人工智能

实验报告

实验名称：基于前向神经网络的英文字符识别

院系：计算机学院·网络空间安全学院

班级：20级计算机科学与技术1班

学号：202005565405

姓名：唐杰

指导老师：肖芬

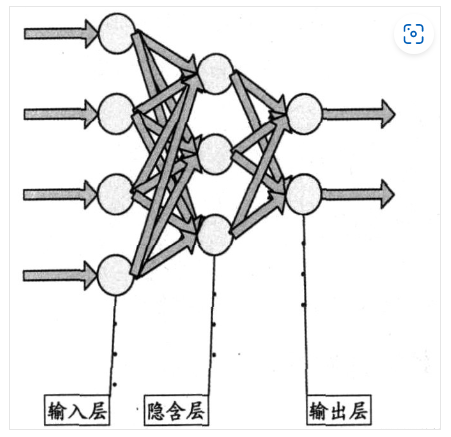
日期：2023年4月23日

**一、实验介绍**

前向网络是对非线性可微分函数进行权值训练的多层前向网络。在人工神经网络的实际应用中大多数的模型都采用了前向网络或其变化形式。采用前向算法的多层前馈网络, 是迄今为止应用最广泛的神经网络, 具有很强的非线性逼近能力以及自适应、自学习能力。本次实验基于前向网络，对英文字符进行识别。

**二、实验分析**

前向网络是目前研究最多的网络形式之一,它包含输入层、隐层以及输出层, 其中隐层可以为一层或者多层,如下图所示：

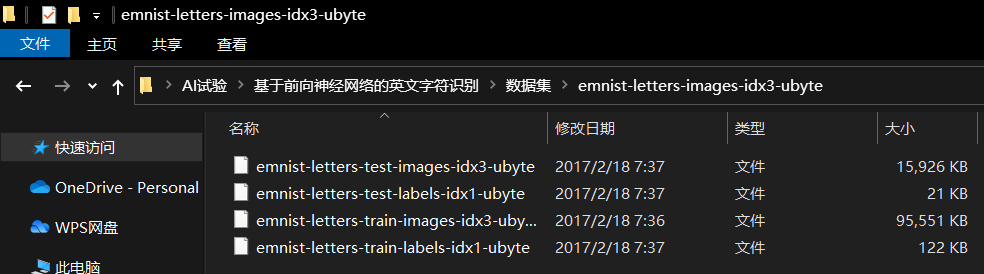


前向网络在本质上是一种输入到输出的映射，它能够学习大量的输入与输出之间的映射关系，而不需要任何输入和输出之间的精确的数学表达式，只要用已知的模式对BP网络加以训练，网络就具有输入输出对之间的映射能力。

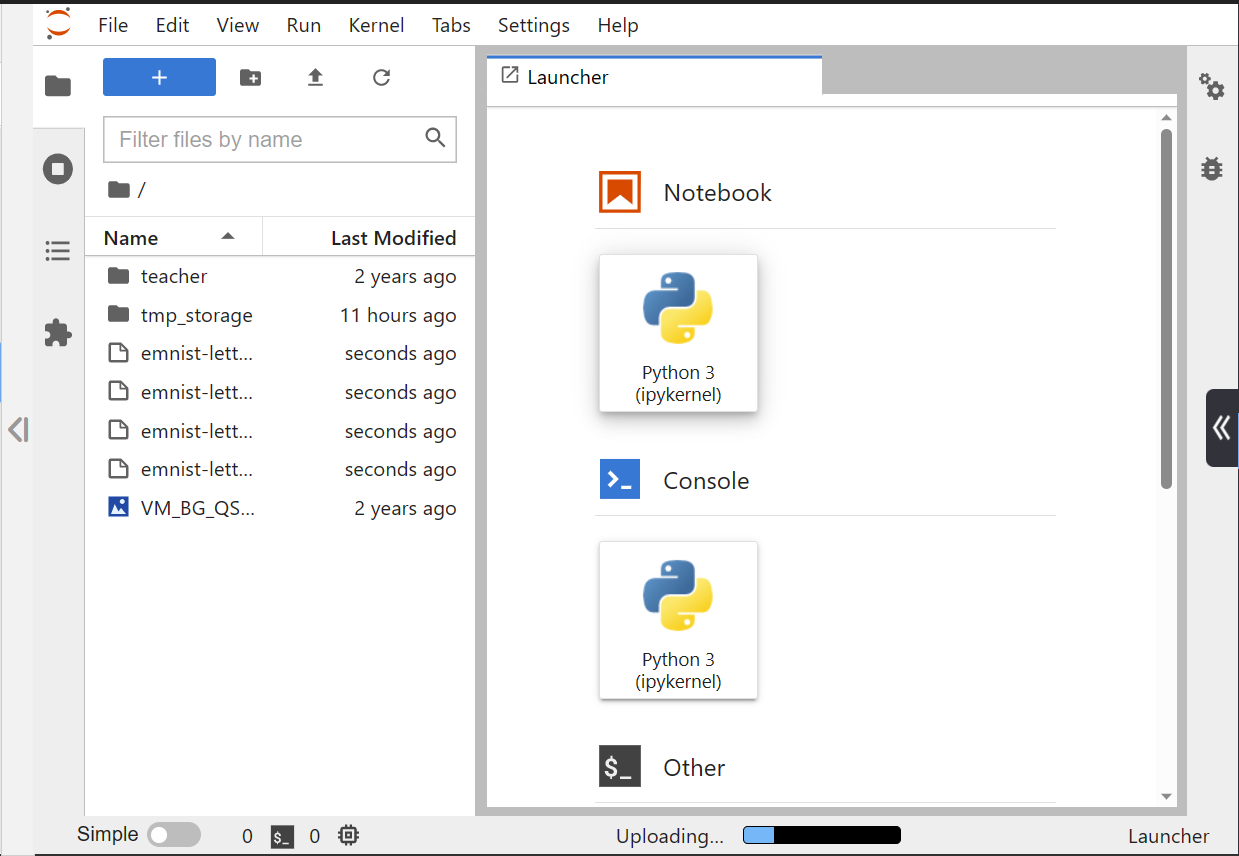
前向算法由两部分组成：信息的正向传递与误差的反向传播。在正向传递过程中，输入信息从输入层经隐含层逐层计算传向输出层，每一层神经元的状态只影响下一层神经元的状态。如果在输出层没有得到期望的输出，则计算输出层的误差变化值，然后反向传播，通过网络将误差信号沿原来的连接通路反传回来，修改各层神经元的权值直至达到期望目标。

**三、数据预处理**

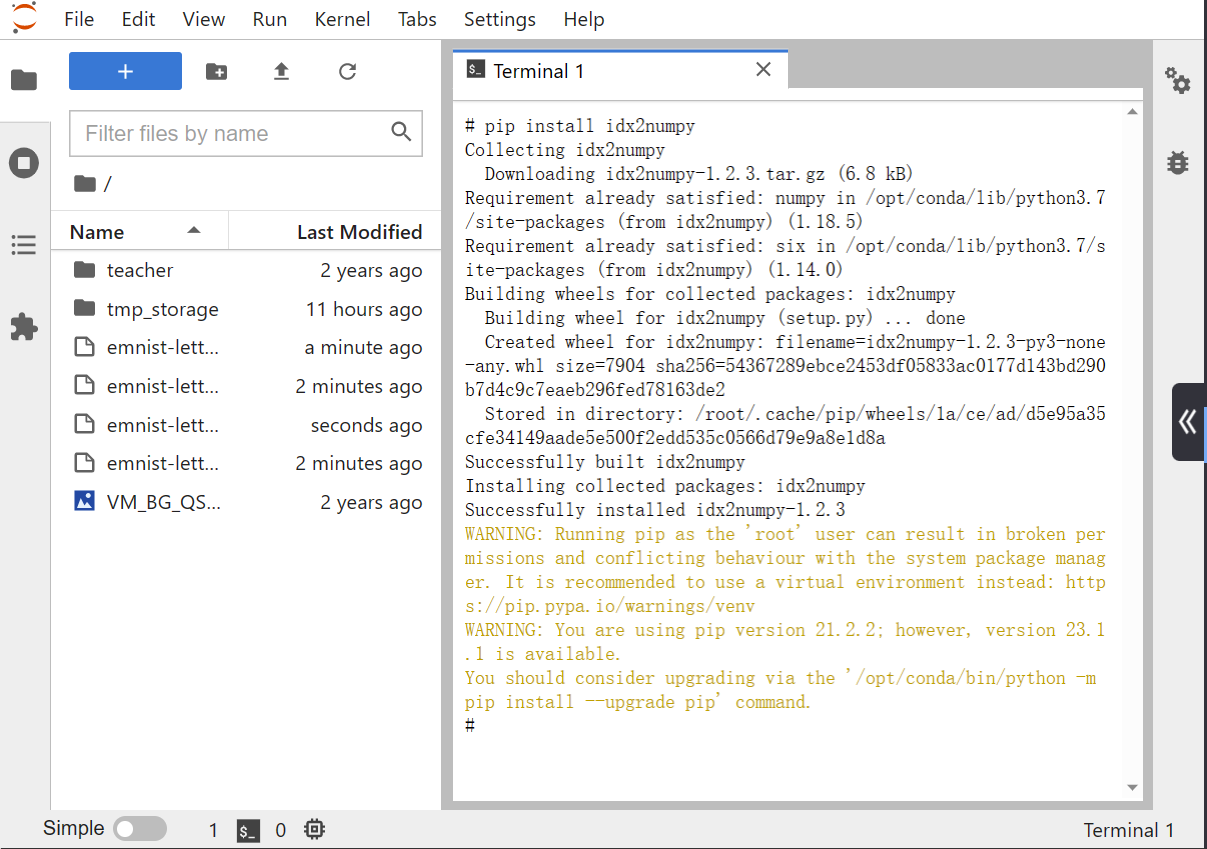
1.下载、解压数据集附件



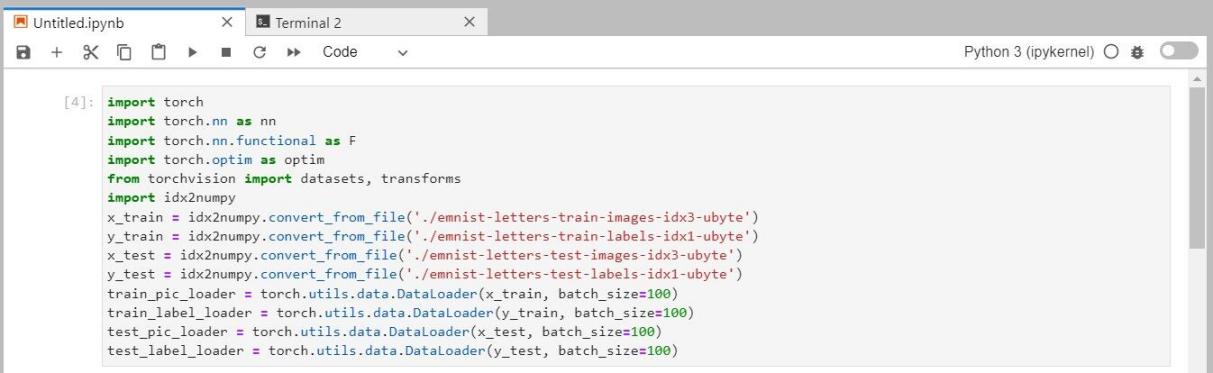
2.第二步上传数据集：将刚解压的文件都上传。



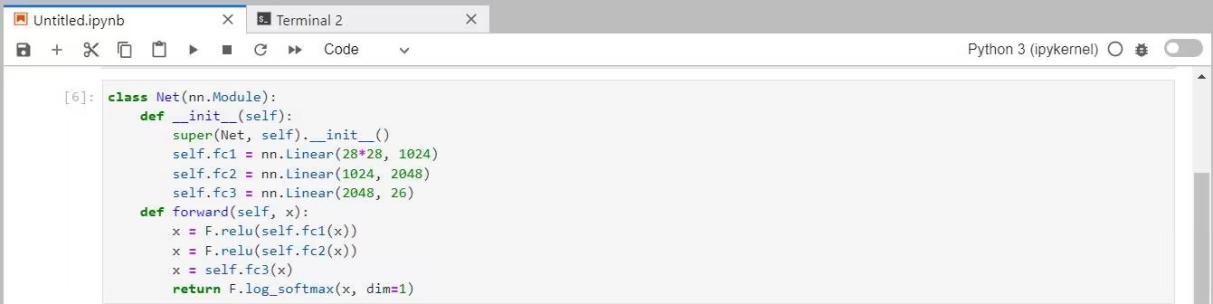
3.安装idx2numpy用来读取数据。



4.notebook，读取数据, 进行数据预处理

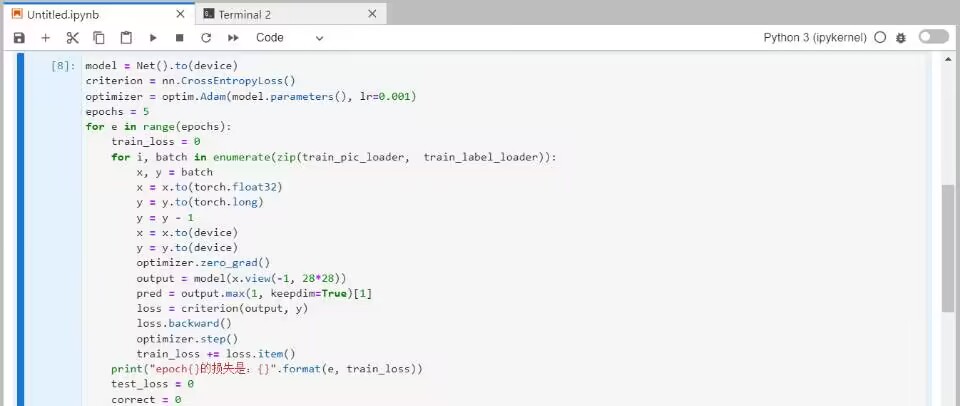


**四、编写前向网络**

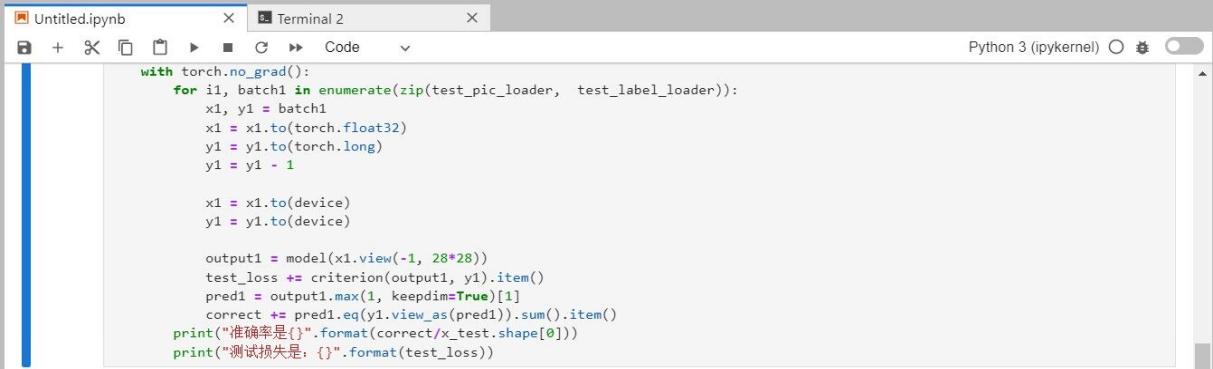
根据实际输入的大小和输出的大小，编写设计对应的前向网络。

**五、训练网络。**

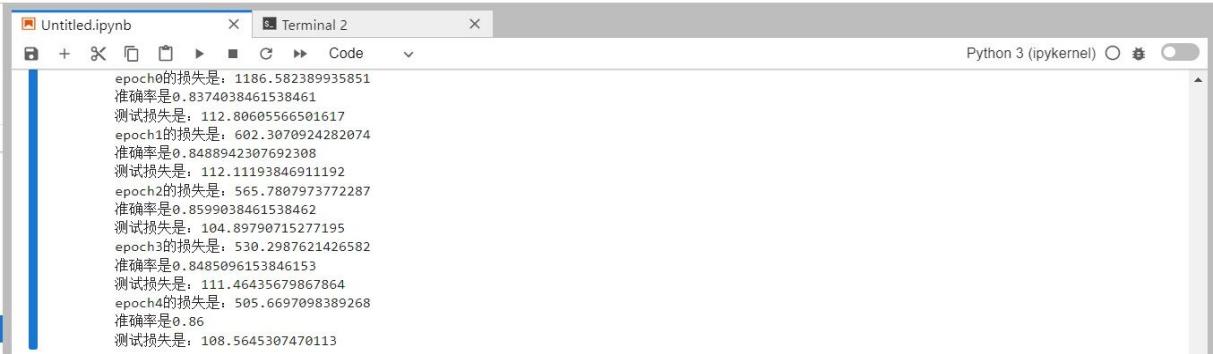
调整网络参数，对网络进行训练。



**六、测试网络**

****

**七、结果**

****

**八、实验心得**

1.**对于利用神经网络进行模型学习分析的流程有了大致了解。**由于此前对字符识别以及前向神经网络了解不多，进行相关资料查找，记录如下:

（1）**字符识别**是指将手写或打印的字符转化为可识别的文本形式的过程。

深度学习技术，尤其是卷积神经网络（CNN），在字符识别领域取得了重大突破。深度学习方法可以自动从数据中学习特征，并在大规模训练集上进行端到端的训练。通过使用大量标注的字符图像数据进行训练，深度学习模型可以准确地识别各种不同的字符。

字符识别技术在许多应用中发挥着重要作用，例如银行支票处理、身份证识别、自动化数据录入和文档数字化等。它大大提高了工作效率并减少了人工错误。

（2）**前向神经网络**（Feedforward Neural Network），也称为前馈神经网络。它是最基本和常见的神经网络类型之一。

前向神经网络的信息流是单向的，从输入层经过一系列隐藏层传递到输出层，没有反馈连接。每个神经元接收来自前一层的输入，并将其与相应的权重进行加权求和，然后通过激活函数进行非线性变换。这样的过程一直持续到达输出层，输出层神经元产生最终的结果。

前向神经网络的训练过程通常采用反向传播算法（Backpropagation），通过最小化损失函数来调整网络中的权重和偏差，以使网络的输出尽可能接近预期输出。反向传播算法使用梯度下降的方法来更新网络参数，使其逐渐优化。

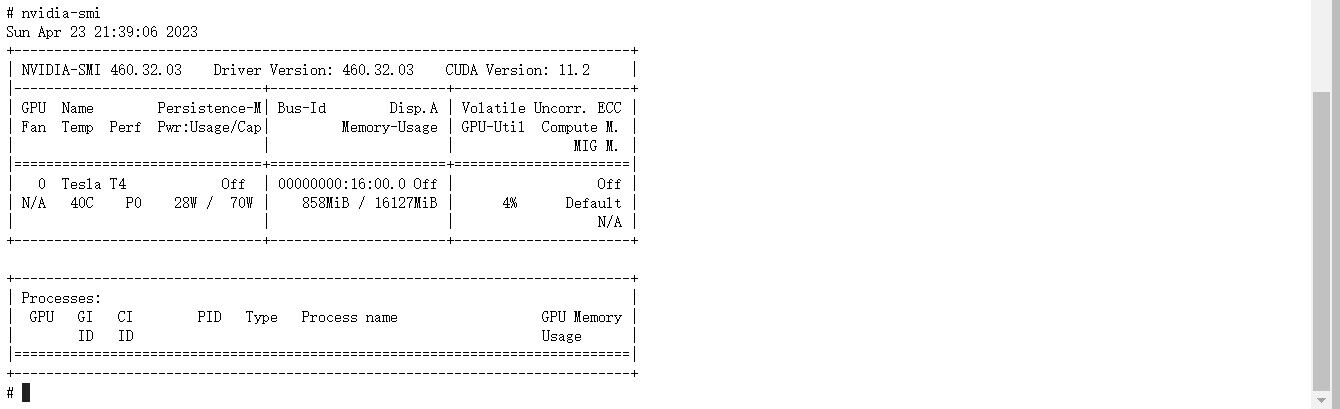
尽管前向神经网络没有显式的记忆能力，但由于其简单性和高效性，它仍然是许多机器学习和深度学习任务中的重要基础模型。

**2.对python的代码组织形式有了初步了解**

在实验过程中，出现了”IndentationError: expected an indented block”的报错。不同于其他语言例如java使用花括号{}来组织代码，python语言是用缩进块来组织代码(相同的缩进代表同一级别)。

**3.了解如何调用GPU以及GPU与CPU的区别。**





**（1）并行计算能力**： GPU（图形处理器）具有大量的并行处理单元，可以同时执行多个任务。相比之下，CPU（中央处理器）的核心数量较少，但每个核心的处理能力较强。这使得GPU在处理大规模数据和并行计算任务时通常比CPU更为高效。

（2）**适用场景**： GPU主要用于并行计算任务，如图形渲染、科学计算、深度学习等需要处理大规模数据和执行大量并行操作的应用。而CPU则更适合于处理顺序计算任务和一般用途的计算，如操作系统运行、文本处理、编译等。

（3）**编程模型**： GPU编程通常采用并行编程模型，如CUDA（Compute Unified Device Architecture）或OpenCL（Open Computing Language）。这些编程模型要求开发人员显式地编写并行代码，以充分利用GPU的并行计算能力。而CPU编程更常见的是使用通用编程语言（如C++、Python）和多线程。