计算机视觉实验指导书

实验三：基于Mindspore的神经网络与深度学习算法实践

华为智能基座课程



广东工业大学

计算机学院

2023年2月

# 实验介绍

## 本次实验介绍

实验前理论课应该完成high-level视觉部分的讲解，学生已了解、掌握神经网络、反向传播、深度卷积网络的基本原理。本实验介绍了Window系统下的深度学习环境—华为MindSpore—的搭建和使用。

## MindSpore介绍

MindSpore是华为公司自研的最佳匹配昇腾AI处理器算力的全场景深度学习框架，为数据科学家和算法工程师提供设计友好、运行高效的开发体验，推动人工智能软硬件应用生态繁荣发展，目前MindSpore支持在EulerOS、Ubuntu、Windows系统上安装，但是在Windows上只支持CPU，Ubuntu上同时支持CPU和GPU、Ascend910，EulerOS上只支持Ascend910。图1展示了MindSpore深度学习框架的总体结构。其特点如下：

1. 全场景覆盖：MindSpore支持多种设备、多种场景下的深度学习模型开发和发布，包括CPU、GPU、Ascend等处理器，同时也支持分布式和端到端联合训练。
2. 非透明自动微分：MindSpore支持动态图，它可以自动微分，而且支持自动微分和数字信号处理等各方面技术。
3. 安全保障机制：MindSpore针对最近出现的深度学习模型安全问题，加入了多种安全保障机制。
4. 速度快、易用性高：MindSpore使用静态图作为开发模式，能够让用户更方便地进行模型优化，在保证速度的同时还能提高易用性。

MindSpore的使用场景包括：

自然语言处理：MindSpore能够帮助开发者快速构建和训练基于自然语言处理的模型，例如机器翻译、语音识别等。

图像处理：MindSpore提供了丰富的算法支持，能够帮助开发者轻松构建和训练图像识别、图像分类等模型。

数据分析和预测：MindSpore能够处理多维向量、矩阵、张量等数据，能够支持数据分析和预测等工作。

MindSpore支持的语言包括Python、C++等多种编程语言，在Python中几乎可以实现全部功能，极大程度提高了开发效率。

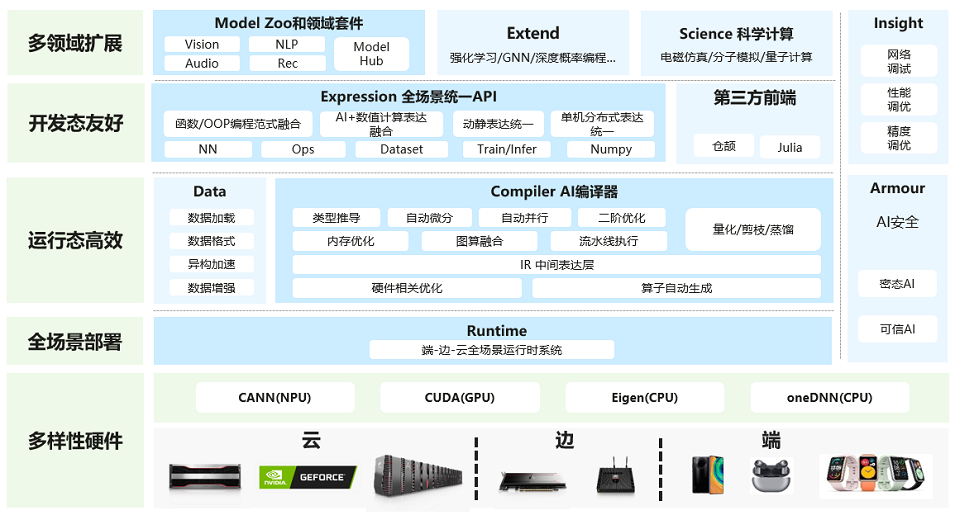


图1. 昇思MindSpore总体架构[[1]](#footnote-1)

# 环境搭建

**注意：先看完教程，再操作。**

**2.1 python版本**

由于MindSpore仅支持python 3.7、3.8、3.9，所以若现有版本过新或过旧，则需重新安装。（也可以创建虚拟环境，其中可指定python的具体版本。该环境独立于之前安装的python。具体操作可参考**附录**。）

在命令提示符下输入python，可显示当前版本，如图2.1所示。



图2.1 查询python版本。

重新安装python包括卸载旧版和安装新版。卸载python可在**控制面板**找到相关项目，然后点击 右键 -> 卸载，如图2.2所示。



图2.2 卸载python.

**注意：** 卸载python，会同时卸载已经安装的软件包。若原有python中已经安装了很多软件包，则推荐使用虚拟环境。或者对软件包列表进行备份：

pip freeze > Path\to\piplist.txt

之后，在新安装的python中，恢复原有软件包列表：

pip install –default-timeout=1000 –r Path\to\piplist.txt

python3.7的下载地址为：

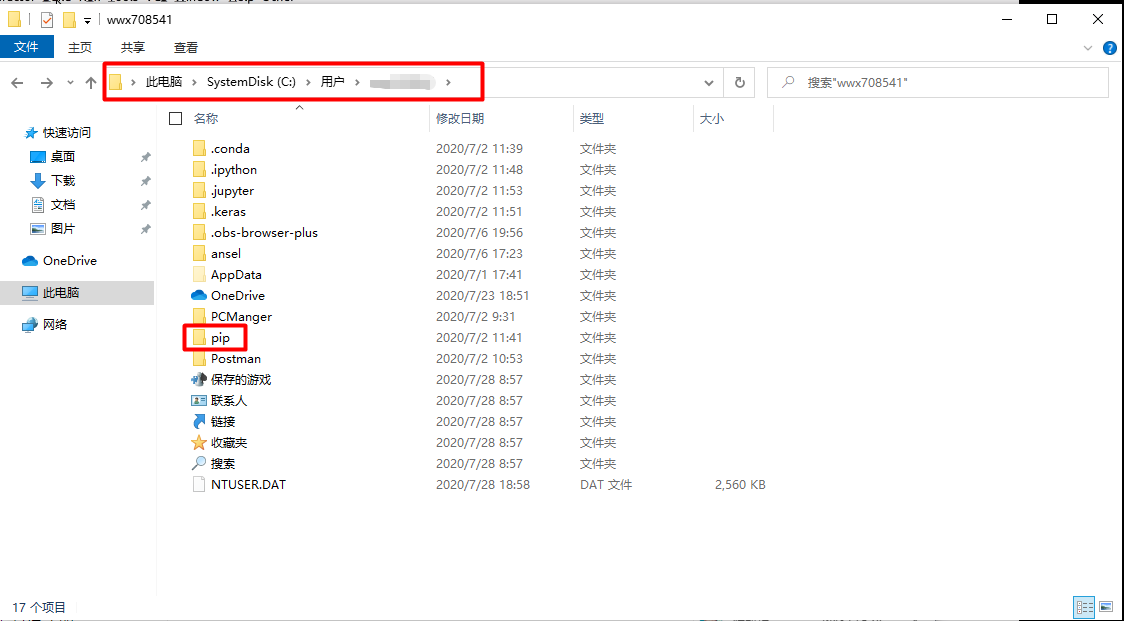
<https://www.python.org/downloads/release/python-377/>

点击“[Windows x86-64 executable installer](https://www.python.org/ftp/python/3.7.7/python-3.7.7-amd64.exe)”即可下载。之后的安装过程与之前相同，在此不再赘述。

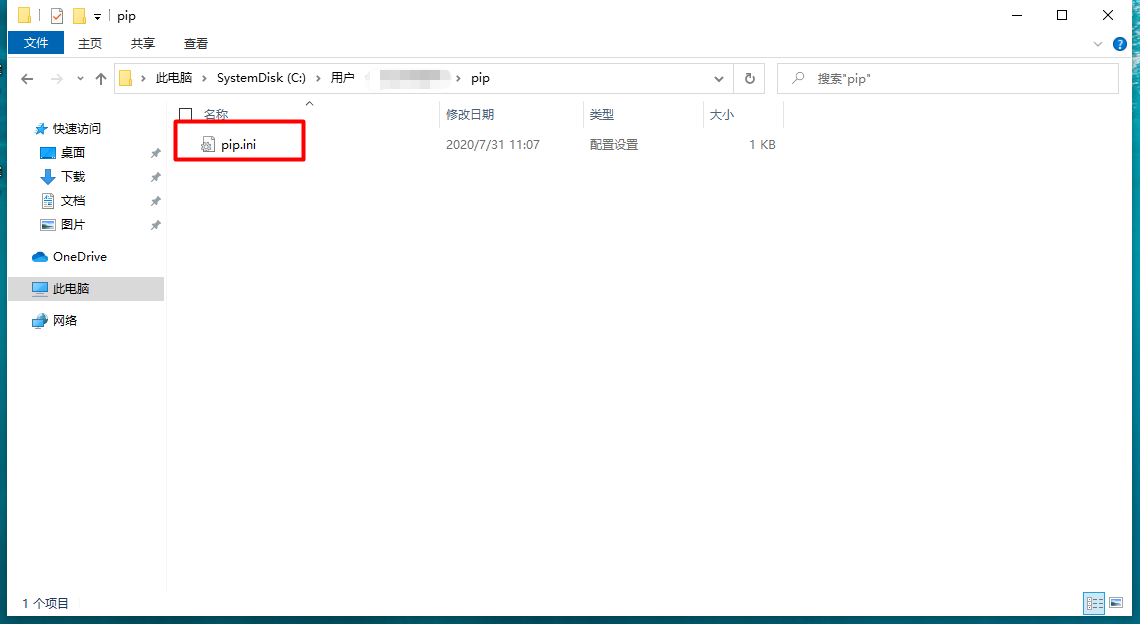
**2.2 pip换源**

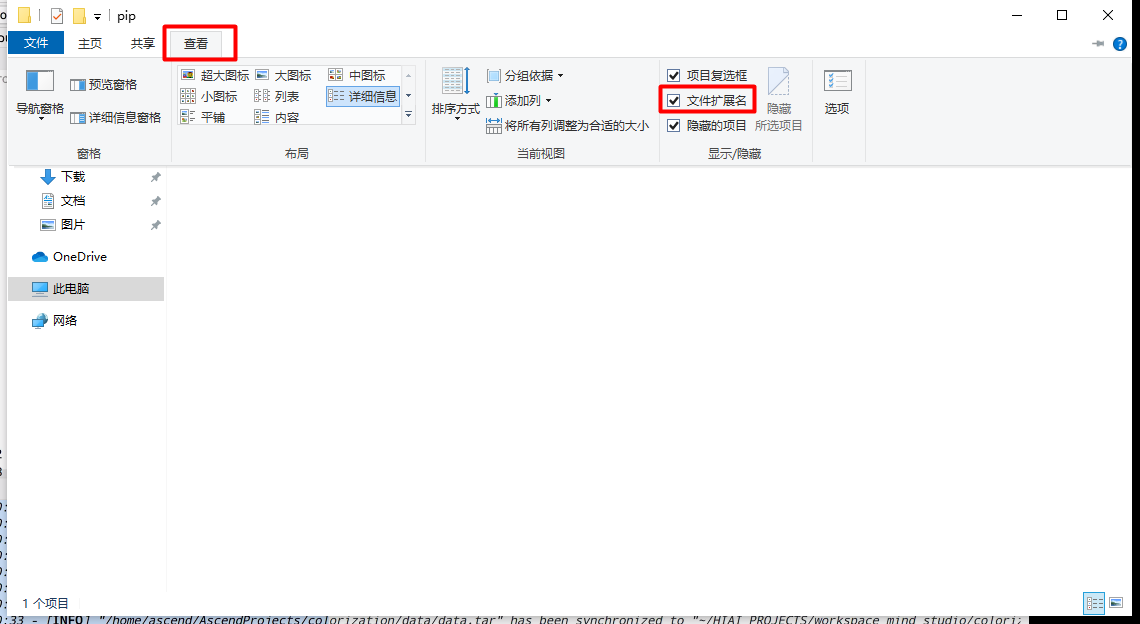
pip是python软件包的管理工具，已经包含在python的安装过程中。由于pip的官方源在国外，直连速度较慢，因此需要换为国内的镜像源。

打开此电脑，进入C盘、用户、用户名，然后新建一个pip文件夹。



新建一个文本文件，然后改名pip.ini，该文件就是pip的配置文件，如果改完之后图标没变化，说明没有显示文件扩展名，点击查看，随后勾选显示文件扩展名。





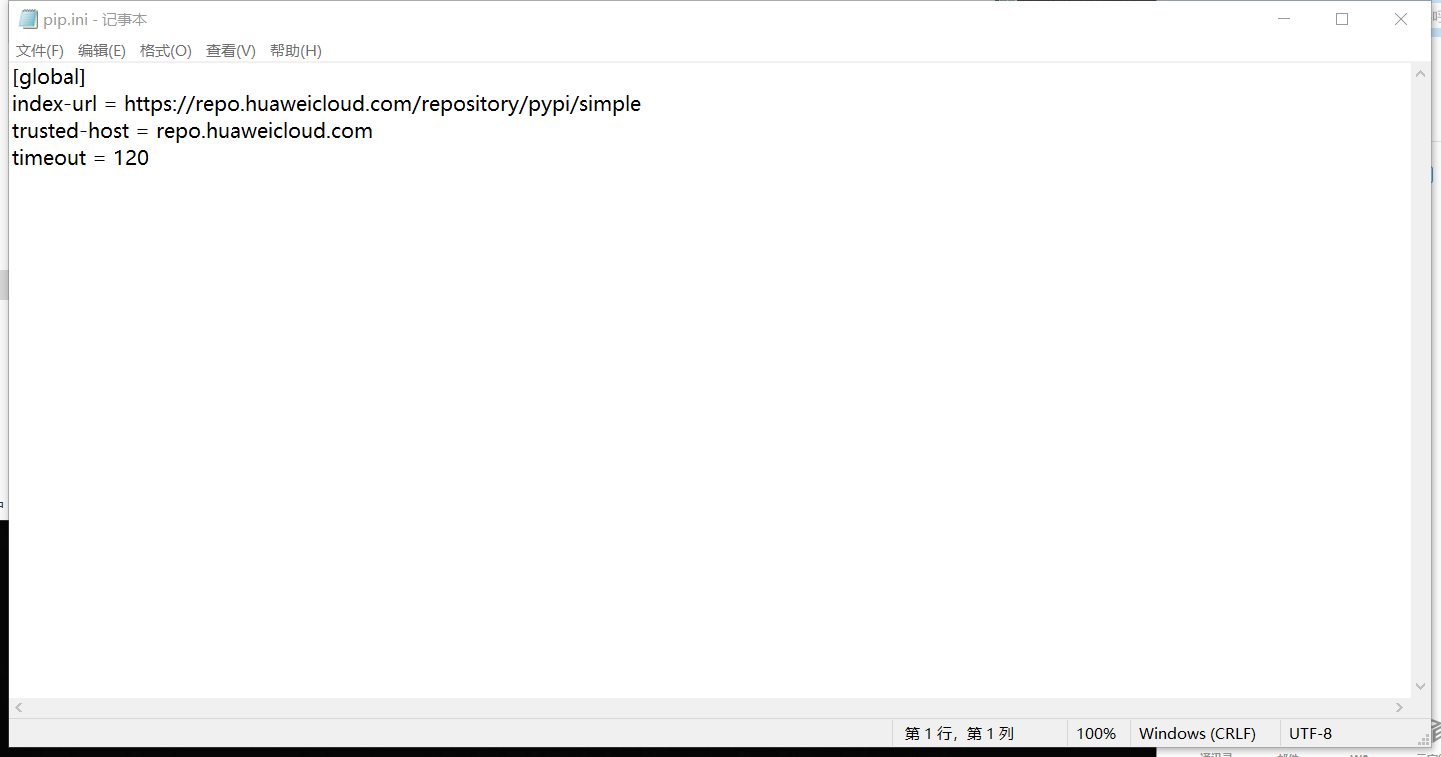
打开pip.ini文件，将以下内容粘贴进去并保存。

[global]

index-url = https://repo.huaweicloud.com/repository/pypi/simple

trusted-host = repo.huaweicloud.com

timeout = 120



可以在命令提示符 输入 pip list，查看已经安装好的python软件包。

**2.3 常用python软件包安装**

如果没有重新安装python，可跳过此节。

若重新安装python，则原有软件包会被清零。可使用如下命令重新安装软件包：

pip install opencv-python

pip install matplotlib

其他以此类推。

**2.4 安装MindSpore**

进入MindSpore官网：

https://www.mindspore.cn/install

根据情况选择对应版本，如图2.3所示。选择好后，最下面出现安装代码。将代码复制到命令提示符，回车，即可开始MindSpore的安装。安装过程会自动下载依赖的其他软件包。

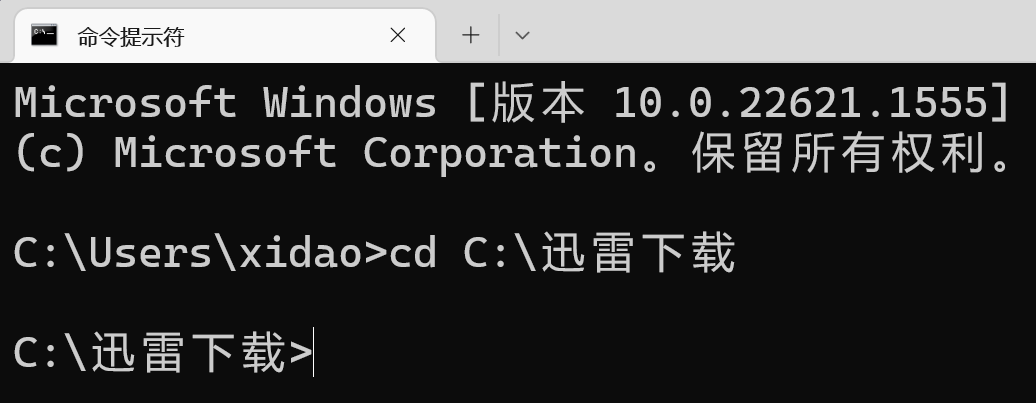


图2.3 MindSpore安装命令获取

有时候，某个软件包（\*.whl）的下载速度太慢，以至于出现安装错误而退出。此时，可在命令提示符窗口内 复制安装包的下载地址，如：

https://pypi.tuna.tsinghua.edu.cn/packages/40/69/4af412d078cef2298f7d90546fa0e03e65a032558bd85319239c72ae0c3c/scipy-1.7.3-cp37-cp37m-win\_amd64.whl

然后用迅雷下载该软件包。最后，cd到下载目录，如：



输入：

pip install **scipy-1.7.3-cp37-cp37m-win\_amd64.whl**

即可手动安装 whl 软件包。之后，可再次输入 MindSpore 的安装命令。

**验证是否成功安装：**

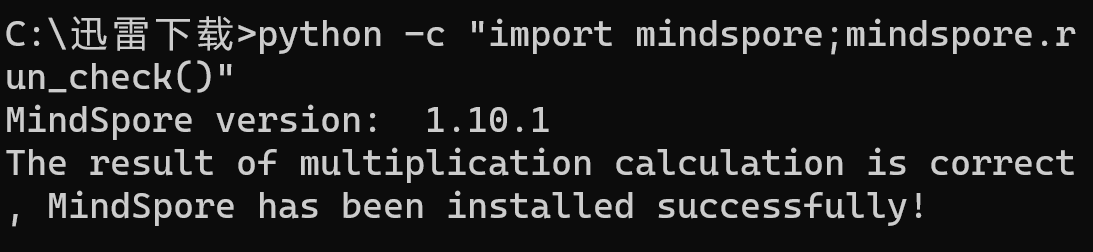
python -c "import mindspore;mindspore.run\_check()"

如果输出：

MindSpore version: 1.10.1

The result of multiplication calculation is correct, MindSpore has been installed successfully!

说明MindSpore安装成功了。



# MindSpore中 张量 的定义与使用

实验要求：结合示例代码，可尝试改变示例代码相关内容与配置，运行结果，并截图记录。(本部分只需要操作，不需要填写代码)

代码：Neural Network\mindspore\_tensor.py

执行python命令，进入python命令行界面。

引入mindspore库。

>>>import numpy as np

>>>import mindspore

>>>from mindspore import ops

>>>from mindspore import Tensor, CSRTensor, COOTensor

若成功引入，通常无任何显示。

张量(Tensor)的概念对于理解深度学习框架中的数据处理和构成是非常重要的。通常，对于计算机视觉任务来说，接触最多的是图像类型数据，通常会出现四维张量，即为(N，H，W，C)维度的tensor。

利用上述张量。主要分以下几个方面：

1. 创建张量：张量的创建方式有多种，构造张量时，支持传入`Tensor`、`float`、`int`、`bool`、`tuple`、`list`和`numpy.ndarray`类型。

**1）由数据直接生成：**

>>>data = [1, 0, 1, 0]

>>>x\_data = Tensor(data)

2）由Numpy数字生成：

>>>np\_array = np.array(data)

>>>x\_np = Tensor(np\_array)

3）使用init初始化器构造张量

当使用init初始化器对张量进行初始化时，支持传入的参数有init、shape、dtype，其中：

init: 支持传入initializer的子类。

shape: 支持传入 list、tuple、 int。

dtype: 支持传入mindspore.dtype。

from mindspore.common.initializer import One, Normal

# 初始化一个全1的tensor

tensor1 = mindspore.Tensor(*shape*=(2, 2), *dtype*=mindspore.float32, *init*=One())

# 初始化一个符合标准正态分布的Tensor

tensor2 = mindspore.Tensor(*shape*=(2, 2), *dtype*=mindspore.float32, *init*=Normal())

# 打印查看结果(print函数，python的debug神器)

print("tensor1:\n", tensor1)

print("tensor2:\n", tensor2)

4）继承另外一个Tensor的大小、类型，例如申请全零和全1张量时

from mindspore import ops

x\_ones = ops.ones\_like(x\_data)

print(f"Ones Tensor: \n {x\_ones} \n")

x\_zeros = ops.zeros\_like(x\_data)

print(f"Zeros Tensor: \n {x\_zeros} \n")

1. 查看张量属性

张量的属性包括形状、数据类型、转置张量、单个元素大小、占用字节数量、维数、元素个数和每一维步长。可通过查看如下属性，显示具体类型，例如：

* 形状（shape）：Tensor的shape，是一个tuple。
* 数据类型（dtype）：Tensor的dtype，是MindSpore的一个数据类型。
* 单个元素大小（itemsize）： Tensor中每一个元素占用字节数，是一个整数。
* 占用字节数量（nbytes）： Tensor占用的总字节数，是一个整数。
* 维数（ndim）： Tensor的秩，也就是len(tensor.shape)，是一个整数。
* 元素个数（size）： Tensor中所有元素的个数，是一个整数。
* 每一维步长（strides）： Tensor每一维所需要的字节数，是一个tuple。

# 可尝试其他数据类型和大小,查看返回的维度等信息

x = Tensor(np.array([[1, 2], [3, 4]]), mindspore.int32)

print("x\_shape:", x.shape)

print("x\_dtype:", x.dtype)

print("x\_itemsize:", x.itemsize)

print("x\_nbytes:", x.nbytes)

print("x\_ndim:", x.ndim)

print("x\_size:", x.size)

print("x\_strides:", x.strides)

1. 张量如何索引

即如何获得具位置、范围Tensor中的数值。Tensor索引与Numpy索引类似，索引从0开始编制，负索引表示按倒序编制，冒号:和 ...用于对数据进行切片。

# 尝试使用其他非示例代码,熟悉检索具体行、列、范围内的数值

tensor = Tensor(np.array([[0, 1], [2, 3]]).astype(np.float32))

print("First row: {}".format(tensor[0]))

print("value of bottom right corner: {}".format(tensor[1, 1]))

print("Last column: {}".format(tensor[:, -1]))

print("First column: {}".format(tensor[..., 0]))

1. 张量运算

张量之间有很多运算，包括算术、线性代数、矩阵处理（转置、标引、切片）、采样等，张量运算和NumPy的使用方式类似，下面介绍其中几种操作：

x = Tensor(np.array([1, 2, 3]), mindspore.float32)

y = Tensor(np.array([4, 5, 6]), mindspore.float32)

output\_add = x + y

output\_sub = x - y

output\_mul = x \* y

output\_div = y / x

output\_mod = y % x

output\_floordiv = y // x

print("add:", output\_add)

print("sub:", output\_sub)

print("mul:", output\_mul)

print("div:", output\_div)

print("mod:", output\_mod)

print("floordiv:", output\_floordiv)

1. 其他张量操作

1）Tensor级联concat：通过指定具体的维度，进行级联

data1 = Tensor(np.array([[0, 1], [2, 3]]).astype(np.float32))

data2 = Tensor(np.array([[4, 5], [6, 7]]).astype(np.float32))

output = ops.concat((data1, data2), *axis*=0)

print(output)

print("shape:\n", output.shape)

2）Tensor的堆叠：

data1 = Tensor(np.array([[0, 1], [2, 3]]).astype(np.float32))

data2 = Tensor(np.array([[4, 5], [6, 7]]).astype(np.float32))

output = ops.stack([data1, data2])

print(output)

print("shape:\n", output.shape)

# 前向传播

代码： Neural Network\forward.py

实现了一个简单的神经网络。代码可直接运行。

**实验报告**要求：

1. 展示运行结果。
2. 分析 class Net(nn.Cell) 的功能。
3. 对代码中红色部分添加注释，解释其作用。注释的位置和格式已经给出，填好后将代码复制到 实验报告 内。

import numpy as np

import mindspore.nn as nn

import mindspore.ops as ops

from mindspore import Tensor

from mindspore import ParameterTuple, Parameter

from mindspore import dtype as mstype

class Net(nn.Cell):

def \_\_init\_\_(self):

super(Net, self).\_\_init\_\_()

#

self.matmul = ops.MatMul()

#

self.z = Parameter(Tensor(np.array([1.0, 1.0, 1.0], np.float32)), name='z')

def construct(self, x, y):

x = x \* self.z

out = self.matmul(x, y)

return out

#

model = Net()

for m in model.parameters\_and\_names():

print(m)

#

x = Tensor([[0.8, 0.6, 0.2], [1.8, 1.3, 1.1]], dtype=mstype.float32)

y = Tensor([[0.11, 3.3, 1.1], [1.1, 0.2, 1.4], [1.1, 2.2, 0.3]], dtype=mstype.float32)

result = model(x, y)

print(result)

# 网络构建

代码： Neural Network\mindspore\_model.py

神经网络模型是由神经网络层和Tensor操作构成的，mindspore.nn提供了常见神经网络层的实现，在MindSpore中，Cell类是构建所有网络的基类，也是网络的基本单元。一个神经网络模型表示为一个Cell，它由不同的子Cell构成。使用这样的嵌套结构，可以简单地使用面向对象编程的思维，对神经网络结构进行构建和管理。下面我们将构建一个用于Mnist数据集分类的神经网络模型。

代码可直接运行。

**实验报告**要求：

1. 分别展示每一个步骤的运行结果。
2. 回答如下问题：
3. 代码第35行，nn.Dense(28\*28, 512) 中的 28\*28是什么含义？
4. 代码第39行，(512, 10) 的含义是什么？
5. 第60行，(1, 28, 28) 的含义是什么？

# LeNet手写体数字识别

代码： LeNet\ Lenet.py 和 test.py

Lenet.py 包括数据集的处理、网络构建、训练和测试。

test.py 包含更详细的测试。

**实验报告**要求：

1. 分别展示每一个步骤的运行结果。
2. 修改 Lenet.py，增加训练过程的 epoch，从而使测试精度达到0.96以上。
3. 回答如下问题：
4. 解释 Lenet.py 代码第110行中 sparse 参数 和 reduction 参数的含义。
5. 解释第113行 momentum 参数的含义。
6. 解释第132行 dataset\_sink\_mode 参数的含义。

提示：可在 MindSpore 官方文档中查看函数、参数的意义：

https://www.mindspore.cn/tutorials/zh-CN/r1.10/beginner/introduction.html

**附录：基于 Miniconda 的虚拟环境安装**

## 1.1 Miniconda介绍

Miniconda[[2]](#footnote-2)是一个轻量级的Anaconda版本，它包括了Python解释器和conda包管理器，但只包含少量的基本包和库。它的主要目的是为了让用户能够更加灵活地配置自己的Python环境，同时避免Anaconda安装过多的软件包和库，造成磁盘空间和性能的浪费。

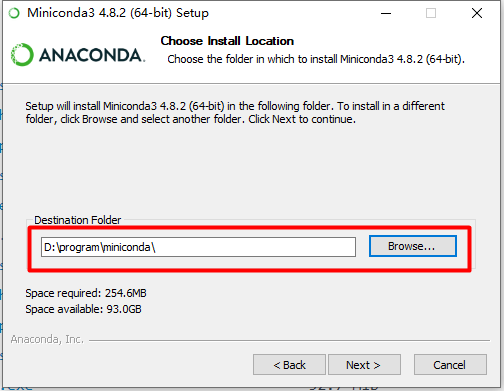
Conda是一个跨平台的包管理器和环境管理器，它可以用于安装、升级和删除软件包，并且可以创建和管理多个独立的Python环境。Conda可以在Linux、Windows和MacOS等操作系统上运行，并且可以管理不同版本的Python和各种依赖库。Conda还可以用于创建和共享自定义软件包，方便其他用户进行安装和使用。

## 1.2 Miniconda安装

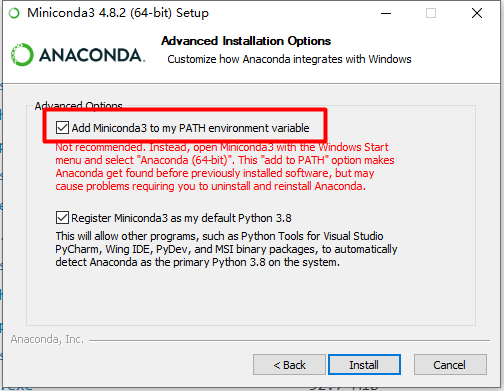
从1.1.2提供的链接下载Miniconda的Windows版本对应的64位安装包，由于官方源下载速度慢，实验所用安装包为清华源下载，带有x86\_64的为64位安装包。



双击安装包进行安装，点击next，然后选择安装位置。



环境变量打勾，这样可以直接在命令行中启动Miniconda。

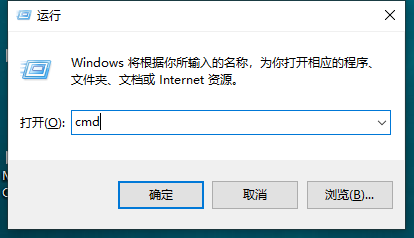


等待安装成功，然后点击Finish。

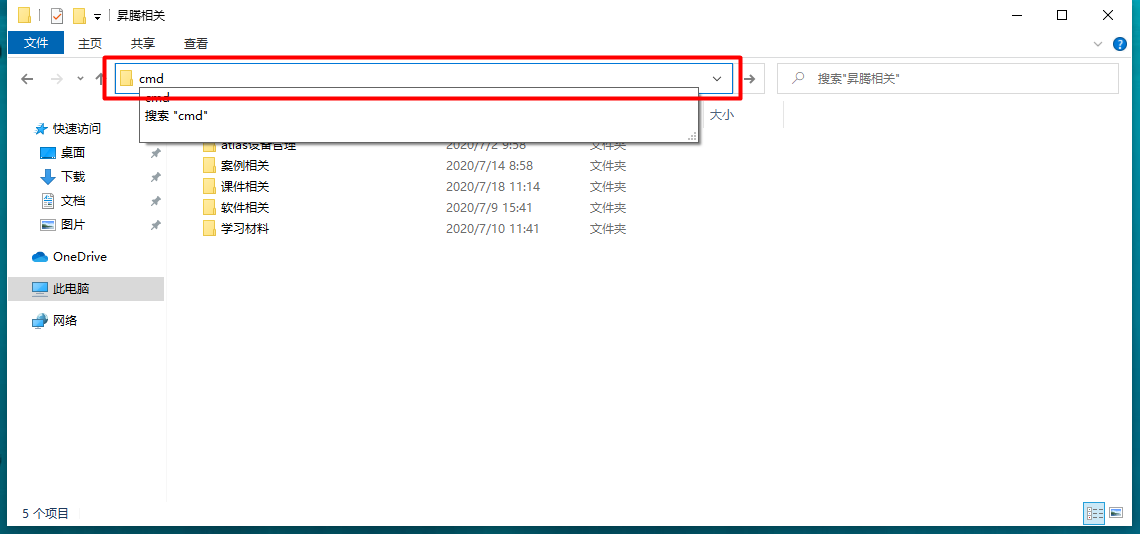
## 1.3 创建虚拟环境

在Window中有多种方式开启命令行窗口，这里介绍两种，按下win+R键，然后输入cmd点击确定，或者任意打开一个文件夹，在上方地址栏输入cmd，然后按回车键。

运行打开命令行界面如下图所示：

z

地址栏打开命令行界面如下图所示：



打开命令行窗口之后，输入以下命令创建虚拟环境，Python版本为3.7.5，创建过程需要输入y确认。

>> conda create -n hwms python==3.8

虚拟环境创建成功后输入对应名称即可进入对应虚拟环境

>> activate hwms

1. <https://www.mindspore.cn/tutorials/zh-CN/r2.0/beginner/introduction.html> [↑](#footnote-ref-1)
2. <https://docs.conda.io/en/latest/miniconda.html> [↑](#footnote-ref-2)