基于 LeNet 的神经网络十分类器实验报告

521030910395 卜家梓

一. 设计思路：

首先，参考$chatgpt$给出的深度学习项目文件夹分类建议，划分$Ten\ classifier\ based\ on\ LeNet$项目为$data$文件夹（存放数据集）、$model$文件夹（存放模型）和$doc$文件夹（存放文档）。

$data$文件夹下原本设计下应该有$download_data.py$、$load\_data.py$、$preprocess_data.py$三个文件，分别用于数据集下载、数据集划分为训练集测试集并打标签以及数据预处理，但是经过简化处理，只保留了$load_data.py$文件，同时具备数据集下载和划分打标签的功能。另外两个文件弃用了，但是没有删除。

$model$文件夹下存有$lenet.py$、$train.py$、$evaluate.py$三个文件，分别是模型和训练并存储模型、评估模型的代码。

doc文件夹存放说明文档，即本文档及其$word$版本。

在这些文件夹之外，我们还另外新建了一个集成它们功能的$main.py$，运行$main.py$即可得到完整结果。其实后来实现$main.py$的时候发现没有必要另外实现$train.py$和$evaluate.py$，只需三个文件就可以完整地实现这个神经网络。

二. 重要的代码实现：

为什么使用$tensorflow$而不是$pytorch$：因为参考的教程是基于$tensorflow$的，实在是不好用，再也不用了。

第一点是$load_data.py$的实现，本来我考虑使用$download_data.py$从网上下载数据集，以失败告终，所以舍弃下载数据的过程，选择直接调用$tensorflow$里的$mnist$数据集，利用$mnist.load_data()$函数划分训练集测试集并打上标签。这里引入$numpy$库的作用是为了调用其中的$expand_dims()$函数，将二维的灰度图像转换形状为$(28,28,1)$的三维数组。

第二点就是$lenet$模型的实现，我们使用$tf.keras.Sequential()$创建一个序列模型，依次加入两个“卷积层 + 池化层”组合，最后加入扁平层、全连接层，具体的参数取值除了分类个数为10以外均参考了$chatgpt$的代码建议。最后的输出层采用$softmax$激活函数，将模型的输出对应到一个概率分布。

最后是$train.py$和$evaluate.py$的实现，我们首先实现在预处理文件中应该完成的部分，利用$tf.keras.utils.to_categorical()$函数将$load_data()$得到的数据预处理为$one-hot$编码，然后调用$create_model()$函数创建模型，再调用$model.fit()$函数训练模型，这里我们只设置$epochs = 10$，因为训练时间有些太长，但是已经可以说明我们的网络可以跑起来。最后我们用$model.save()$保存.$h5$格式的模型，便于$evaluate.py$调用。其实我们已经有评估模型的环节，但是为了测试模型能否复用，还是额外设计了$evaluate.py$。

$evaluate.py$中我们先在保存的路径中$load$模型，然后使用$model.compile()$，在测试集上测试模型，最后使用$model.evaluate()$获得$loss$和$accuracy$。

只需运行$main.py$即可得到全部结果，由于参考许多不同的代码，最后代码中有诸多冗余。