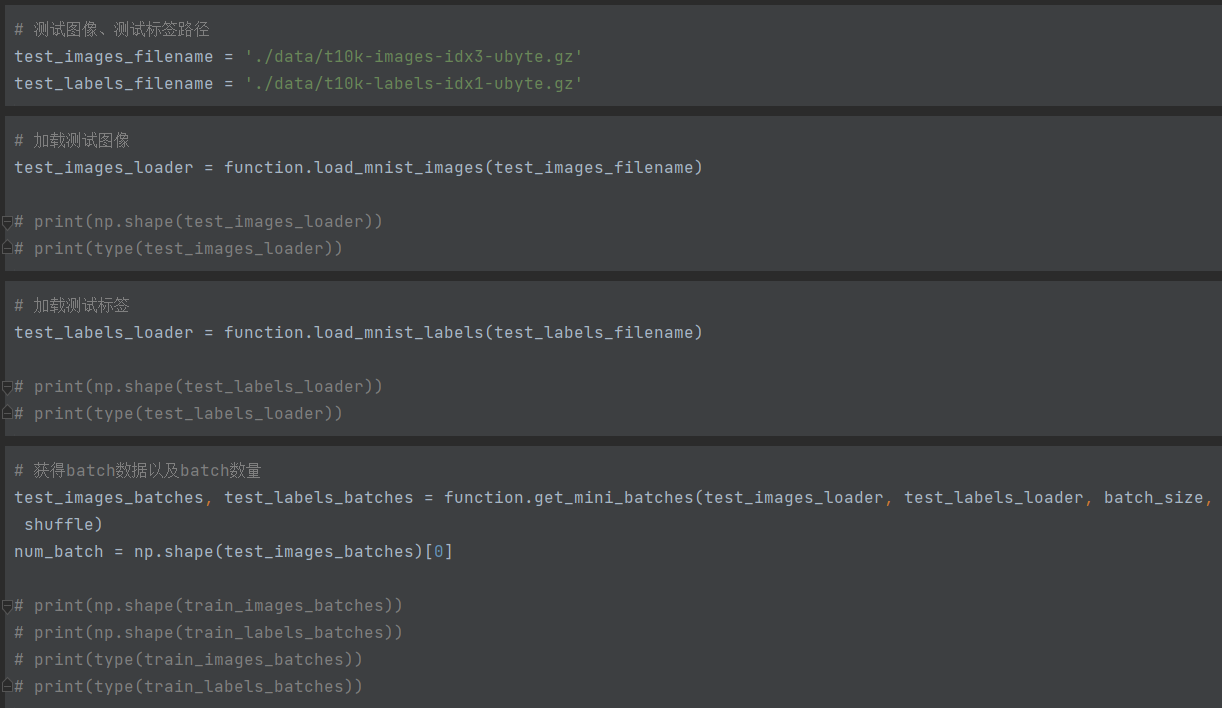


图4.2 测试数据准备

1. **加载权重与偏置**

利用np.load函数加载训练过程中保存的参数，利用字典格式分别读取各层权重与偏置。权重与偏置的加载相关代码见图4.3。

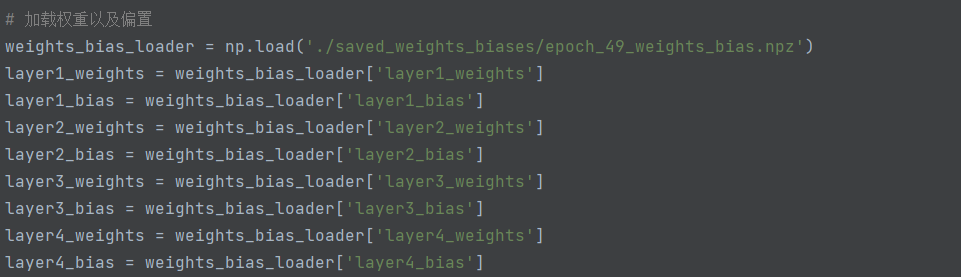


图4.3 权重与偏置的加载

1. **测试过程**
2. **测试数据预处理**

在测试过程中，测试数据预处理主要分为获得单批测试图像数据与标签数据、展平测试图像数据、将标签数据转换成独热码格式三个步骤。其中单批测试图像数据与标签数据可以利用函数get\_mini\_batches输出的数据取该轮数的维度作为单批测试集；展平测试图像数据可以利用np.reshape函数将其展平格式为(50,784)的数据；将标签数据转换成独热码格式可以利用函数one\_hot，获得大小为(50,10)的独热码。测试数据预处理相关代码见图4.4所示。

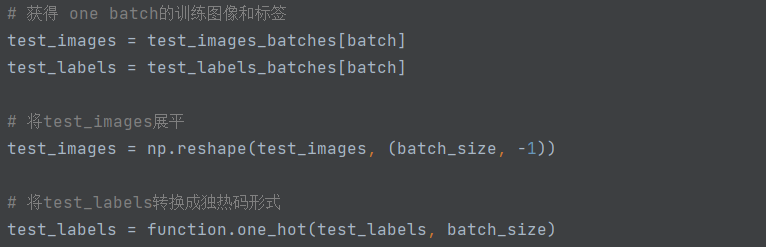


图4.4 测试数据预处理

1. **前向传播**

测试集的前向传播类似于第三章中训练过程中的前向传播，相关代码如图4.5所示。

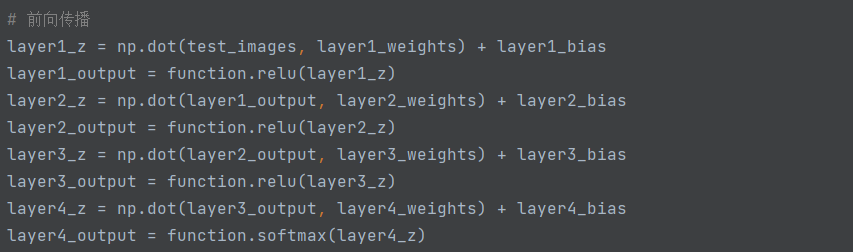


图4.5 前向传播

1. **计算并打印正确率**

针对测试集可计算每轮次预测正确的个数并计算正确率，利用print打印正确率。相关代码如图4.6所示。

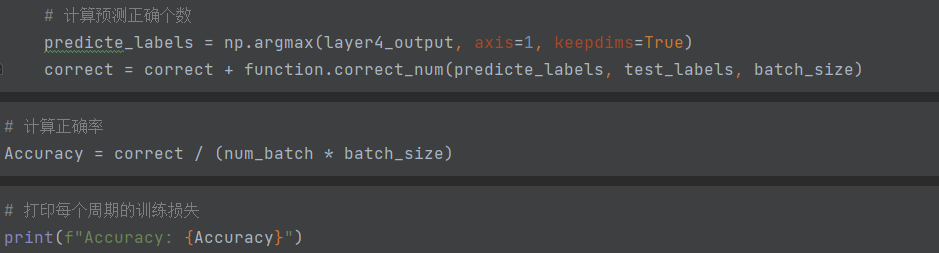


图4.5 正确率计算

1. **实验结果**

训练结果如图5.1所示，测试结果如图5.2所示。最终正确率达到97.58%。

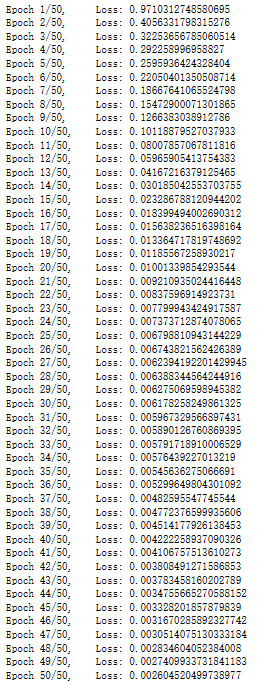


图5.1 训练结果



图5.2 测试结果