

Introduction to Deep Learning (CS474)

Lecture 1

Outline

- Brief overview of Pytorch.
- Popular Deep Learning Frameworks.
- Installing Pytorch in your machine.
- Introduction to Google Colab.
- Getting started with Google Colab.

Brief overview of 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.

Popular Deep Learning Frameworks

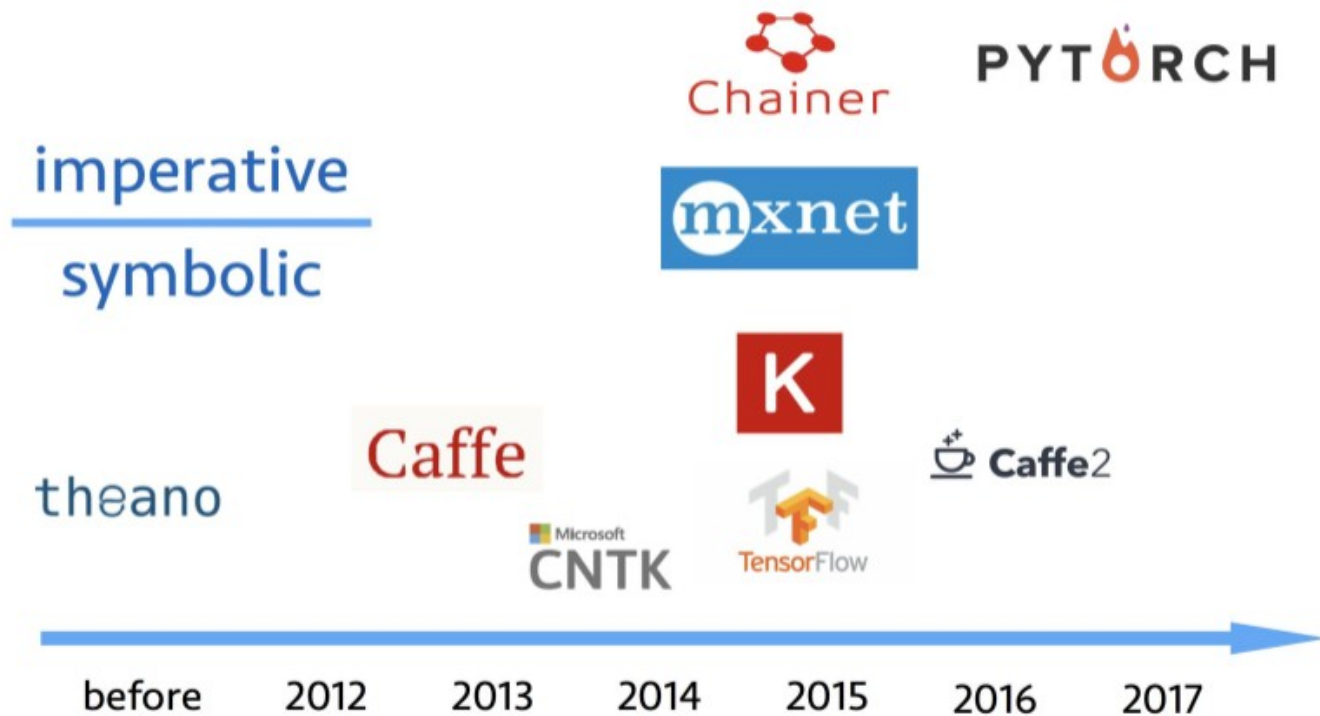


Image credit: Gluon: new MXNet interface to accelerate research

Popular Deep Learning Frameworks

Imperative: Imperative-style programs perform computation as you run them

```
import numpy as np
a = np.ones(10)
b = np.ones(10) * 2
c = b * a
d = c + 1
```

Symbolic: define the function first, then compile them

```
A = Variable('A')
B = Variable('B')
C = B * A
D = C + Constant(1)
# compiles the function
f = compile(D)
d = f(A=np.ones(10), B=np.ones(10)*2)
```

Image credit: Gluon: new MXNet interface to accelerate research

Installing Pytorch in your machine

Let's start locally!

- I am assuming that
 - you have installed python 3.5.
 - Your OS is Ubuntu!
- You may look into the following figure along with the following URL.

<https://pytorch.org/get-started/locally/>

PyTorch Build	Stable (1.6.0)		Preview (Nightly)	
Your OS	Linux		Mac	Windows
Package	Conda	Pip	LibTorch	Source
Language	Python		C++ / Java	
CUDA	9.2	10.1	10.2	None

Installing Pytorch in your machine

- To install the PyTorch binaries, you will need to use one of two supported package managers: Anaconda or pip.
- To install Anaconda, you will use the command-line installer. Right-click on the 64-bit installer link, select Copy Link Location, and then use the following commands:

```
# The version of Anaconda may be different depending on when you are installing'  
curl -O https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh  
sh Miniconda3-latest-Linux-x86_64.sh  
# and follow the prompts. The defaults are generally good.'
```

```
You may have to open a new terminal or re-source your ~/.bashrc to get access to the conda  
command.
```

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

Installing Pytorch in your machine

- Verification

```
(base) rupam-iiitbh@rupamiiitbh-HP-348-G4:~$ mkdir pytorch_projects
(base) rupam-iiitbh@rupamiiitbh-HP-348-G4:~$ cd pytorch_projects/
(base) rupam-iiitbh@rupamiiitbh-HP-348-G4:~/pytorch_projects$ mkdir basic
(base) rupam-iiitbh@rupamiiitbh-HP-348-G4:~/pytorch_projects$ cd basic/
(base) rupam-iiitbh@rupamiiitbh-HP-348-G4:~/pytorch_projects/basic$ gedit test_torch.py
(base) rupam-iiitbh@rupamiiitbh-HP-348-G4:~/pytorch_projects/basic$ python3 test_torch.py
tensor([[0.4025, 0.9654, 0.7734],
        [0.2092, 0.0662, 0.0105],
        [0.1729, 0.1292, 0.8992],
        [0.3075, 0.3552, 0.5083],
        [0.3234, 0.7774, 0.3524]])
(base) rupam-iiitbh@rupamiiitbh-HP-348-G4:~/pytorch_projects/basic$
```

- **test_torch.py** contain following lines of code which will construct a randomly initialized tensor.

```
from __future__ import print_function
import torch
x = torch.rand(5, 3)
print(x)
```


Installing Pytorch in your machine

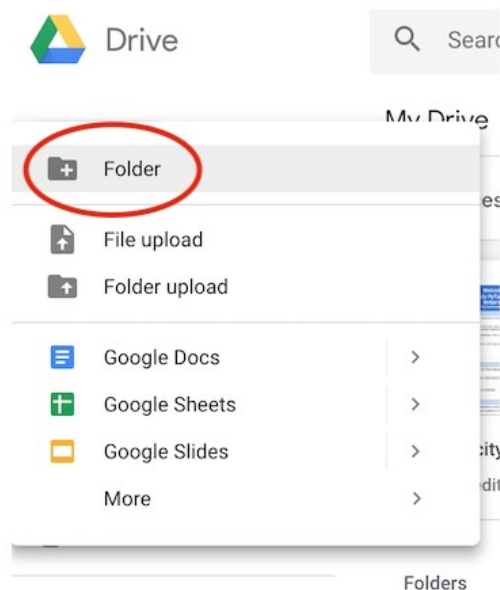
- Now, we have installed pytorch correctly.
 - This demonstration don't showcase the GPU utilization.
 - PyTorch utilize CUDA for fast processing which we have not demonstrated till now.
 - The NVIDIA CUDA Toolkit provides a development environment for creating high performance GPU-accelerated applications.

Introduction to Google Colab.

- Free GPUs for Everyone!
- **Google Colaboratory** is a free online cloud-based Jupyter notebook environment that allows us to train our machine learning and deep learning models on CPUs, GPUs, and TPUs.
- Colab gives us **12 hours** of continuous execution time. After that, the whole virtual machine is cleared and we have to start again.

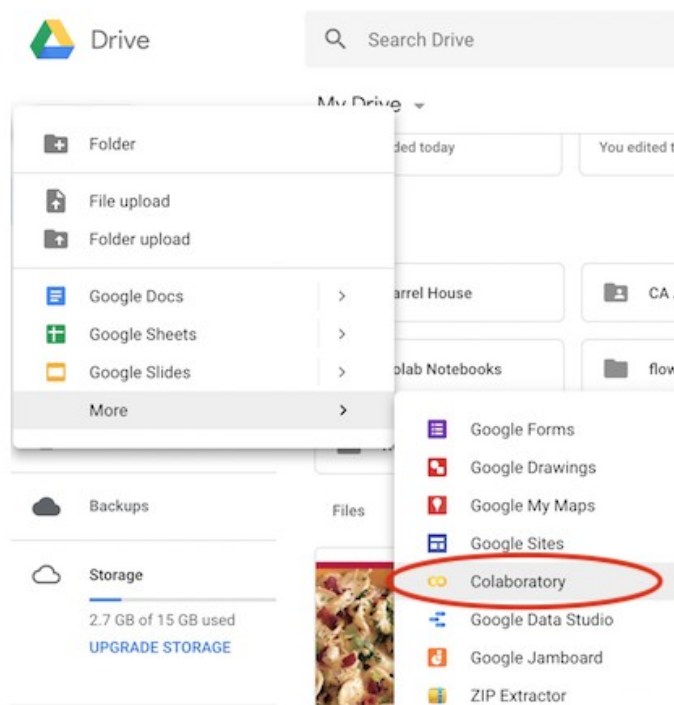
Getting started with Google Colab.

- Setting up your Google drive.
 - Create a folder for your Jupyter notebooks.



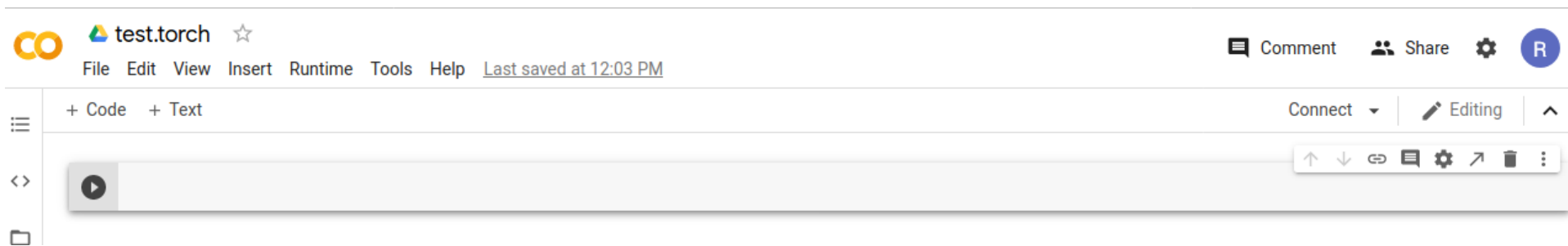
Getting started with Google Colab.

- Creation of a new **notebook** within that *folder*.



Getting started with Google Colab.

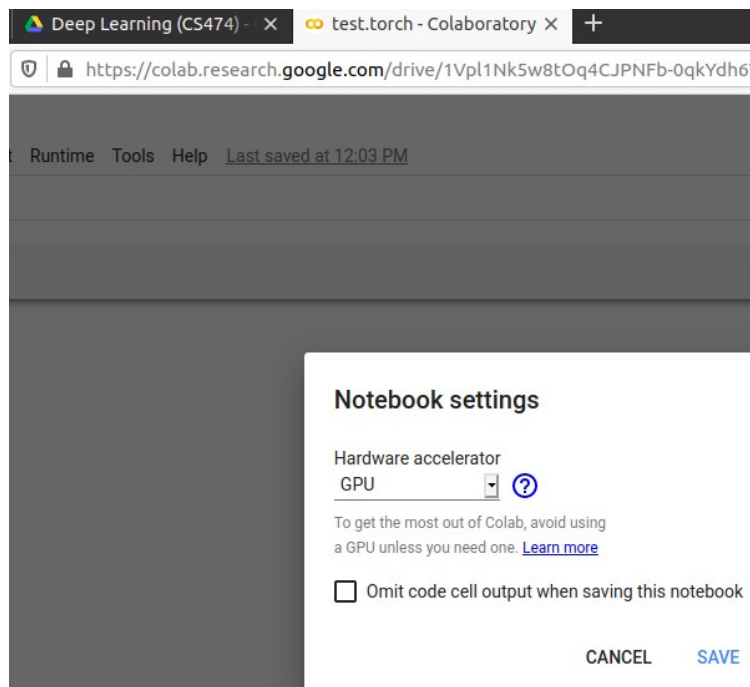
- Rename the notebook to “test_torch” present within that *folder*.
- Open that file in Google Collaboratory. You may observe the following things in your web browser.



Getting started with Google Colab.

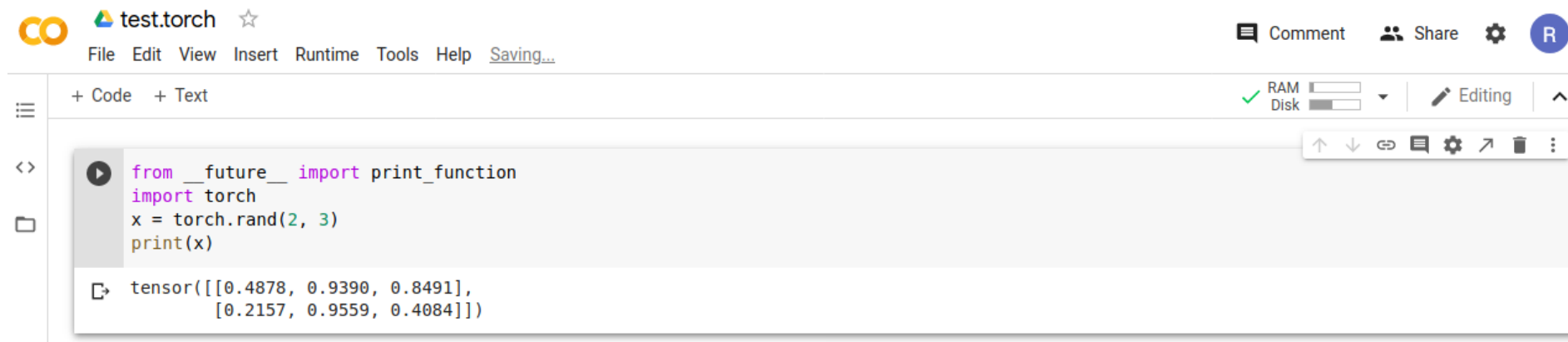
- Set up your GPU.

- Go to the “runtime” drop down menu, selecting “**change runtime type**” and selecting GPU in the hardware accelerator drop-down menu!



Getting started with Google Colab.

- Write your code within the *test_torch* notebook.



The screenshot shows a Google Colab notebook interface. At the top, the notebook is titled 'test.torch' with a star icon. Below the title is a menu bar with options: File, Edit, View, Insert, Runtime, Tools, Help, and a 'Saving...' status. On the right side of the menu bar are icons for Comment, Share, and a settings gear, followed by a circular profile icon with the letter 'R'. Below the menu bar, there are tabs for '+ Code' and '+ Text'. The main area of the notebook displays a code cell with the following Python code:

```
from __future__ import print_function
import torch
x = torch.rand(2, 3)
print(x)
```

Below the code cell, the output is shown: a tensor with two rows and three columns of random values.


```
tensor([[0.4878, 0.9390, 0.8491],
        [0.2157, 0.9559, 0.4084]])
```

On the right side of the notebook, there are status indicators for RAM and Disk usage, a green checkmark, and a dropdown menu. Below these are icons for undo, redo, link, comment, settings, and a trash can.

- Google *Colab* now supports native **Pytorch**.
- We run the the same code in Google colab which was executed in our *local* machine.

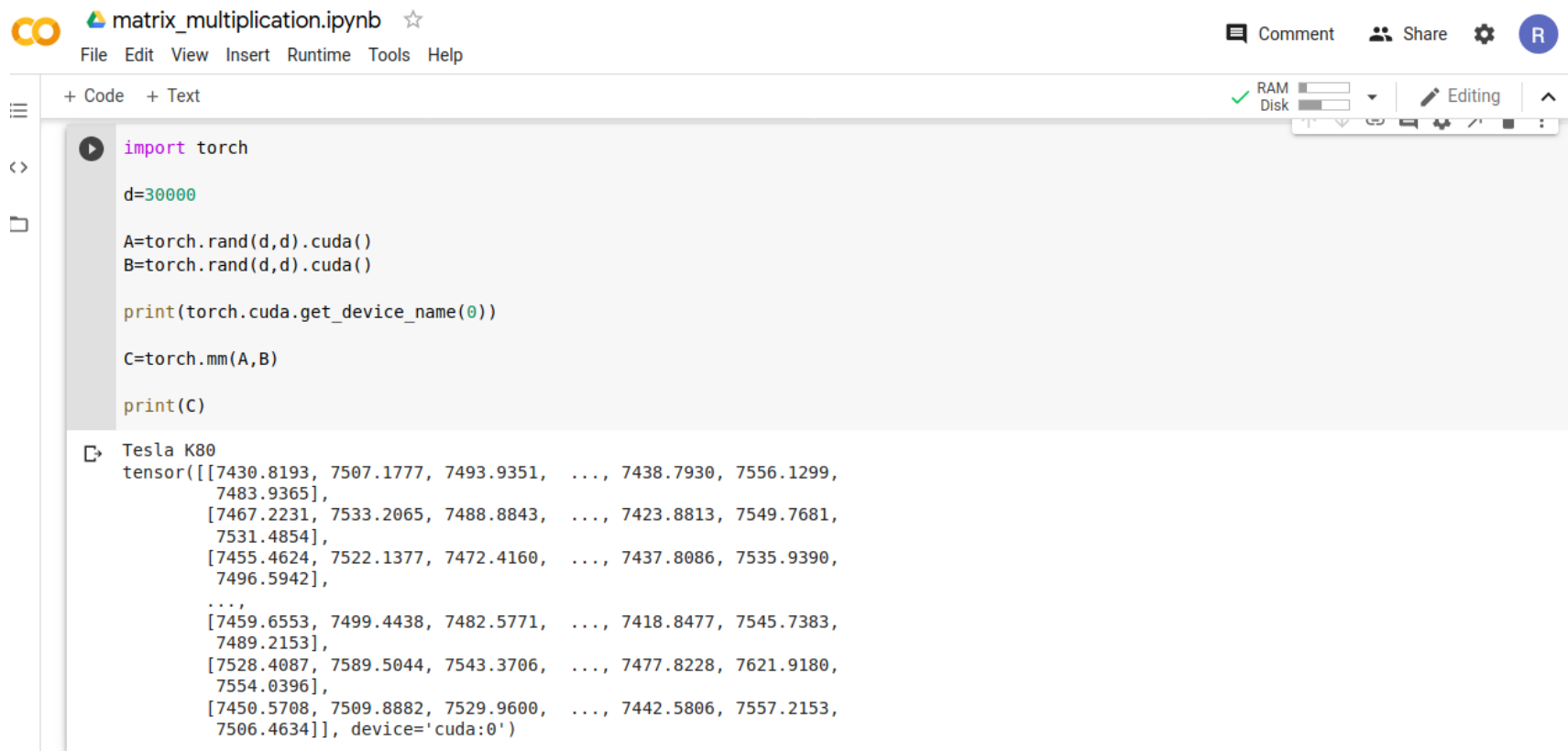
Getting started with Google Colab.

- Now, we would like to perform matrix multiplication through our **CPU** and **numpy**!

```
Open  ▾   matrix_multiplication_numpy.py  
~/pytorch_projects/basic  
  
import numpy as np  
  
d=30000  
  
A=np.random.rand(d,d).astype(np.float32)  
B=np.random.rand(d,d).astype(np.float32)  
  
C=A.dot(B)  
  
print(C)
```


Getting started with Google Colab.

- Now, we would like to perform matrix multiplication through GPU and torch!



The screenshot shows a Google Colab notebook interface. At the top, the title bar reads 'matrix_multiplication.ipynb' with a star icon. Below it is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the right side of the title bar are icons for 'Comment', 'Share', a settings gear, and a user profile icon labeled 'R'. The main area of the notebook is divided into two sections: a code editor and an output area. The code editor has a toolbar with '+ Code' and '+ Text' buttons. The code being executed is as follows:

```
import torch

d=30000

A=torch.rand(d,d).cuda()
B=torch.rand(d,d).cuda()

print(torch.cuda.get_device_name(0))

C=torch.mm(A,B)

print(C)
```

The output area shows the result of the execution. It starts with 'Tesla K80' indicating the GPU used. Below that is a large tensor of floating-point numbers, representing the result of the matrix multiplication. The tensor is displayed in a truncated format, showing the first and last few rows and columns.

```
tensor([[7430.8193, 7507.1777, 7493.9351, ..., 7438.7930, 7556.1299,
        7483.9365],
        [7467.2231, 7533.2065, 7488.8843, ..., 7423.8813, 7549.7681,
        7531.4854],
        [7455.4624, 7522.1377, 7472.4160, ..., 7437.8086, 7535.9390,
        7496.5942],
        ...,
        [7459.6553, 7499.4438, 7482.5771, ..., 7418.8477, 7545.7383,
        7489.2153],
        [7528.4087, 7589.5044, 7543.3706, ..., 7477.8228, 7621.9180,
        7554.0396],
        [7450.5708, 7509.8882, 7529.9600, ..., 7442.5806, 7557.2153,
        7506.4634]], device='cuda:0')
```

References

- All the contents present in the slides are taken from various online resources. These slides are used for *academic* purposes only.
 - <https://pytorch.org/>
 - <https://developer.nvidia.com/cuda-toolkit>