电子科技大学信息与软件工程学院

**实 验 报 告**

学 号 2019091604016

姓 名 夏野

（实验） 课程名称 人工智能

实验时间 11月14日

实验教师 周帆

**电 子 科 技 大 学**

**实 验 报 告**

1. **实验名称：**

手写数字识别——CNN的应用

1. **实验学时：**

4学时

1. **实验目的：**

正式进入深度学习模块，掌握深度学习的原理、模型的搭建、框架的使用，同时对深度学习基础网络CNN有初步的了解。通过这次试验，能够独立编写出简单的深度学习训练网络，包括数据处理、模型搭建、模型训练和模型评估。

1. **实验原理：**
2. 深度学习：深度学习是学习样本数据的内在规律和表示层次，这些学习过程中获得的信息对诸如文字，图像和声音等数据的解释有很大的帮助。它的最终目标是让机器能够像人一样具有分析学习能力，能够识别文字、图像和声音等数据。 深度学习是一个复杂的机器学习算法，在语音和图像识别方面取得的效果，远远超过先前相关技术。深度学习在搜索技术，数据挖掘，机器学习，机器翻译，自然语言处理，多媒体学习，语音，推荐和个性化技术，以及其他相关领域都取得了很多成果。深度学习使机器模仿视听和思考等人类的活动，解决了很多复杂的模式识别难题，使得人工智能相关技术取得了很大进步。
3. CNN：卷积神经网络，神经网络由大量的神经元相互连接而成。每个神经元接受线性组合的输入后，最开始只是简单的线性加权，后来给每个神经元加上了非线性的激活函数，从而进行非线性变换后输出。每两个神经元之间的连接代表加权值，称之为权重（weight）。不同的权重和激活函数，则会导致神经网络不同的输出。
4. 激活函数：是神经网络中非常重要的东西，作用不亚于卷积。激活函数是为了模拟神经元的激活和抑制状态的一个组件，自然界的神经活动都是通过一部分神经元的激活，一部分神经元受到抑制而实现的，同时激活的神经元也会在一定条件抑制，抑制也会转换为激活，这种状态之间的切换，在宏观上就呈现了不同的神经活动。
5. TensorFlow：是一个使用数据流图进行数值计算的开放源代码软件库。图中的节点代表数学运算，而图中的边则代表在这些节点之间传递的多维数组（张量）。借助这种灵活的架构，您可以通过一个 API 将计算工作部署到桌面设备、服务器或移动设备中的一个或多个 CPU 或 GPU。TensorFlow 最初是由 Google Brain 团队（隶属于 Google 机器智能研究部门）中的研究人员和工程师开发的，旨在用于进行机器学习和深度神经网络研究。但该系统具有很好的通用性，还可以应用于众多其他领域。
6. **实验内容：**
7. 加载数据集

|  |
| --- |
| (x\_train, y\_train), (x\_test, y\_test) = tf.keras.datasets.mnist.load\_data() |

1. 搭建CNN模型。
   1. 优化器（optimizer）：adam
   2. 评估指标（metrics）：accuacy。
2. 训练好模型后，将其用于测试。
3. **实验器材（设备、元器件）：**

1. 硬件：mac笔记本电脑

2. 软件：macOS系统，PyCharm

3. 框架：Tensorflow

1. **实验步骤：**
2. 加载数据集：首先在keras.datasets中有mnist类，使用类中方法load\_data()方法进行加载。加载后，得到两个元组，一个是训练数据和对应的label，一个是测试数据和对应的label。其中，训练数据的大小是(60000, 28,28)，测试数据是(10000, 28,28)，因此比例是6:1；
3. 处理数据：首先，在keras中，conv2D接收的数据维度应该为3（除掉batch这一维），而我们现在的数据是二维数据，因此少了一个通道对应的维度。而对于图片而言，如果是RGB编码，那么通道数应该为3，此实验是灰度级图片，因此通道数就是1，我们就直接使用reshape方法增加一个维度为(60000, 28,28,1)即可。除此之外，为了对数据进行标准化，我选择对每一个值都除以255。而对于target，因为是一个多分类问题，我使用keras.utils中的to\_categorical()方法对target进行one-hot编码；
4. 搭建模型：tensorflow.keras模块中提供了非常容易的模型搭建方法——直接使用Sequential、functional API，我采用简单方法Sequential进行搭建。Conv2D和Maxpooling这两层重复3次，搭建起CNN，之后通过一个Flatten将数据压缩为2维，这样容易通过Dense层，最后Dense层的单元数为分类数字10，并且将激活函数设置为softmax，表明输出为概率值；
5. 模型编译与训练：使用compile函数进行模型的编译，其中需要设置优化器，选择adam；损失函数设置为categorical\_crossentropy；最后，matrics设置为acc；最后，利用fit方法训练模型，设置epoch=10，batch\_size=64；
6. 结果可视化：使用matplotlib进行结果可视化，fit函数可以使用history进行接收返回值，其中包含两个健值——acc和loss，这样可以用plt绘制acc和loss曲线。
7. **实验结果与分析（含重要数据结果分析或核心代码流程分析）**

1. 重要数据结果分析：

(1)数据可视化结果

Chart, line chart

Description automatically generated

结果分析：

成功拟合出Traning Accuracy 以及 Training Loss两个图表。顺利完成实验。

（2）代码运行结果

Text

Description automatically generated

结果分析：

模型成功读取数据文件、并进行训练，得到我们想要的数据。并显示。

2. 核心代码流程分析

（1）装载数据集代码

(train\_images, train\_labels), (test\_images, test\_labels) = mnist.load\_data()

代码分析：

在keras.datasets中有mnist类，我们可以使用该类中方法load\_data()方法进行装载数据。当成功装载数据后加载后，我们可以得到两个元组，一个是训练数据和对应的label，另一个是测试数据和对应的label。其中，训练数据的大小是(60000, 28,28)，测试数据是(10000, 28,28)，因此比例是6:1；

（2）数据处理代码

train\_images = train\_images.reshape((60000, 28, 28, 1))  
train\_images = train\_images.astype('float') / 255  
  
test\_images = test\_images.reshape((10000, 28, 28, 1))  
test\_images = test\_images.astype('float') / 255  
  
train\_labels = to\_categorical(train\_labels)  
test\_labels = to\_categorical(test\_labels)

代码分析：

在keras类中，conv2D接收的数据维度应该为3（除掉batch这一维），而我们现在的数据是二维数据，因此少了一个通道对应的维度。而对于图片而言，如果是RGB编码，那么通道数应该为3，此实验是灰度级图片，因此通道数就是1，所以我们在这里直接使用我们train\_images中的reshape()方法增加一个维度为(60000, 28,28,1)就可以了。除此之外，为了对数据进行标准化，我们需要把每一个值都除以255。而对于target变量而言，因为我们现在处理的问题是一个多分类问题，所以我在这里使用keras.utils中的to\_categorical()方法对target进行one-hot编码。

（3）模型搭建代码

model = keras.models.Sequential([  
 keras.layers.Conv2D(32, (3, 3), activation='relu',  
 input\_shape=(28, 28, 1)),  
 keras.layers.MaxPooling2D((2, 2)),  
 keras.layers.Conv2D(64, (3, 3), activation='relu'),  
 keras.layers.MaxPooling2D((2, 2)),  
 keras.layers.Conv2D(64, (3, 3), activation='relu'),  
 keras.layers.MaxPooling2D((2, 2)),  
 keras.layers.Flatten(),  
 keras.layers.Dense(64, activation='relu'),  
 keras.layers.Dense(10, activation='softmax')  
])

代码分析：

在tensorflow.keras提供了非常容易的模型搭建方法，我们可以直接使用Sequential()方法来搭建。在Conv2D和Maxpooling这两层我们需要重复3次来搭建起CNN，然后我们通过一个Flatten将数据压缩为2维，这样的话，我们就更加容易通过Dense层；最后，在Dense层的单元数为分类数字10，并且我们将激活函数设置为softmax，表明输出为概率值。

（4）编译与模型训练代码

model.compile(optimizer=keras.optimizers.Adam(),  
 loss='categorical\_crossentropy',  
 metrics=['accuracy'])  
model.summary()  
history = model.fit(train\_images, train\_labels, epochs=10, batch\_size=64)

代码分析：

这里我们使用model类中的compile函数进行模型的编译，其中需要设置优化器，我们将optimizer=keras.optimizers.Adam()作为参数传入来选择adam；将`loss=categorical\_crossentropy`作为参数传入来损失函数设置为categorical\_crossentropy；最后，我们将matrics设置为acc；调用summery()方法来训练；最后，利用fit方法训练模型，设置epoch=10，batch\_size=64来确定训练次数。

（5）数据可视化代码

accuracy = history.history['accuracy']  
loss = history.history['loss']  
  
plt.subplot(1, 2, 1)  
plt.plot(accuracy, label='Training Loss')  
plt.title('Training Accuracy')  
plt.legend()  
  
plt.subplot(1, 2, 2)  
plt.plot(loss, label='Training Loss')  
plt.title('Traiining Loss')  
plt.legend()  
plt.show()

代码分析：

我们这里还是使用matplotlib类中的方法来进行结果可视化，fit函数可以使用history进行接收返回值，其中包含两个健值——acc和loss，这样可以用plt绘制acc和loss曲线。同时设置一些简单的图表上的内容，就可以完成线束数据的样子。

1. **总结及心得体会：**

通过学习和完成实验四，我正式进入深度学习模块，初步掌握深度学习的原理、模型的搭建、框架的使用，同时对深度学习基础网络CNN有初步的了解。通过这次试验，我能够能够独立编写出简单的深度学习训练网络，包括数据处理、模型搭建、模型训练和模型评估再到最后实验的可视化。每一个步骤都是十分重要缺一不可的。

在实验过程中我对于python语言更加熟悉，对于python的debug能力也有了一定提升，受益匪浅。

同时对于epoch的选择有了更深的理解，如果epoch过多会造成过拟合的效果，不能达到我们的预期。