EE 569 Discussion

Zhiruo Zhou

04/02/2021

Agenda

- Today: HW5
- 04/09: review for midterm#2 by Zohreh
- 04/16: HW6 part 1 by Yao
- …

Content

- Setup your environment
  - Method 1: Setup local environment on your own laptop
    - Get everything in your control
    - Complicated in setup
    - You can only work on CPU if you don’t have GPU
  - Method 2: Use online environment
    - Slow response sometimes
    - No need for any local setup!
    - Free GPU (with certain usage limitations, no guarantee for availability)

- Coding a neural network

Setup local environment on your own laptop

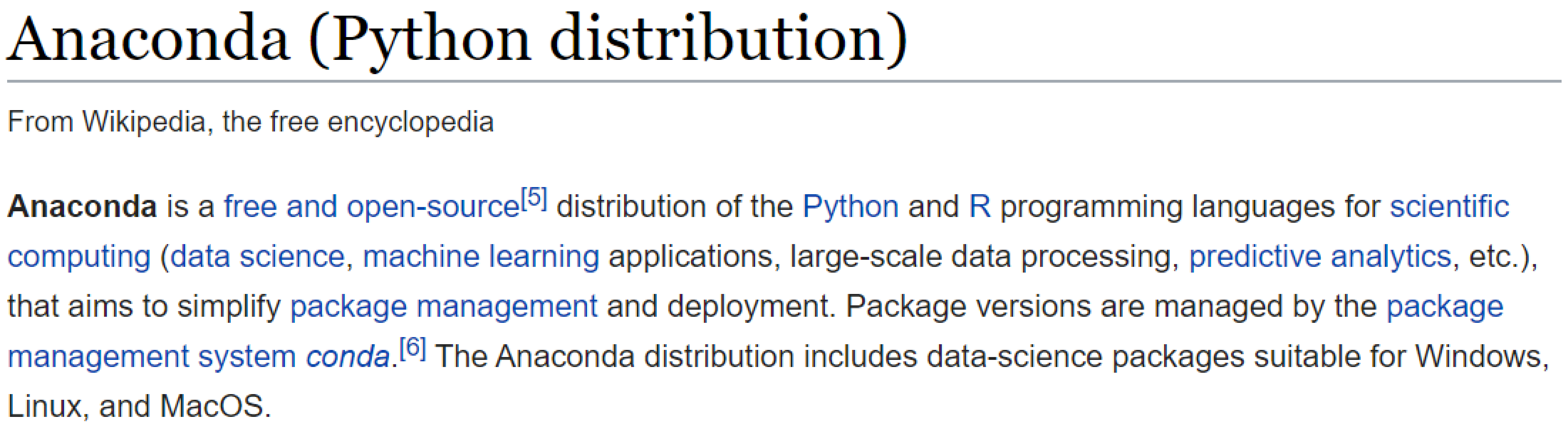
Set up the coding environment

- You will use **Python3**.
- Free to use any environment or libraries as you like.
- We provide recommendations on environment settings, but will NOT help you setup your environment. Because the setup process is not like solving a problem which has known answers. Lots of bugs can occur and they are very dependent on the specific environment of your computer. Google is recommended for solving any bugs you encounter during this process.
- Start ASAP.

Recommended setting

- Recommended environment setting:
  - Anaconda (Python 3) + Pytorch / Keras
- We provide reference on installation steps for users of **Windows**.
- Users of other operation systems can figure out corresponding steps through Google or reading documents on relevant websites.
  - Anaconda: https://docs.anaconda.com/anaconda/install/
  - Pytorch: https://pytorch.org/

Anaconda

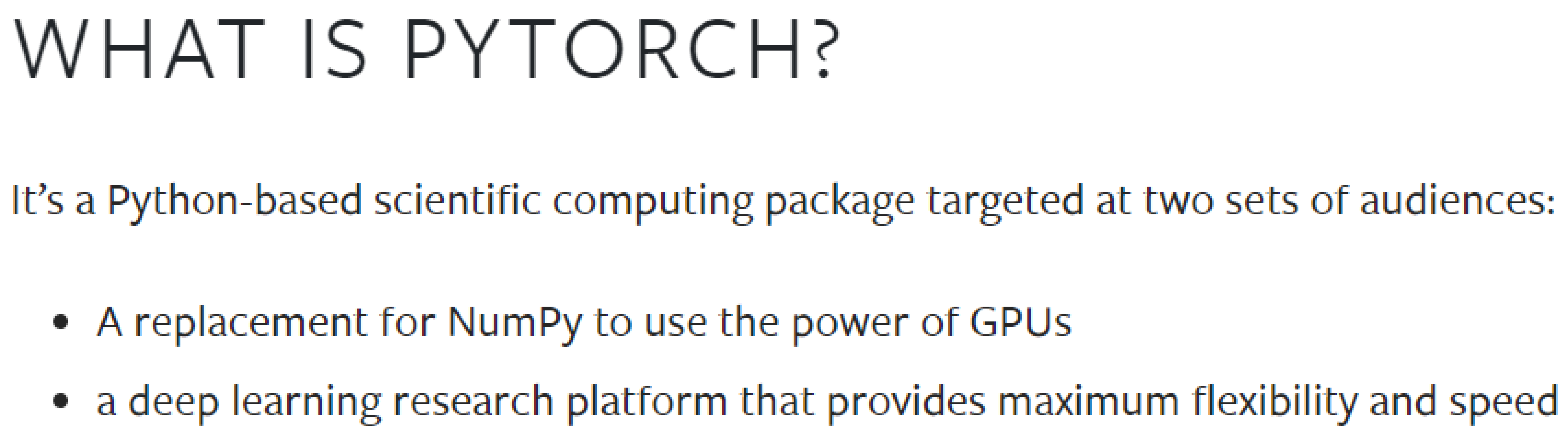


Anaconda (Python distribution)

From Wikipedia, the free encyclopedia

Anaconda is a free and open-source[5] distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system *conda*.[6] The Anaconda distribution includes data-science packages suitable for Windows, Linux, and MacOS.

Pytorch



WHAT IS PYTORCH?

It’s a Python-based scientific computing package targeted at two sets of audiences:

- A replacement for NumPy to use the power of GPUs
- a deep learning research platform that provides maximum flexibility and speed

- See the following website for more info.
  - https://pytorch.org/tutorials/beginner/blitz/tensor_tutorial.html

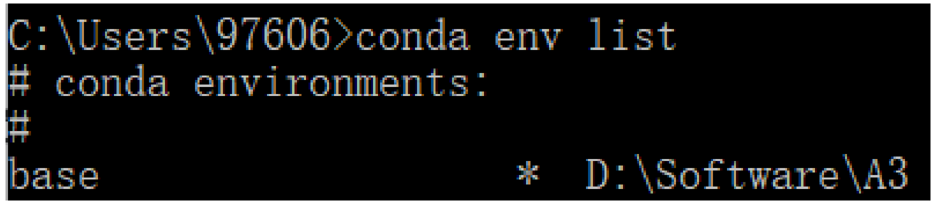
Installation steps for Windows

- Step 1: install Anaconda
  - Follow the instruction on
  - https://docs.anaconda.com/anaconda/install/windows/

Installation steps for Windows

- Step 2: create virtual environment (VE) in Anaconda

- Open cmd
- Run command ‘`conda env list`’ to see your current VEs. Now you only have base.

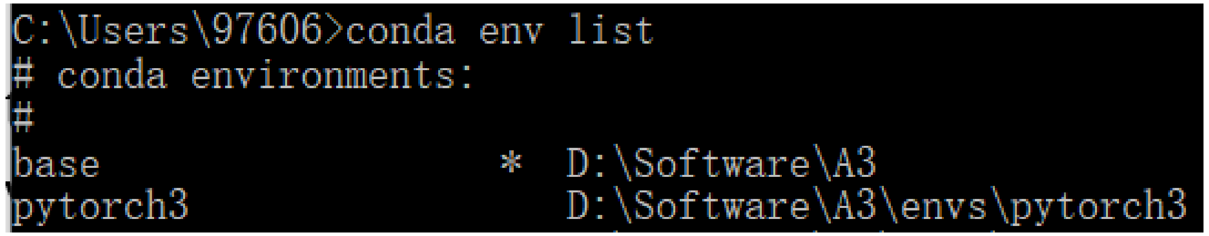


```
C:\Users\97606>conda env list
# conda environments:
#
base                  *  D:\Software\A3
```

- Create a new VE named pytorch3:
  - run command ‘`conda create -n pytorch3 python=X.X`’
  - x.x is the version of Python you want, for example 3.5

Installation steps for Windows

- After creating the new VE pytorch3, you'll be able to see it if you run command `'conda env list'`

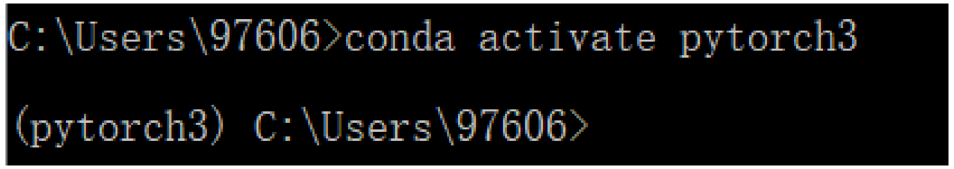


```
C:\Users\97606>conda env list
# conda environments:
#
base                  *  D:\Software\A3
pytorch3                 D:\Software\A3\envs\pytorch3
```

- Now activating the VE pytorch3:

  run command `'conda activate pytorch3'`

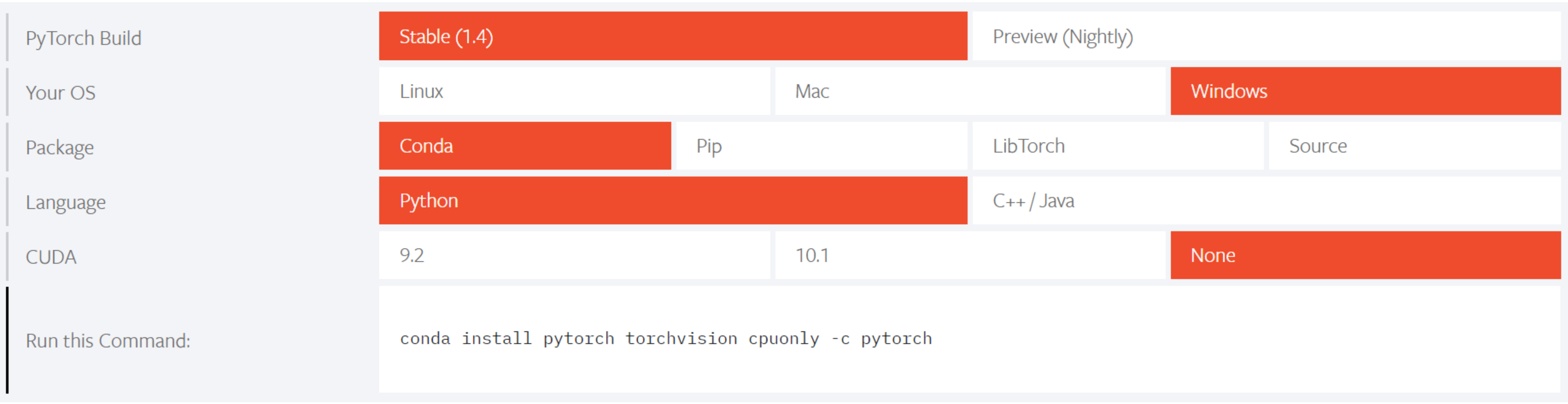
  you can see the (pytorch3) before the command line:



```
C:\Users\97606>conda activate pytorch3

(pytorch3) C:\Users\97606>
```

Installation steps for Windows

- Step 3: install Pytorch in the new VE
  - Go to https://pytorch.org/ and find the one that fits you
  - Run the corresponding command



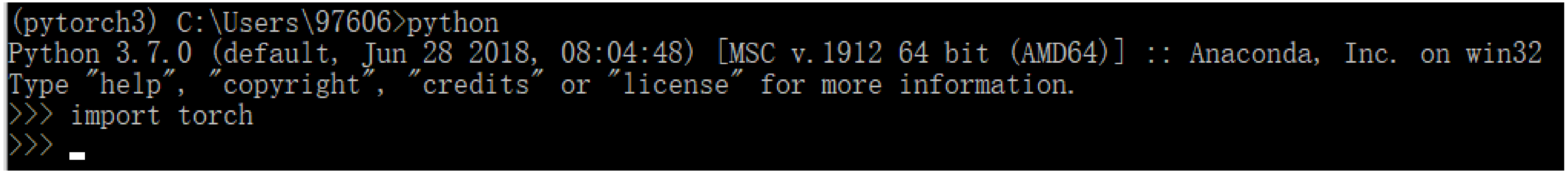
PyTorch Build	[✓ Stable (1.4)]	Preview (Nightly)		
Your OS	Linux	Mac	[✓ Windows]	
Package	[✓ Conda]	Pip	LibTorch	Source
Language	[✓ Python]	C++ / Java		
CUDA	9.2	10.1	[✓ None]	

Run this Command:

```
conda install pytorch torchvision cpuonly -c pytorch
```

Installation steps for Windows

- Test whether you install Pytorch successfully
  - Try to import torch in python. If no error occurs, then it's installed successfully.

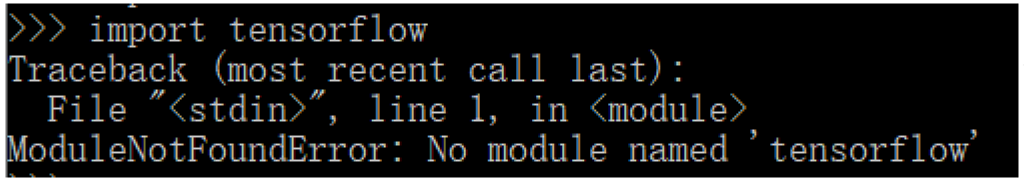


```
(pytorch3) C:\Users\97606>python
Python 3.7.0 (default, Jun 28 2018, 08:04:48) [MSC v.1912 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> _
```

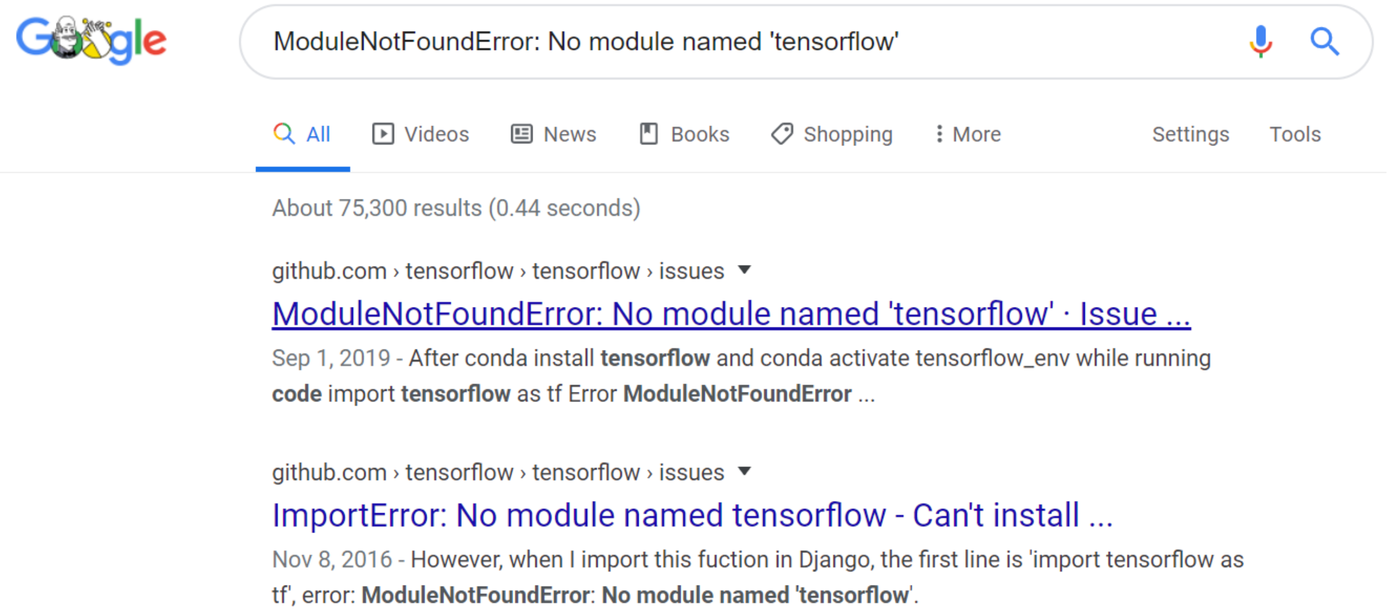
- If you successfully finish this step, then congratulations! The initial environment setting is done.
- You may want to install other libraries later. Help yourself with Google.

Installation steps for Windows

- Again, these are just examples. Google if you encounter any difficulties during the process.
- How to google my error? Example provided below. As easy as you think.



```
>>> import tensorflow
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'tensorflow'
```



ModuleNotFoundError: No module named 'tensorflow'

About 75,300 results (0.44 seconds)

github.com › tensorflow › tensorflow › issues

ModuleNotFoundError: No module named 'tensorflow' · Issue ...

Sep 1, 2019 - After conda install **tensorflow** and conda activate tensorflow_env while running **code** import **tensorflow** as tf Error **ModuleNotFoundError** ...

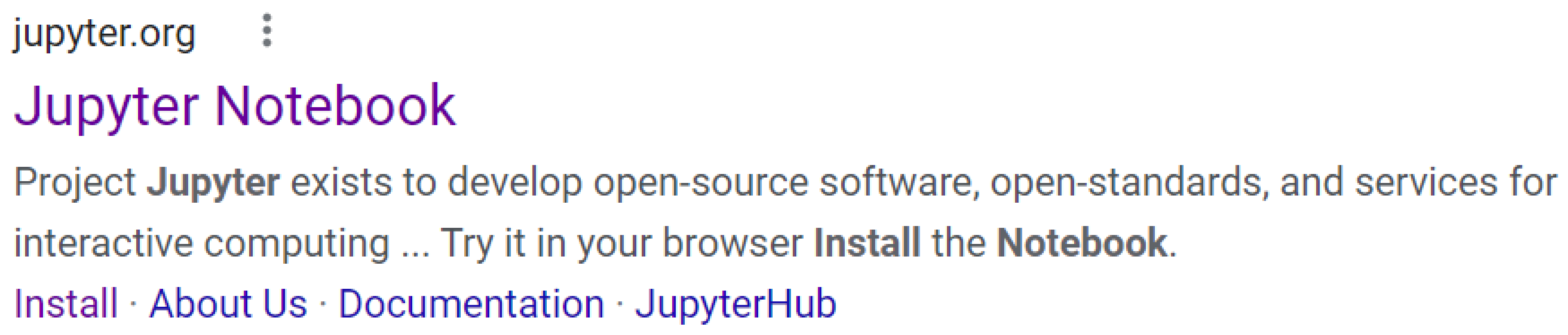
github.com › tensorflow › tensorflow › issues

ImportError: No module named tensorflow - Can't install ...

Nov 8, 2016 - However, when I import this fuction in Django, the first line is 'import tensorflow as tf', error: **ModuleNotFoundError**: **No module named 'tensorflow'**.

Jupyter notebook (optional tool)

- An interactive programming tool working on your browser



jupyter.org

Jupyter Notebook

Project **Jupyter** exists to develop open-source software, open-standards, and services for interactive computing ... Try it in your browser **Install** the **Notebook**.

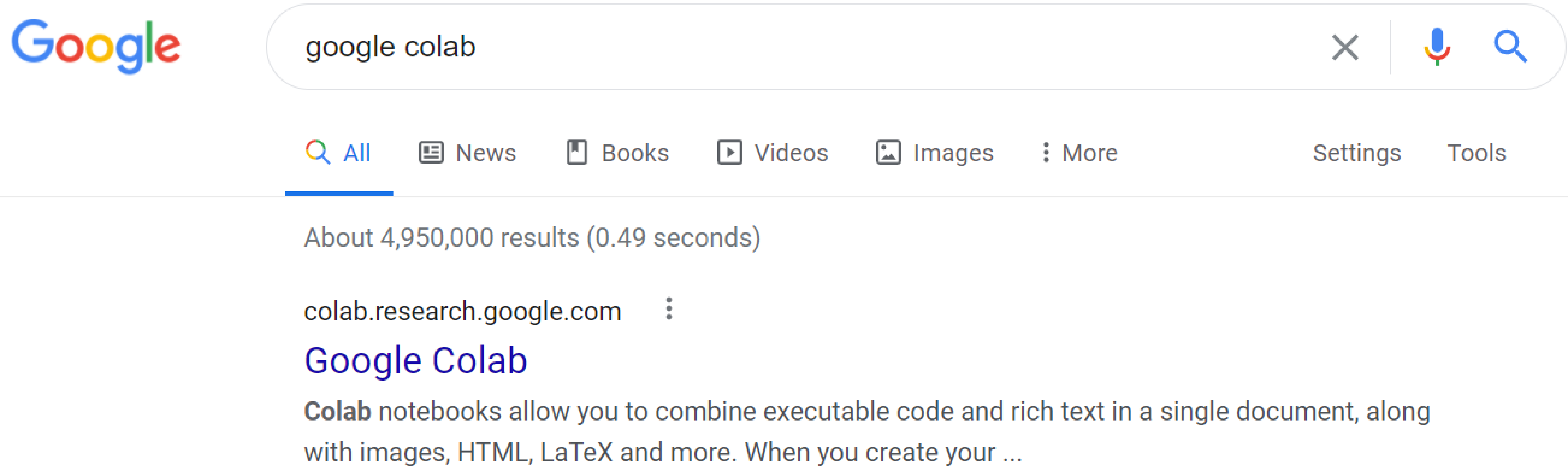
Install · About Us · Documentation · JupyterHub

- Installation tutorial
  - https://test-jupyter.readthedocs.io/en/latest/install.html
  - Any other workable resource you can google

Use online environment

Google Colab

- Register an account. It's related with your google drive account.



google colab

All News Books Videos Images More Settings Tools

About 4,950,000 results (0.49 seconds)

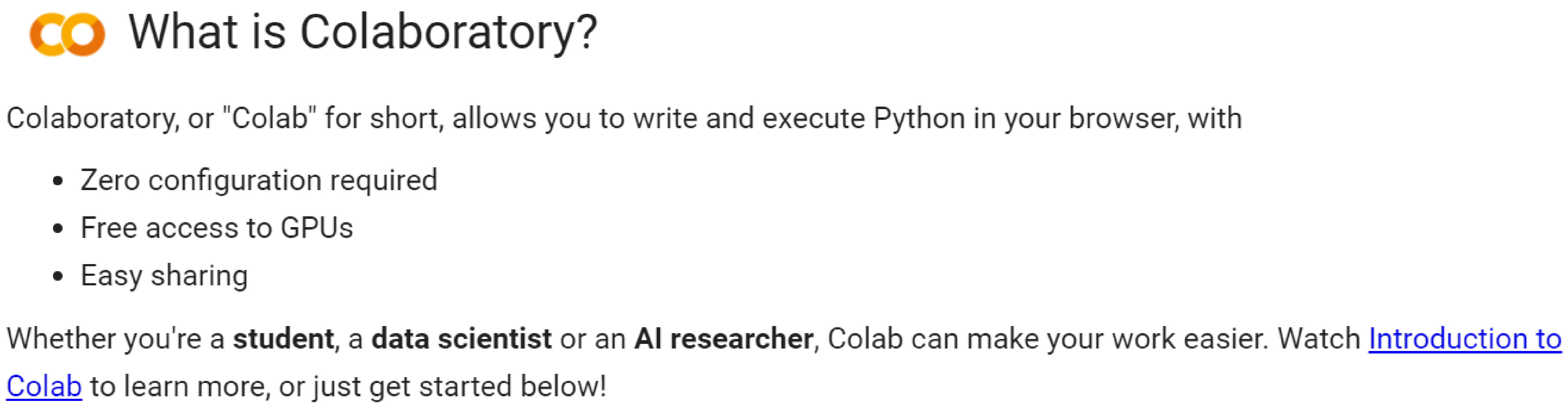
colab.research.google.com

Google Colab

Colab notebooks allow you to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. When you create your ...

Tutorials

- The welcome page on Colab



What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch Introduction to Colab to learn more, or just get started below!

- Tutorial from other websites:
  - https://www.tutorialspoint.com/google_colab/index.htm

Code a neural network via Pytorch

- https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html

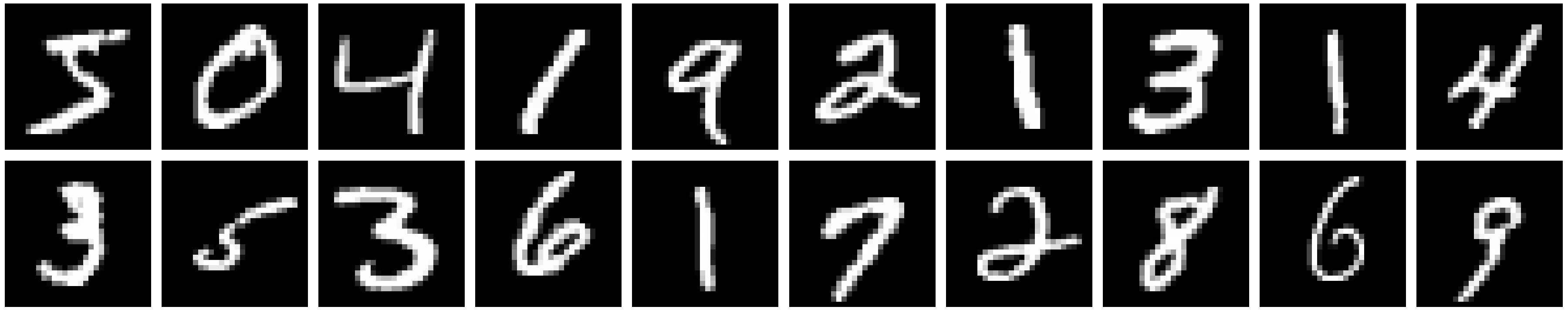
Dataset

- Classical datasets for image classification

- MNIST
- Fashion MNIST
- CIFAR-10
- CIFAR-100
- ImageNet

MNIST

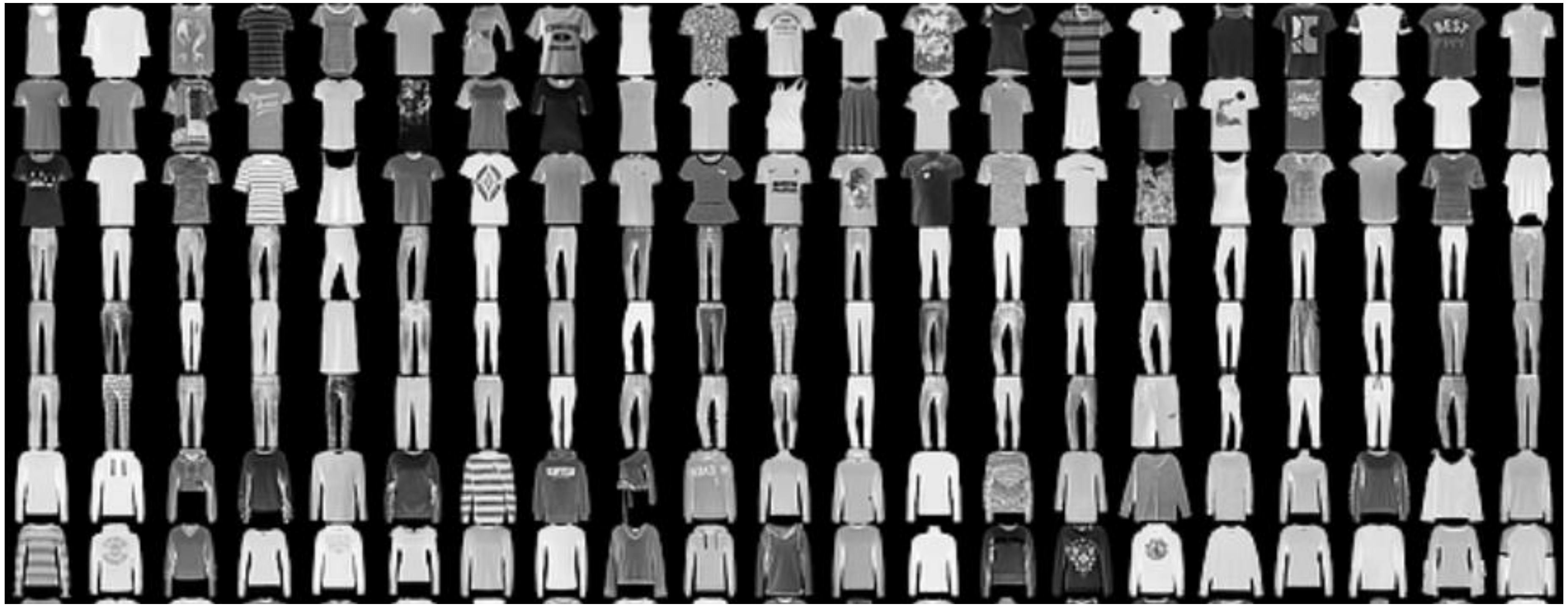
- Dataset of handwritten digits
  - 10 classes
  - Training set 60,000 samples
  - Testing set 10,000 samples



http://yann.lecun.com/exdb/mnist/

Fashion MNIST

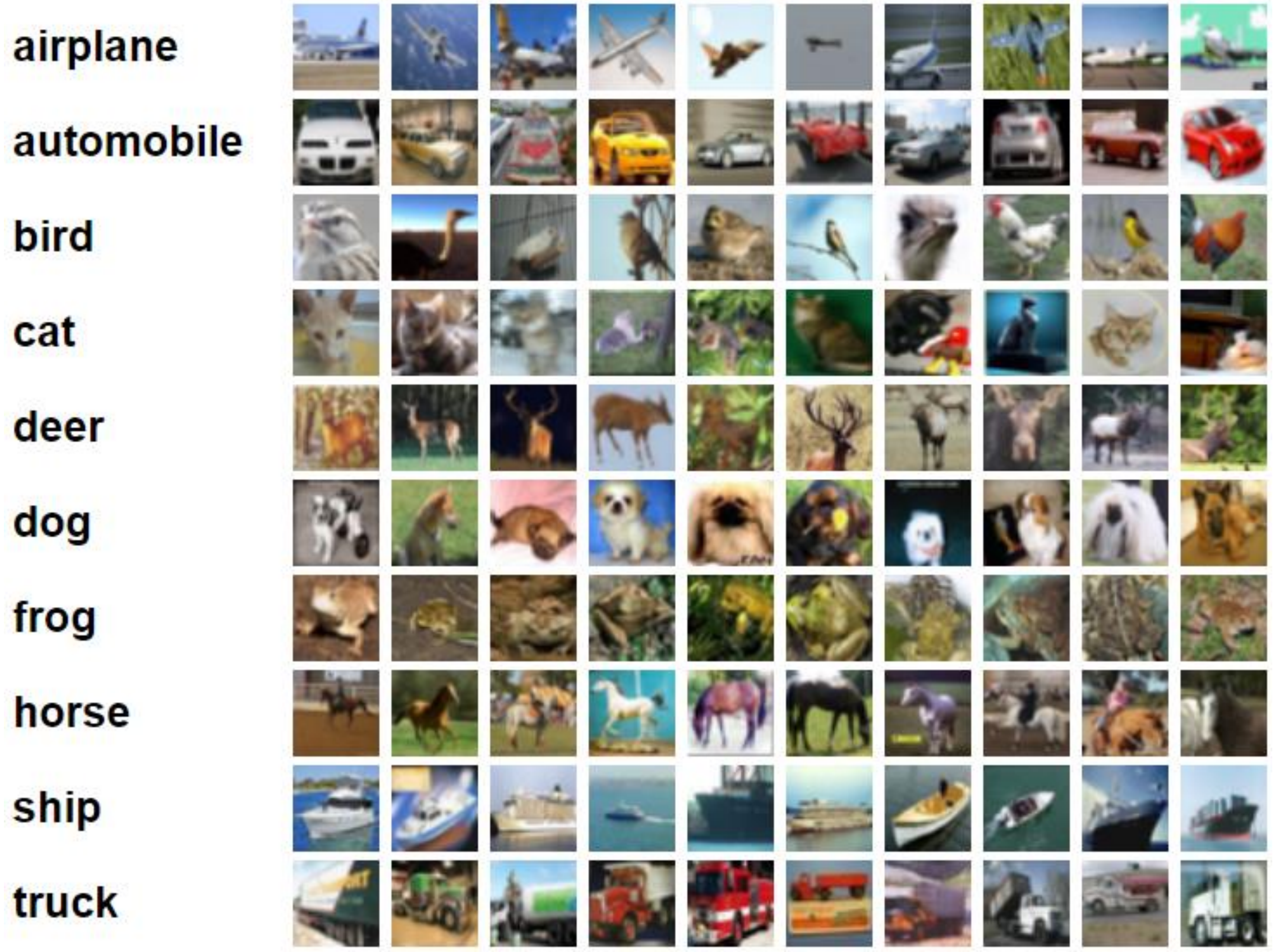
- Dataset of Zalando's article images
  - 10 classes
  - Training set 60,000 samples
  - Testing set 10,000 samples



https://github.com/zalandoresearch/fashion-mnist

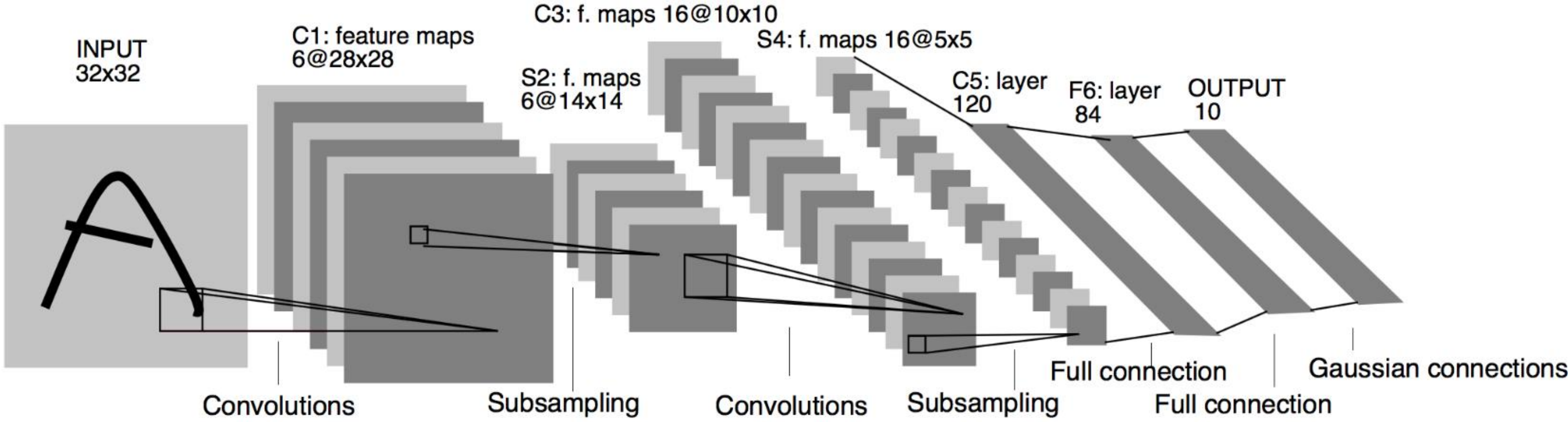
CIFAR-10

- Dataset of color images of objects
  - 10 classes
  - Training set 50,000 samples
  - Testing set 10,000 samples



https://www.cs.toronto.edu/~kriz/cifar.html

Example network on MNIST

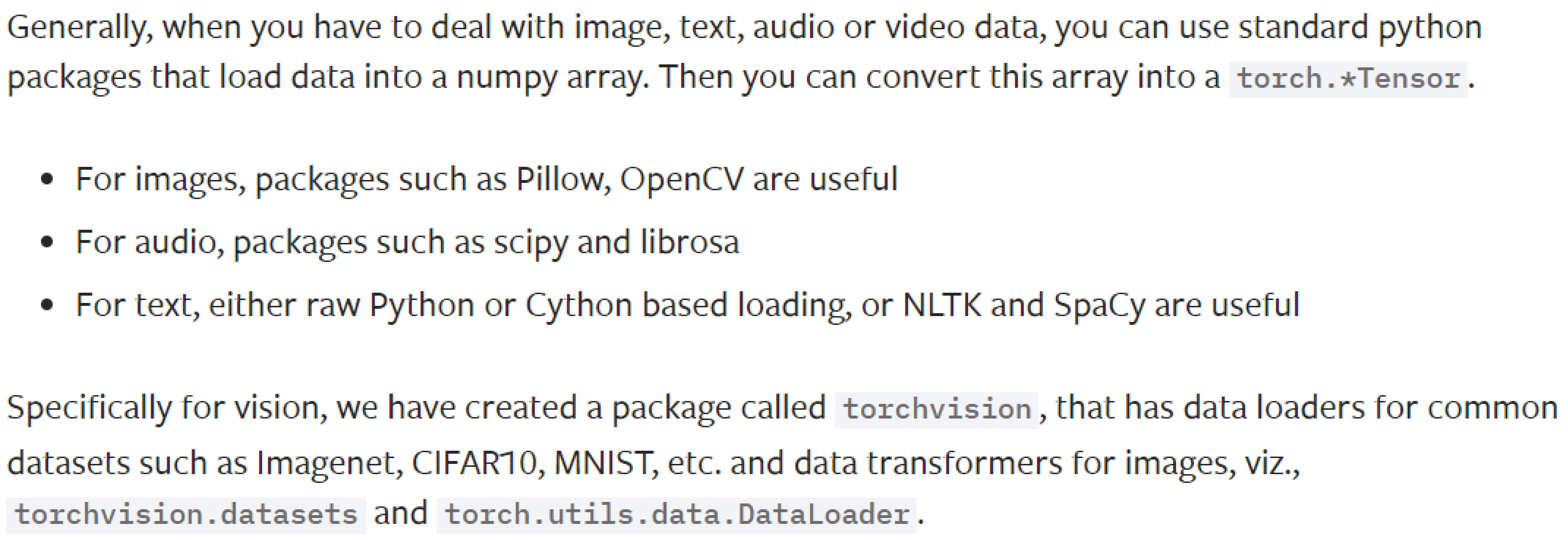


Define a network

```python
import torch
import torch.nn as nn
import torch.nn.functional as F
```

```python
class Net(nn.Module):

    def __init__(self):
        super(Net, self).__init__()
        # 1 input image channel, 6 output channels, 5x5 square convolution
        # kernel
        self.conv1 = nn.Conv2d(1, 6, 5)
        self.conv2 = nn.Conv2d(6, 16, 5)
        # an affine operation: y = Wx + b
        self.fc1 = nn.Linear(16 * 5 * 5, 120)  # 5*5 from spatial dimension
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        # Max pooling over a (2, 2) window
        x = F.max_pool2d(F.relu(self.conv1(x)), (2, 2))
        # If the size is a square you can only specify a single number
        x = F.max_pool2d(F.relu(self.conv2(x)), 2)
        x = x.view(-1, self.num_flat_features(x))
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

    def num_flat_features(self, x):
        size = x.size()[1:]  # all dimensions except the batch dimension
        num_features = 1
        for s in size:
            num_features *= s
        return num_features
```

Load your data



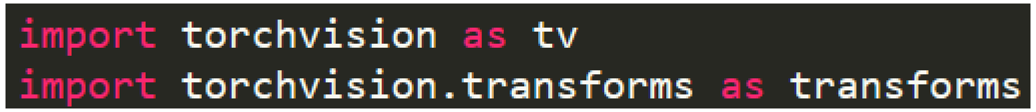
Generally, when you have to deal with image, text, audio or video data, you can use standard python packages that load data into a numpy array. Then you can convert this array into a `torch.*Tensor`.

- For images, packages such as Pillow, OpenCV are useful
- For audio, packages such as scipy and librosa
- For text, either raw Python or Cython based loading, or NLTK and SpaCy are useful

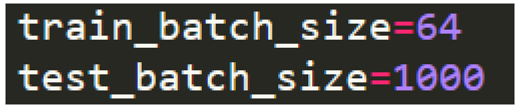
Specifically for vision, we have created a package called `torchvision`, that has data loaders for common datasets such as Imagenet, CIFAR10, MNIST, etc. and data transformers for images, viz., `torchvision.datasets` and `torch.utils.data.DataLoader`.

Load your data

- Example: load and normalize MNIST



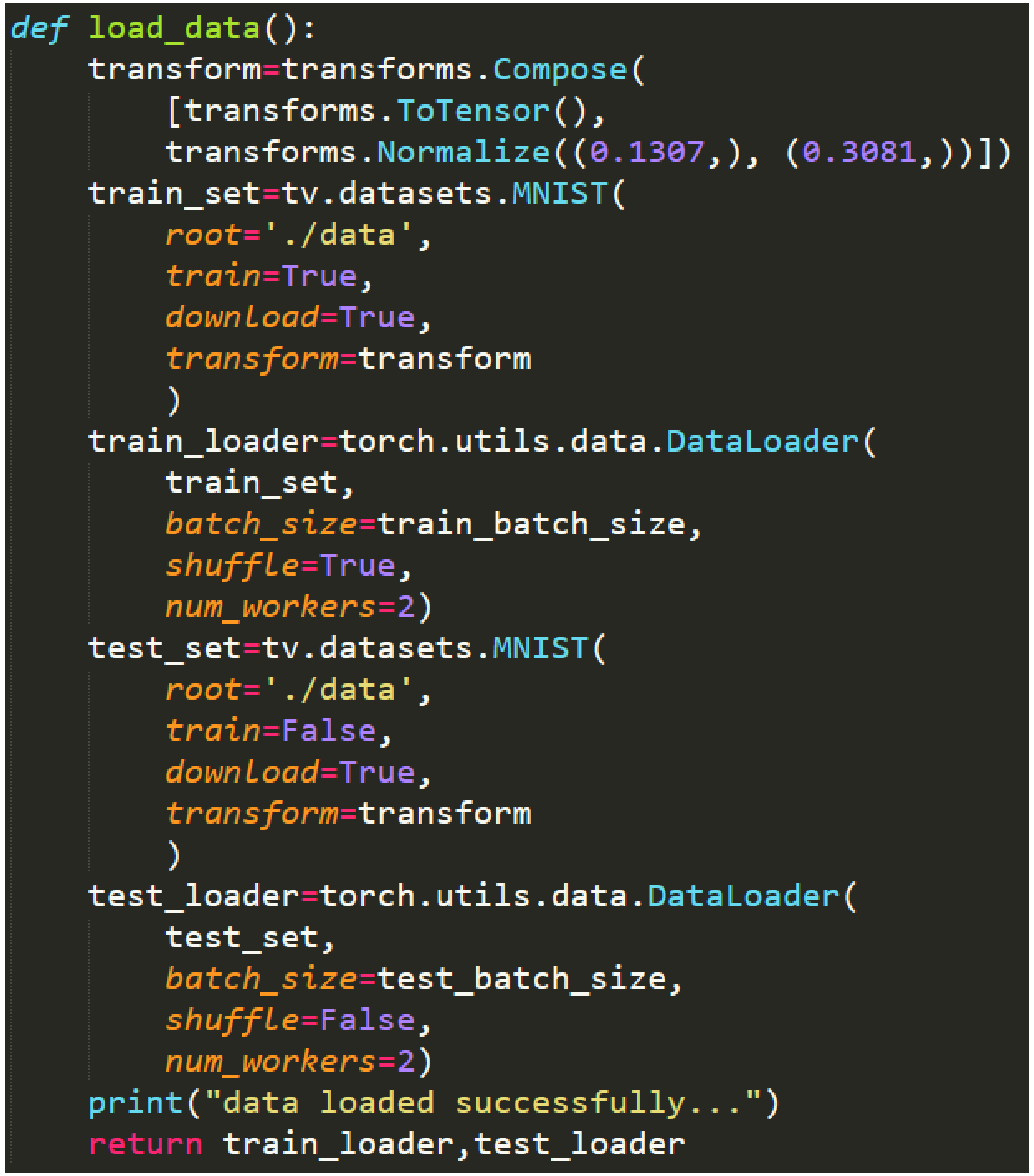
```python
import torchvision as tv
import torchvision.transforms as transforms
```



```python
train_batch_size=64
test_batch_size=1000
```

- Other datasets:

  https://pytorch.org/vision/0.8/datasets.html#



```python
def load_data():
    transform=transforms.Compose(
        [transforms.ToTensor(),
        transforms.Normalize((0.1307,), (0.3081,))])
    train_set=tv.datasets.MNIST(
        root='./data',
        train=True,
        download=True,
        transform=transform
        )
    train_loader=torch.utils.data.DataLoader(
        train_set,
        batch_size=train_batch_size,
        shuffle=True,
        num_workers=2)
    test_set=tv.datasets.MNIST(
        root='./data',
        train=False,
        download=True,
        transform=transform
        )
    test_loader=torch.utils.data.DataLoader(
        test_set,
        batch_size=test_batch_size,
        shuffle=False,
        num_workers=2)
    print("data loaded successfully...")
    return train_loader,test_loader
```

Training

Mini Batch Gradient Descent

Steps in one epoch:

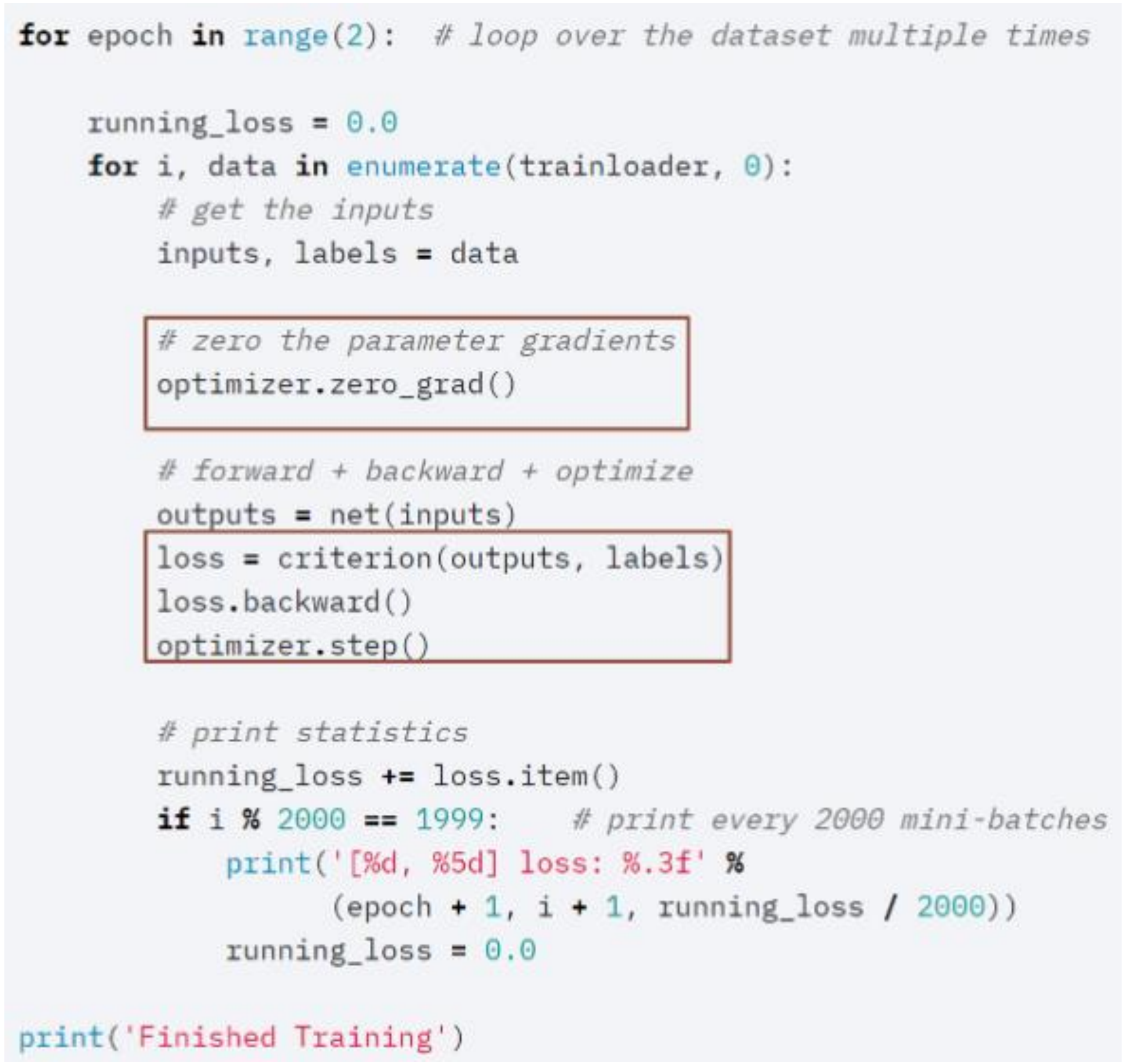
1. Pick a mini-batch

2. Feed it to Neural Network

3. Calculate the mean gradient of the mini-batch

4. Use the mean gradient we calculated in step 3 to update the weights

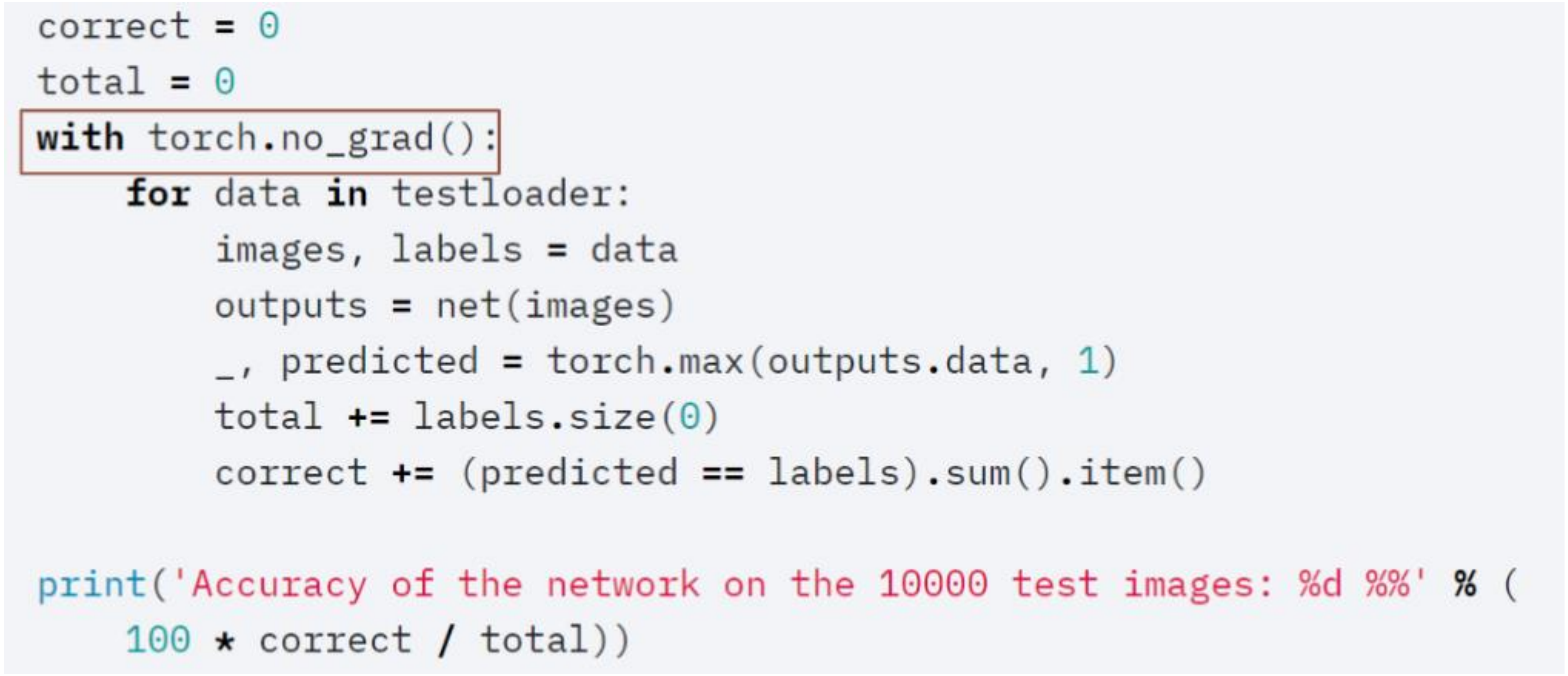
5. Repeat steps 1–4 for the mini-batches we created



```
for epoch in range(2):  # loop over the dataset multiple times

    running_loss = 0.0
    for i, data in enumerate(trainloader, 0):
        # get the inputs
        inputs, labels = data

        # zero the parameter gradients
        optimizer.zero_grad()

        # forward + backward + optimize
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        # print statistics
        running_loss += loss.item()
        if i % 2000 == 1999:    # print every 2000 mini-batches
            print('[%d, %5d] loss: %.3f' %
                  (epoch + 1, i + 1, running_loss / 2000))
            running_loss = 0.0

print('Finished Training')
```

https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a

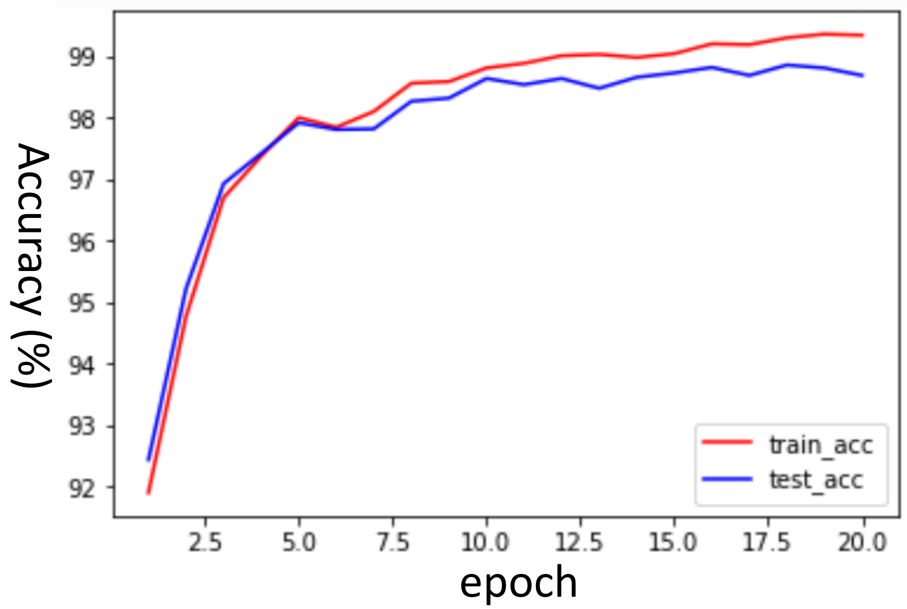
Testing



```
correct = 0
total = 0
with torch.no_grad():
    for data in testloader:
        images, labels = data
        outputs = net(images)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

print('Accuracy of the network on the 10000 test images: %d %%' % (
    100 * correct / total))
```

- The result depends largely on hyperparameters and so on
  - You must report your settings in detail
    - Number of epochs, loss function, optimizer and its settings, regularization, weight initialization……
- Randomness of results
  - Different runs may get different accuracies.
  - You should run a certain setting for at least 5 times, and use the mean and std of your training/testing accuracy to evaluate its performance.
  - In your report, there should be a table listing the performance of a certain setting as
    - [best acc among 5 runs, mean acc of 5 runs, std of acc among 5 runs].

- Expected curves
  - Epoch-accuracy curve
  - Draw curves for training/testing on the same plot



  - You only need one plot for one setting (pick any one from your 5 runs)

Q
A