

重庆邮电大学

自动化学院学生实验报告

课程名称: 多传感器融合感知技术

实验内容: 手写数字识别

专业班级: 08052102

姓 名: 王忠全

学 号: 2021212981

成 绩:

指导教师 岑汝平

学年学期: 2023 - 2024 学年 ☒春季 ☐秋季 学期

重庆邮电大学自动化学院制

实验名称	基于百度飞桨 PaddlePaddle 的手写数字识别		
地点	C304	时间	第七周 9-12 节
指导教师	岑汝平	成绩	

一、实验目的

实验目的：通过手写数字识别任务，了解深度学习中常用的卷积神经网络模型 AlexNet 的原理和结构，并掌握使用百度 AI Studio 平台进行模型训练和推理的基本步骤。同时，通过对比实际数据运行结果和理论预期结果，帮助学生深入理解深度学习模型的优化和调参方法。

1. 了解 AlexNet 的网络结构和原理，包括卷积层、池化层、全连接层等组成部分；
2. 在百度 AI Studio 平台上配置环境、上传数据集、定义模型、进行训练和推理；
3. 运行测试数据集并查看实际结果，分析模型性能和效果；
4. 总结遇到的问题和心得体会，深入思考深度学习模型的优化和调参方法。

通过完成本实验，获得对深度学习中常用卷积神经网络模型 AlexNet 的深入理解，并掌握在百度 AI Studio 平台上进行模型训练和推理的基本技能。

二、实验测验所需仪器设备和软件



PaddlePaddle 2.4.0

环境：Python 3.0

CPU 2 Cores

RAM 8GB

Disk 100GB

三、实验题目

1. Alexnet 网络原理
2. 百度平台配置步骤
3. 数据集运行结果
4. 实际数据运行结果
5. 总结（遇到的问题+心得体会）

【接下来对每一个问题进行分析与设计】

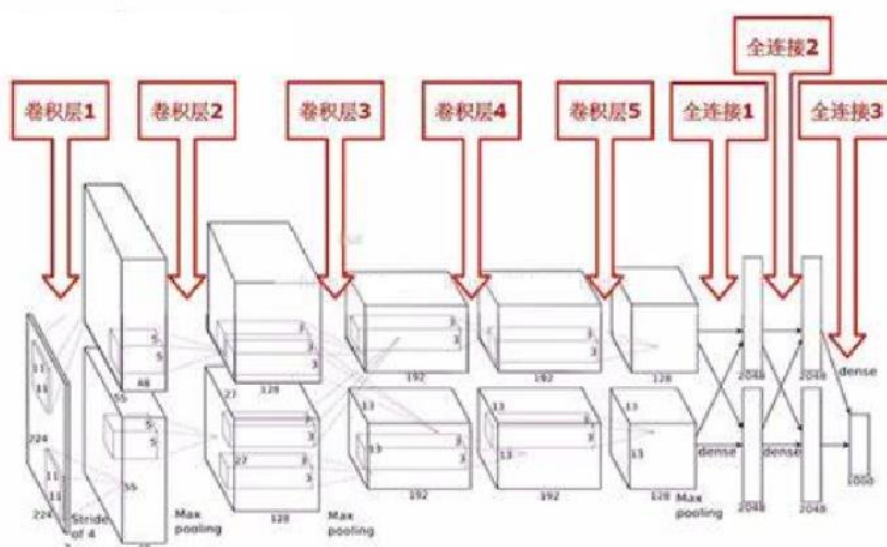
1. 【Alexnet 网络原理】

AlexNet 跟 LeNet-5 类似也是一个用于图像识别的卷积神经网络，但其网络结构更加复杂，参数更多。

AlexNet 的作者是多伦多大学的 Alex Krizhevsky 等人。Alex Krizhevsky 是 Hinton 的学生。网上流行说 Hinton、LeCun 和 Bengio 是神经网络领域三巨头，LeCun 是 LeNet5 的作者 (Yann LeCun)。

AlexNet 整体的网络结构包括：

- 1 个输入层 (input layer)
- 5 个卷积层 (C1、C2、C3、C4、C5)
- 2 个全连接层 (FC6、FC7)
- 1 个输出层 (output layer)。



1. 输入层 (Input layer)

AlexNet 的输入图像尺寸是 $224 \times 224 \times 3$ 。但是实际图像尺寸为 $227 \times 227 \times 3$ 。据说 224×224 可能是写 paper 时候的手误或是后来对网络又做了调整。

2. 卷积层 (C1)

该层的处理流程是：卷积-->ReLU-->局部响应归一化 (LRN) -->池化

对于卷积来说：

卷积运算

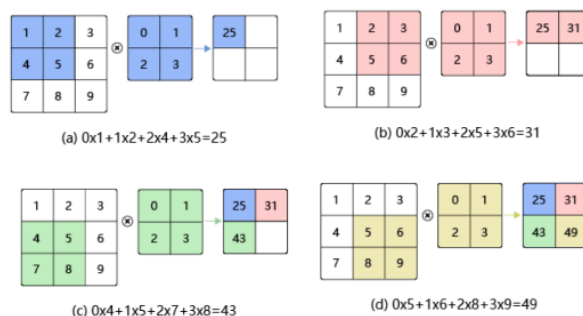
卷积 (Convolution) 运算主要涉及以下内容：

填充 (padding)

步幅 (stride)

感受野 (Receptive Field)

多输入通道、多输出通道和批量操作



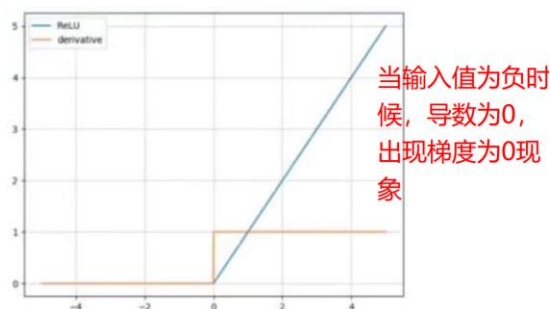
输入是 $227 \times 227 \times 3$ ，使用 96 个 $11 \times 11 \times 3$ 的卷积核进行卷积，padding=0，stride=4，根据公式： $(\text{input_size} + 2 * \text{padding} - \text{kernel_size}) / \text{stride} + 1 = (227 + 2 * 0 - 11) / 4 + 1 = 55$ ，得到输出是 $55 \times 55 \times 96$ 。

对 ReLU 激活函数来说：

c) ReLU函数

ReLU函数是目前神经网络里常用的激活函数，由于ReLU函数是线性特点使其收敛速度比 Sigmoid、Tanh更快，而且没有梯度饱和的情况出现。计算更加高效，相比于Sigmoid、Tanh函数，只需要一个阈值就可以得到激活值，不需要对输入归一化来防止达到饱和

$$\text{Relu} = \max(0, x)$$
$$g'(z) = \begin{cases} 1 & \text{if } z > 0 \\ \text{undefined} & \text{if } z = 0 \\ 0 & \text{if } z < 0 \end{cases}$$



对于局部响应归一化来说：

局部响应归一化层简称 LRN，是在深度学习中提高准确度的技术方法。一般是在激活、池化后进行。LRN 对局部神经元的活动创建竞争机制，使得其中响应比较大的值变得相对更大，并抑制其他反馈较小的神经元，增强了模型的泛化能力。

其公式为：

$$b_{x,y}^i = a_{x,y}^i / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^j)^2 \right)^\beta$$

对于池化来说：

池化 (Pooling)

池化是使用某一位置的**相邻输出的总体统计特征**代替网络在该位置的输出，其好处是当输入数据做出少量平移时，经过池化函数后的大多数输出还能保持不变。

平均池化：这里使用大小为 2×2 的池化窗口，每次移动的步幅为2，对池化窗口覆盖区域内的像素取平均值，得到相应的输出特征图的像素值。

最大池化：对池化窗口覆盖区域内的像素取最大值，得到输出特征图的像素值。当池化窗口在图片上滑动时，会得到整张输出特征图。

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

3.5	5.5
11.5	13.5

平均值池化

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

6	8
14	16

最大值池化

使用 3×3 , stride=2 的池化单元进行最大池化操作 (max pooling)。这里使用的是重叠池化，即 stride 小于池化单元的边长。根据公式： $(55+2*0-3)/2+1=27$ ，每组得到的输出为 $27 \times 27 \times 48$ 。

3. 卷积层 (C2、C3、C4、C5)

这些个层的处理都同 C1 方法一样，处理流程是：卷积-->ReLU-->局部响应归一化 (LRN) -->池化

4. 全连接层 (FC6、FC7)

该层的流程为：（卷积）全连接 --> ReLU --> Dropout （卷积）

全连接：输入为 $6 \times 6 \times 256$ ，使用 4096 个 $6 \times 6 \times 256$ 的卷积核进行卷积，由于卷积核尺寸与输入的尺寸完全相同，即卷积核中的每个系数只与输入尺寸的一个像素值相乘——对应，根据公式： $(input_size + 2 * padding - kernel_size) / stride + 1 = (6 + 2 * 0 - 6) / 1 + 1 = 1$ ，得到输出是 $1 \times 1 \times 4096$ 。既有 4096 个神经元，该层被称为全连接层。

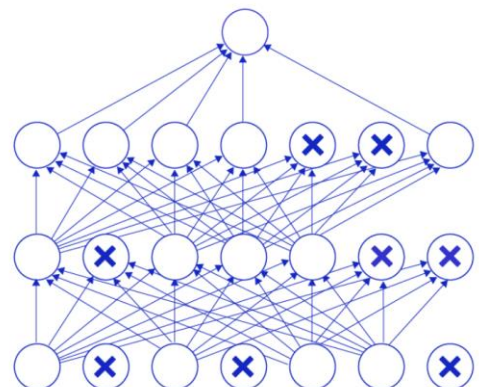
ReLU：这 4096 个神经元的运算结果通过 ReLU 激活函数中。

Dropout：随机的断开全连接层某些神经元的连接，通过不激活某些神经元的方式防止过拟合。4096 个神经元也被均分到两块 GPU 上进行运算。

■ 丢弃法 (Dropout)

丢弃法 (Dropout) 是深度学习中一种常用的**抑制过拟合的方法**，其做法是在神经网络学习过程中，随机删除一部分神经元。训练时，随机选出一部分神经元，将其输出设置为 0，这些神经元将不对外传递信号。

图16 是Dropout示意图，左边是完整的神经网络，右边是应用了Dropout之后的网络结构。应用Dropout之后，会将标了×的神经元从网络中删除，让它们不向后面的层传递信号。在学习过程中，丢弃哪些神经元是随机决定，因此模型不会过度依赖某些神经元，能一定程度上抑制过拟合。



5. 输出层 (output layer)

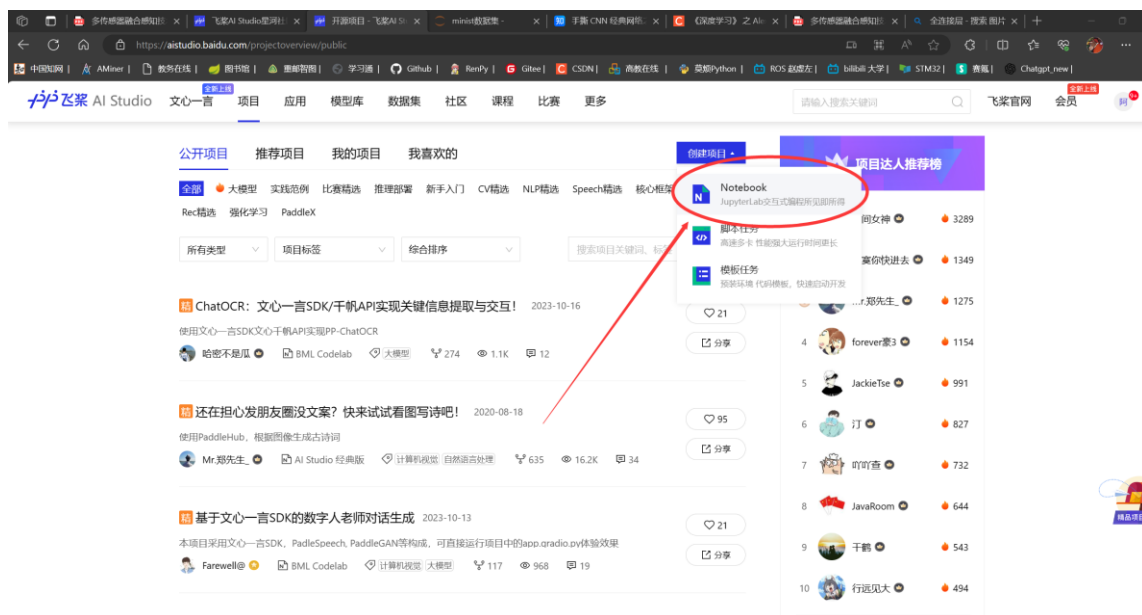
该层的流程为：（卷积）全连接 --> Softmax

全连接：输入为 4096 个神经元，输出是 1000 个神经元。这 1000 个神经元即对应 1000 个检测类别。

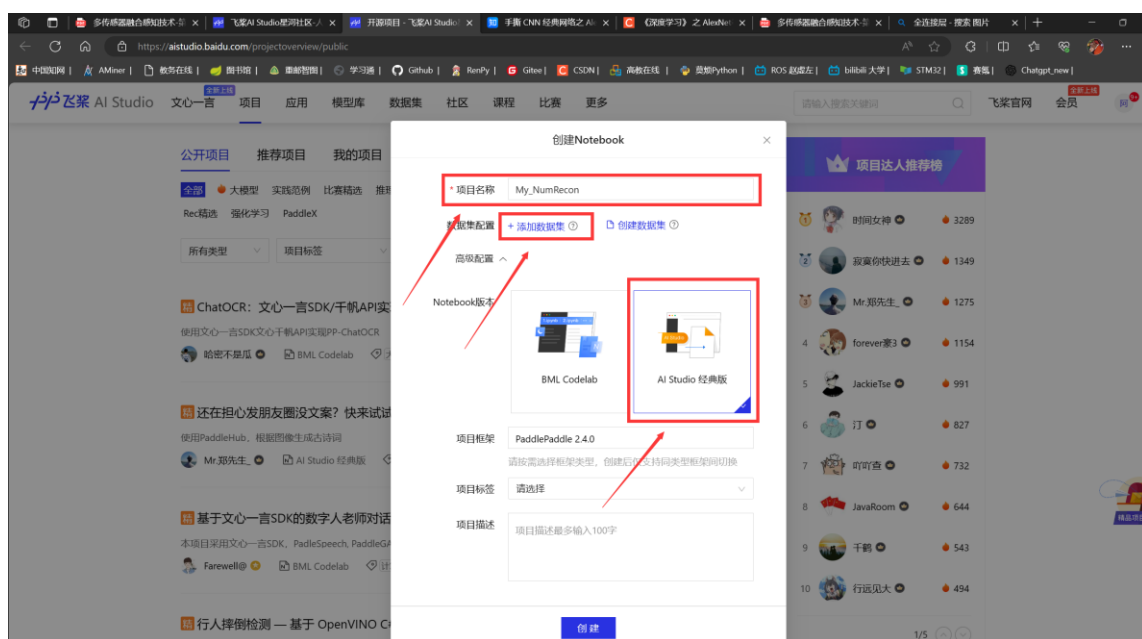
Softmax：这 1000 个神经元的运算结果通过 Softmax 函数中，输出 1000 个类别对应的预测概率值。

2. 【百度平台配置步骤】

<https://aistudio.baidu.com/projectoverview/public> 【百度飞桨】



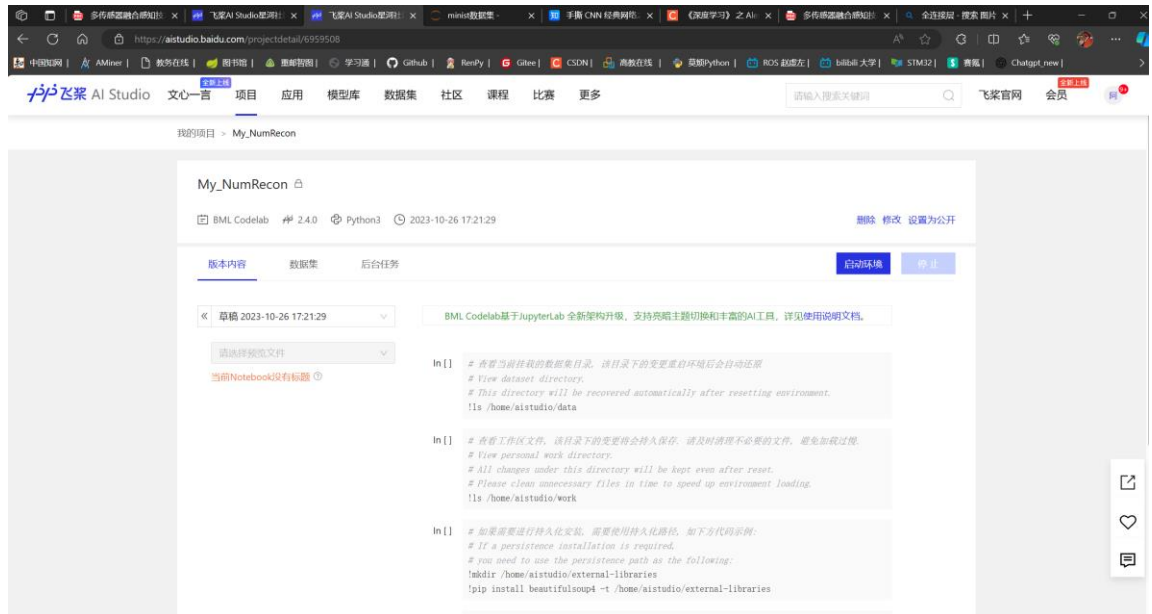
点击创建自己的项目



之后给这个项目取个好听的名字，然后添加数据集，搜索官方的MINIST数据集，选择 Notebook 经典版就好啦。



打开后，就是这样的界面：



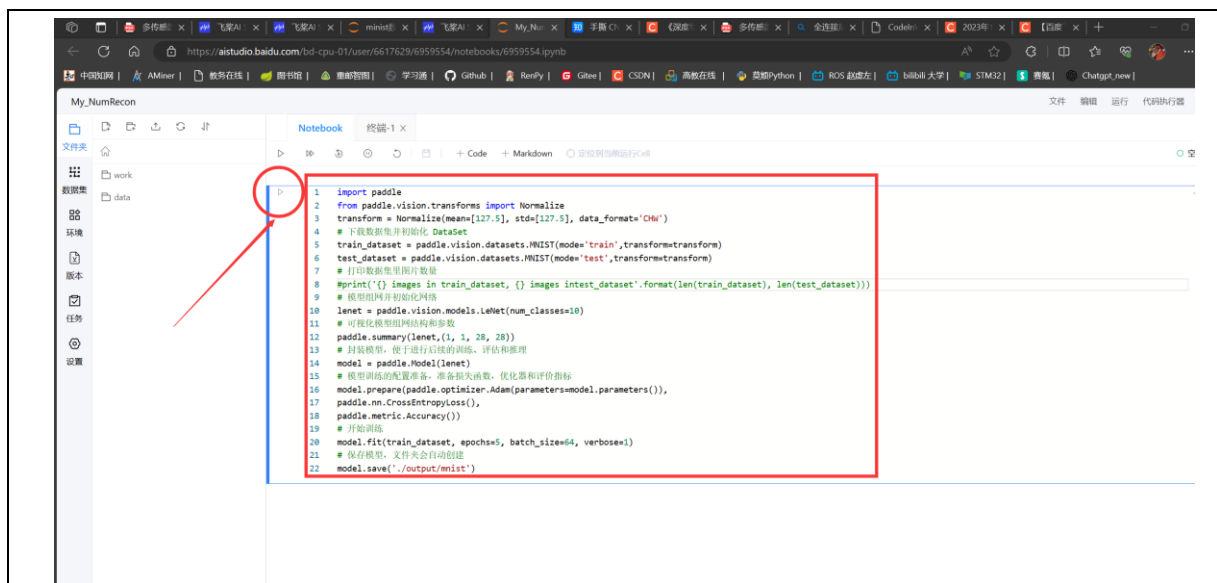
点击左上角的启动环境，选择基础版的环境就行。



3. 【数据集运行结果】

接下来就需要训练模型，通过以下的代码，在百度飞桨云平台，我们可以云处理数据集 MINIST，训练网络，之后将模型的参数保存在指定目录下。

点击左上角的三角符号运行。



代码：

```
1 import paddle
2 from paddle.vision.transforms import Normalize
3 transform = Normalize(mean=[127.5], std=[127.5], data_format='CHW')
4 # 下载数据集并初始化 DataSet
5 train_dataset = paddle.vision.datasets.MNIST(mode='train', transform=transform)
6 test_dataset = paddle.vision.datasets.MNIST(mode='test', transform=transform)
7 # 打印数据集里图片数量
8 #print('{} images in train_dataset, {} images in test_dataset'.format(len(train_dataset), len(test_dataset)))
9 # 模型组网并初始化网络
10 lenet = paddle.vision.models.Lenet(num_classes=10)
11 # 可视化模型组网结构和参数
12 paddle.summary(lenet, (1, 1, 28, 28))
13 # 封装模型，便于进行后续的训练、评估和推理
14 model = paddle.Model(lenet)
15 # 模型训练的准备工作，准备损失函数，优化器和评价指标
16 model.prepare(paddle.optimizer.Adam(parameters=model.parameters()),
17               paddle.nn.CrossEntropyLoss(),
18               paddle.metric.Accuracy())
19 # 开始训练
20 model.fit(train_dataset, epochs=5, batch_size=64, verbose=1)
21 # 保存模型，文件夹会自动创建
22 model.save('./output/mnist')
```

点击运行后，界面下方输出实时训练信息（大约需要两分钟），训练结束后，可以看到训练后模型参数保存在目录下：output

运行时长: 2分04秒41毫秒 结束时间: 2023-10-26 17:41:55

Download finished

Layer (type)	Input Shape	Output Shape	Param #
Conv2D-1	[1, 1, 28, 28]	[1, 6, 28, 28]	60
ReLU-1	[1, 6, 28, 28]	[1, 6, 28, 28]	0
MaxPool2D-1	[1, 6, 28, 28]	[1, 6, 14, 14]	0
Conv2D-2	[1, 6, 14, 14]	[1, 16, 10, 10]	2,416
ReLU-2	[1, 16, 10, 10]	[1, 16, 10, 10]	0
MaxPool2D-2	[1, 16, 10, 10]	[1, 16, 5, 5]	0
Linear-1	[1, 400]	[1, 120]	48,120
Linear-2	[1, 120]	[1, 84]	10,164
Linear-3	[1, 84]	[1, 10]	850

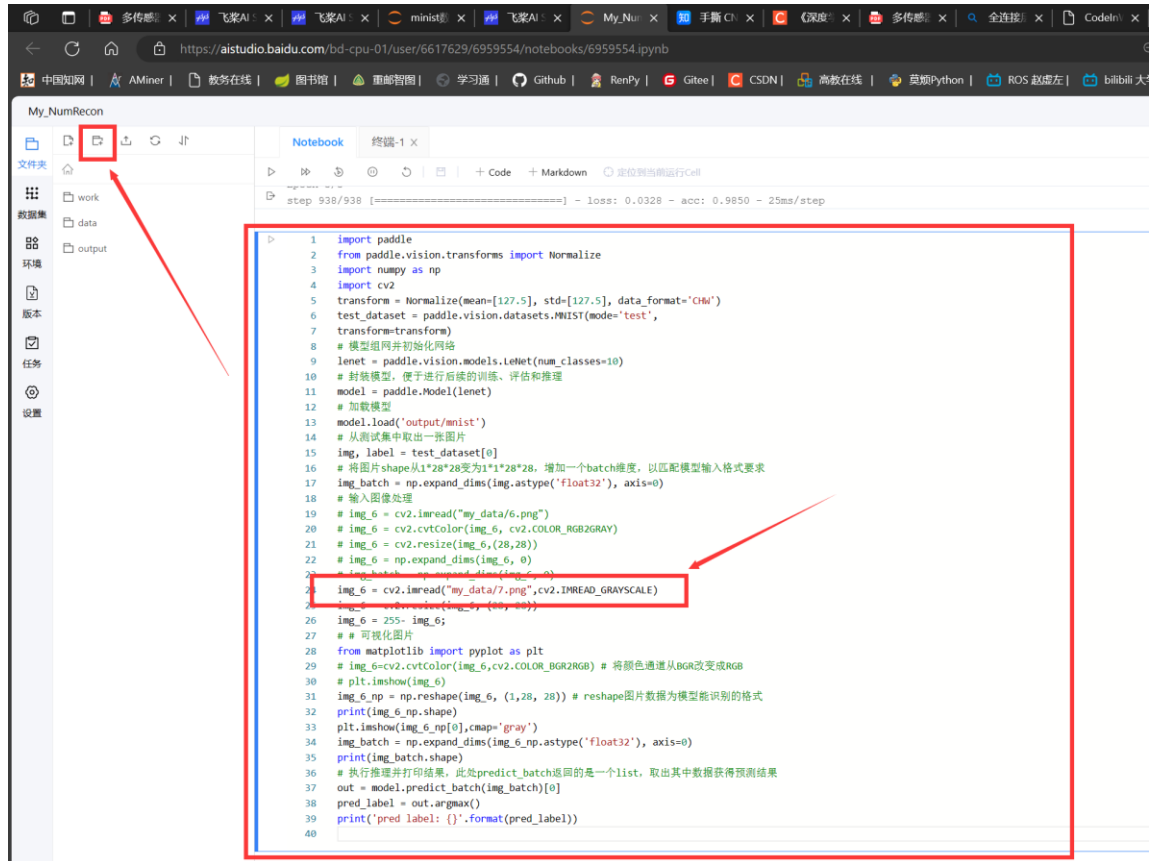
Total params: 61,610
Trainable params: 61,610
Non-trainable params: 0

Input size (MB): 0.00
Forward/backward pass size (MB): 0.11
Params size (MB): 0.24
Estimated Total Size (MB): 0.35

The loss value printed in the log is the current step, and the metric is the average value of previous steps.

Epoch 1/5
step 938/938 [-----] - loss: 0.0339 - acc: 0.9286 - 25ms/step
Epoch 2/5
step 938/938 [-----] - loss: 0.1799 - acc: 0.9757 - 25ms/step
Epoch 3/5
step 938/938 [-----] - loss: 0.1799 - acc: 0.9757 - 25ms/step
Epoch 4/5
step 938/938 [-----] - loss: 0.1799 - acc: 0.9757 - 25ms/step
Epoch 5/5
step 938/938 [-----] - loss: 0.1799 - acc: 0.9757 - 25ms/step

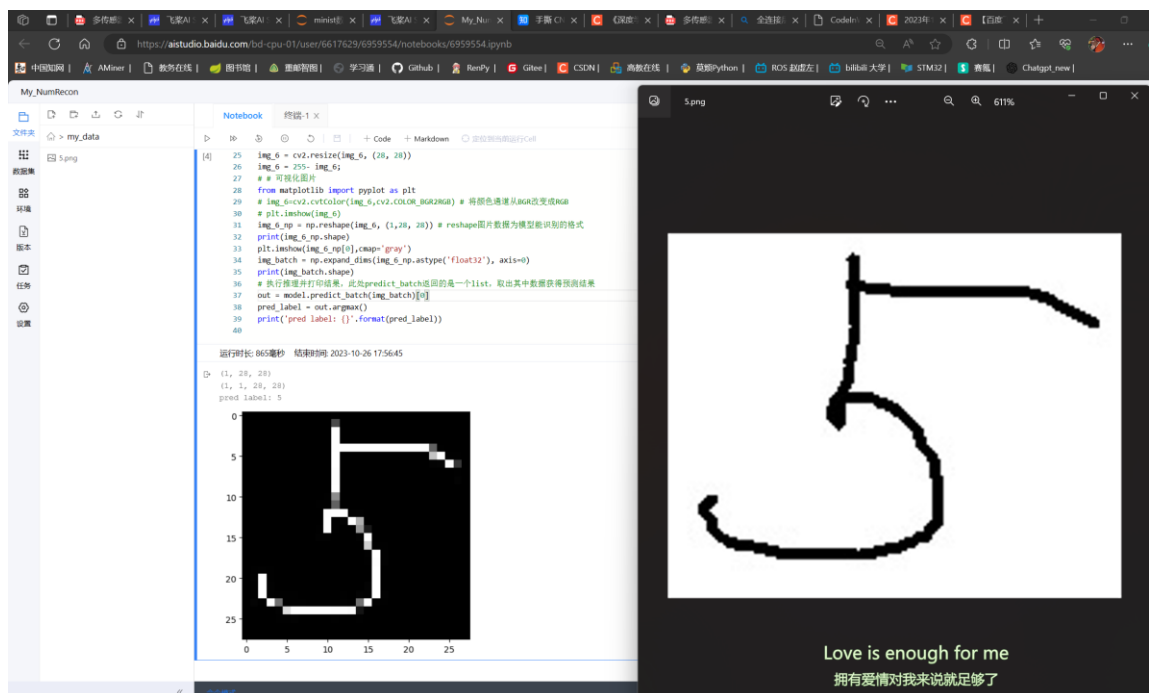
然后新建 Code 片段，输入测试手写数字功能代码，然后将你手写的数字图片上传到百度飞桨平台，新建文件夹 my_data【如果不想叫这个需要改写第 24 行代码的路径】，之后上传你的图片，点击运行。



```
1 import paddle
2 from paddle.vision.transforms import Normalize
3 import numpy as np
4 import cv2
5 transform = Normalize(mean=[127.5], std=[127.5], data_format='chw')
6 test_dataset = paddle.vision.datasets.MNIST(mode='test',
7 transform=transform)
8 # 模型组网并初始化网络
9 lenet = paddle.vision.models.Lenet(num_classes=10)
10 # 封装模型，便于进行后续的训练、评估和推理
11 model = paddle.Model(lenet)
12 # 加载模型
13 model.load('output/mnist')
14 # 从测试集中取出一张图片
15 img, label = test_dataset[0]
16 # 将图片shape从1*28*28变为1*1*28*28，增加一个batch维度，以匹配模型输入格式要求
17 img_batch = np.expand_dims(img.astype('float32'), axis=0)
18 # 输入图像处理
19 img_6 = cv2.imread("my_data/6.png")
20 img_6 = cv2.cvtColor(img_6, cv2.COLOR_RGB2GRAY)
21 img_6 = cv2.resize(img_6, (28, 28))
22 img_6 = np.expand_dims(img_6, 0)
23 # img_batch = np.expand_dims(img_6, 0)
24 img_6 = cv2.imread("my_data/7.png", cv2.IMREAD_GRAYSCALE)
25 img_6 = cv2.resize(img_6, (28, 28))
26 img_6 = 255 - img_6;
27 # 可视化图片
28 from matplotlib import pyplot as plt
29 img_6 = cv2.cvtColor(img_6, cv2.COLOR_BGR2RGB) # 将颜色通道从BGR改变成RGB
30 plt.imshow(img_6)
31 img_6_np = np.reshape(img_6, (1, 28, 28)) # reshape图片数据为模型能识别的格式
32 print(img_6_np.shape)
33 plt.imshow(img_6_np[0], cmap='gray')
34 img_batch = np.expand_dims(img_6_np.astype('float32'), axis=0)
35 print(img_batch.shape)
36 # 执行推理并打印结果，此处predict_batch返回的是一个list，取出其中数据获得预测结果
37 out = model.predict_batch(img_batch)[0]
38 pred_label = out.argmax()
39 print('pred label: {}'.format(pred_label))
40
```

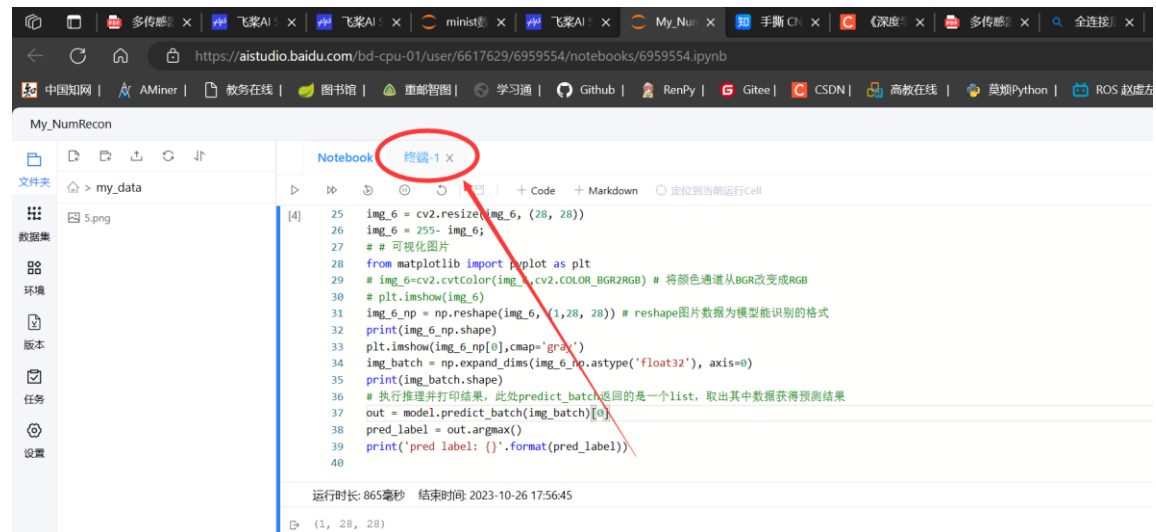
4. 【实际数据运行结果】

这里我上传了一张图片名叫 5.png 的手写数字 5 图片，通过模型预测，输出预测结果为 5，预测结果正确。

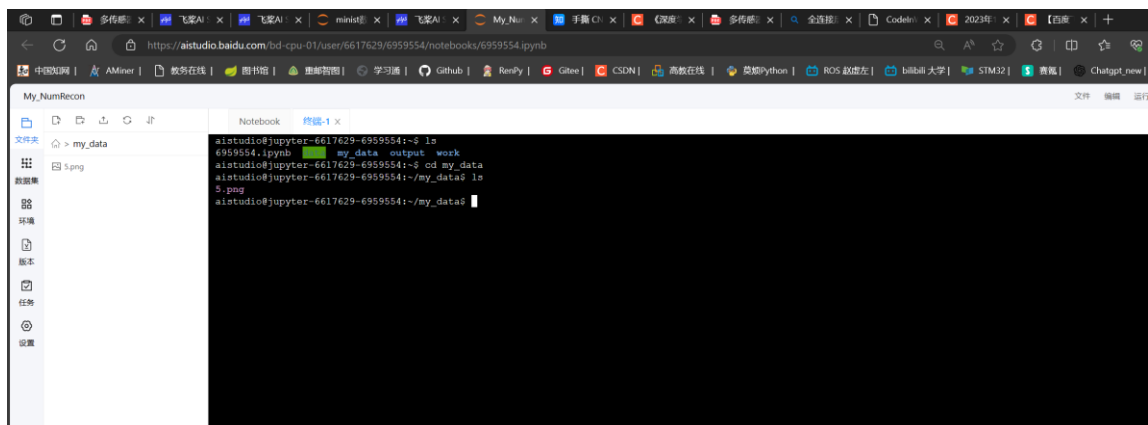


更多细节：

我比较感兴趣的是百度飞桨提供了一个终端：



对于 Linux 操作系统选手有很好的锻炼学习机会。



进入目录文件夹：cd xxx

返回上一级目录：cd ..

返回根目录：cd /

查看目录文件夹文件：ls

查看文件夹下文件大小：df -h

以管理员身份复制目录：sudo cp xx / xx

以管理员身份用 gedit 打开文件：sudo gedit 【gedit 是文档编辑器】

创建文件：mkdir xx

删除文件：rm xx 【rm -f xx 不询问直接删】

查看这个文件的路径名字：pwd

五、实验总结与体会

在本次实验中，成功实现了手写数字识别任务，并通过使用 AlexNet 模型在百度 AI Studio 平台上进行了训练和推理。

首通过学习 AlexNet 的网络结构和原理，我对卷积神经网络的工作方式有了更深入的理解。了解到卷积层、池化层和全连接层等组成部分在图像识别任务中的重要作用，能够有效提取图像特征并进行分类。

在配置百度 AI Studio 平台环境、上传数据集、定义模型等步骤中，我遇到了一些问题。例如，环境配置时需要注意选择合适的 GPU 资源，以加速模型训练过程。此外，数据集的准备和上传也需要仔细检查，确保数据的完整性和正确性。

在模型训练和推理过程中，我发现了优化和调参的重要性。通过调整学习率、批量大小和训练轮数等超参数，可以显著影响模型的性能和收敛速度。同时，我还学到了使用验证集进行模型选择和调参的方法，以避免过拟合和欠拟合问题。

总的来说，本次实验使我对深度学习中常用的卷积神经网络模型有了更深入的理解，并掌握了在百度 AI Studio 平台上进行模型训练和推理的基本技能。通过遇到问题、调试和优化的过程，我不仅加深了对模型原理的理解，还提高了解决问题和调参的能力。这对我今后在深度学习领域的学习和应用都具有重要意义。

六、评阅意见

评阅人签字：

评阅日期：