Tutorials for Google Colab and PyTorch

CIS 4930/5930 Spring 2026

FSU | FLORIDA STATE UNIVERSITY

Today's Outline

- Pre-requisite
- Google Colab and Alternatives
- PyTorch Basic
- CNN Example: Image Classification
- Presenting your model: Gradio and Streamlit

FSU

Pre-requisite

1. Basic python knowledge
2. Google account for Colab (OR PC/Mac with a GPU OR Other online platform)

- - Why you need a GPU for deep learning

Python Basic

- Python is a high-level, interpreted, general-purpose programming language known for its simplicity and readability. Unlike C++, Python is a scripting language that does not need to be compile.

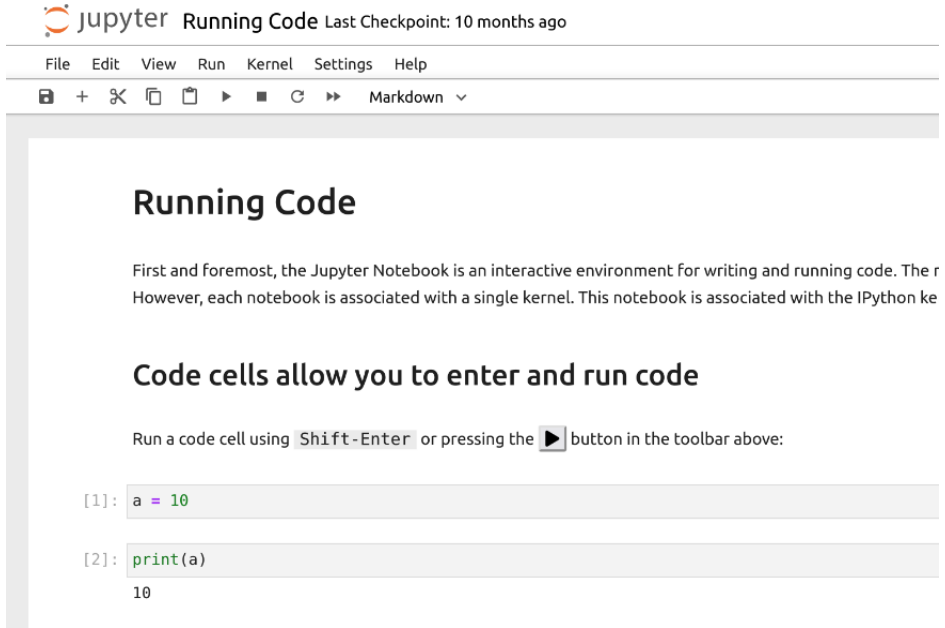
Run by executing file:
`*python main.py*`

```
import random

# Generate a random number between 1 and 10
random_number = random.randint(1, 10)

print("Random number:", random_number)
```

Run in console interactively:

```
>>> print("Hello, I'm Python!")
Hello, I'm Python!
```

Run in Jupyter notebook – like tools (In our class):



Python Basic (con't)

- In this class, we assume that you already know some of python usages. Like basic "if", "for", how to install and import package, etc.

Below are some free tutorials you can find online. if you are not familiar with python environment and python language, you may find they are useful.

Tutorial: How to install Python
windows: https://www.youtube.com/watch?v=TNAu6DvB9Ng
mac: https://www.youtube.com/watch?v=nhv82tvFfkM

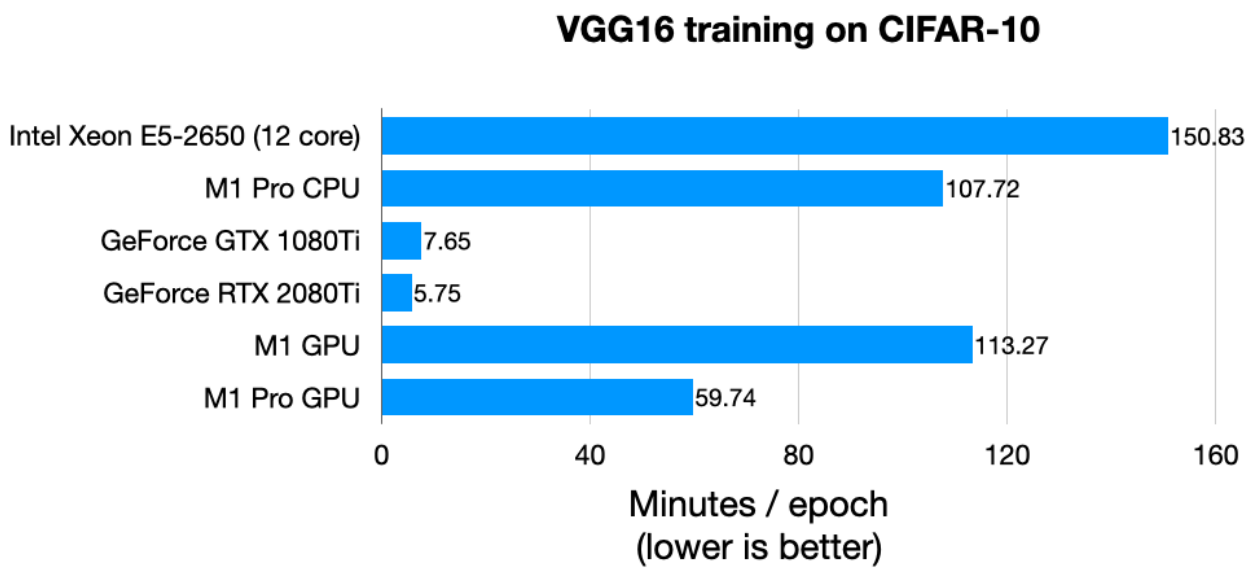
Tutorial: Create Anaconda Virtual Environment for Jupyter Notebook
windows: https://www.youtube.com/watch?v=sv0ca-6liM8
mac: https://www.youtube.com/watch?v=SF_w7LBV_Zo

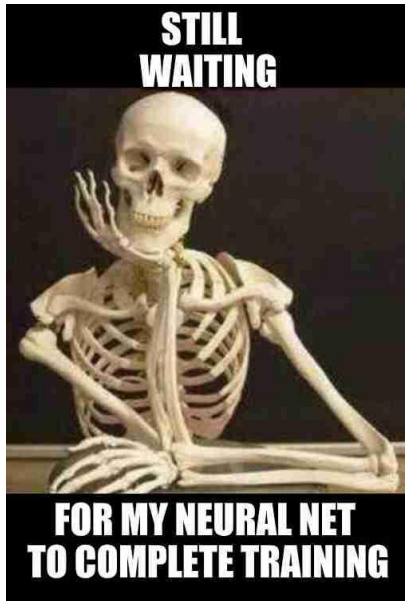
Tutorial (Python for beginners) (about 1 hour):
https://www.youtube.com/watch?v=kqtD5dpn9C8

A Machine with GPU

- In this class you need to have a machine with GPU in order to train your neural network.

Why you need a GPU?





A Machine with GPU (con't)

- You can use your local computer with Nvidia GPU. (For mac, you can use MPS with not promising performance)

Correct corresponding version:
Nvidia driver, CUDA toolkit, cuDNN libray

cuDNN Package [1]	CUDA Toolkit Version	Supports static linking? [2]	NVIDIA Driver Version for Linux	NVIDIA Driver Version f
cuDNN 9.5.0 for CUDA 12.x	› 12.6 › 12.5 › 12.4 › 12.3 › 12.2 › 12.1 › 12.0	Yes	>=525.60.13	>=527.41

https://docs.nvidia.com/deeplearning/cudnn/latest/reference/support-matrix.html

Command line to check if you already have CUDA toolkit installed

```
(base) C:\Users\User>nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2024 NVIDIA Corporation
Built on Tue_Feb_27_16:28:36_Pacific_Standard_Time_2024
Cuda compilation tools, release 12.4, V12.4.99
Build cuda_12.4.r12.4/compiler.33961263_0
```

How to install cuda toolkit on windows: https://wiki.cci.arts.ac.uk/books/how-to-guides/page/how-to-install-cuda-toolkit-on-your-personal-windows-pc
Accelerated PyTorch training on Mac: https://developer.apple.com/metal/pytorch/

Create Running Environment

- Each Python project can have its own unique running environment
- Recommended tool: Anaconda and Pip
  - Allows users to create and manage isolated environments easily
  - different projects can use different versions of Python and packages
    - Example: Numpy, Sklearn, PyTorch, Pandas, etc.
  - ensures consistent package versions and environment behavior across Windows, macOS, and Linux
  - Useful tutorials:
    - https://www.anaconda.com/docs/getting-started/getting-started
    - https://www.youtube.com/watch?v=1nDNhiRg4DY
- Other choice: uv
  - Documentations: https://docs.astral.sh/uv/

FSU

Google Colab and Alternatives

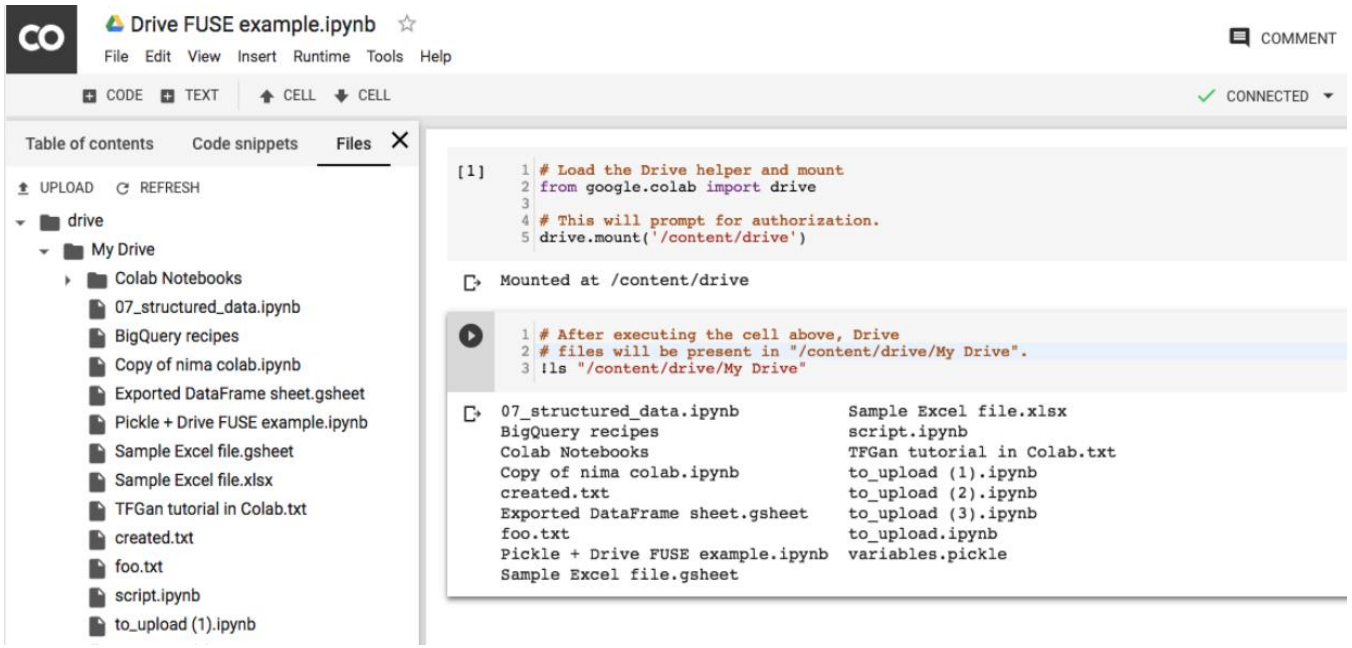
1. Getting start with Colab:
   - Runing Code, GPU Runtime, File Manipulation, Linux Commands on Colab, Tips
2. Alternatives:
   - PC(Recommended)/MAC with a GPU: Jupyter notebook with Conda virtual environment(optional)
   - Paid options: Colab Pro, Paperspace, etc.

Google Colab

- A ready-to-use python environment with free GPU hours

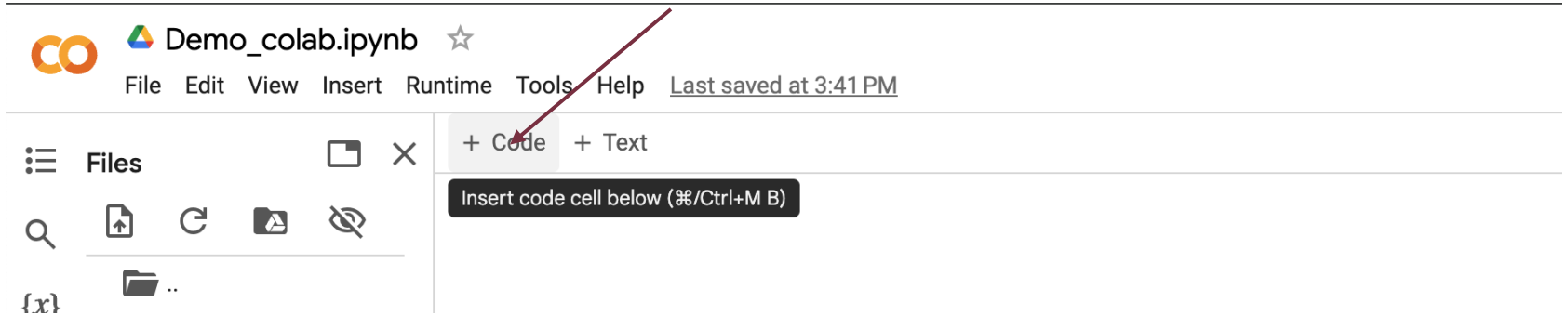
You can see it as an online version of Jupyter notebook. I recommend you to use this since it already comes with some common packages and python environment if you do not have a decent GPU on your local computer

https://colab.research.google.com/
You need a free google account in order to use it.

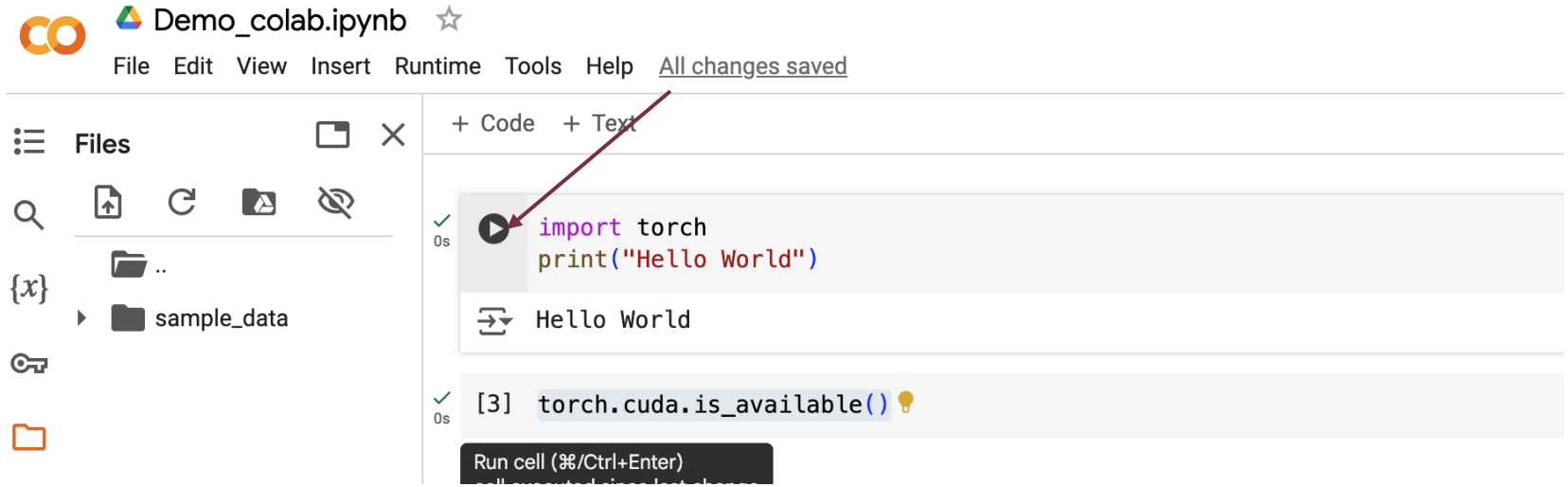


Add and Run Code

Click "+Code" to add python code
OR
Click "+Text" to add plain text(markdown)

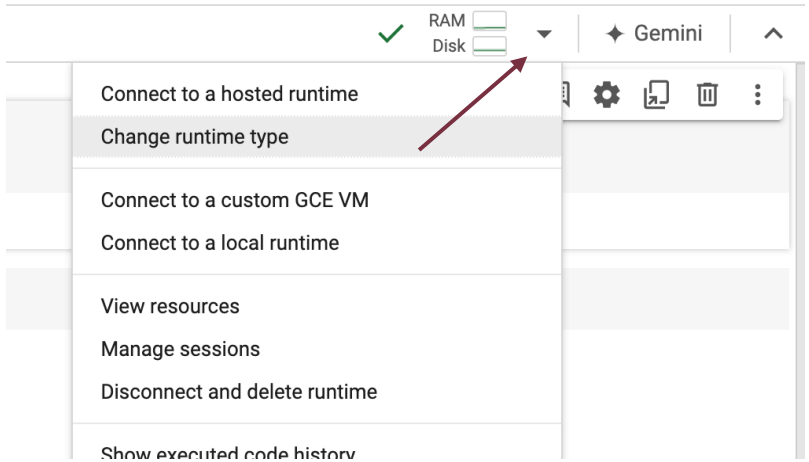


Click the run button to run the code
OR
You can select the "Runtime" in the menu to run multiple blocks of code

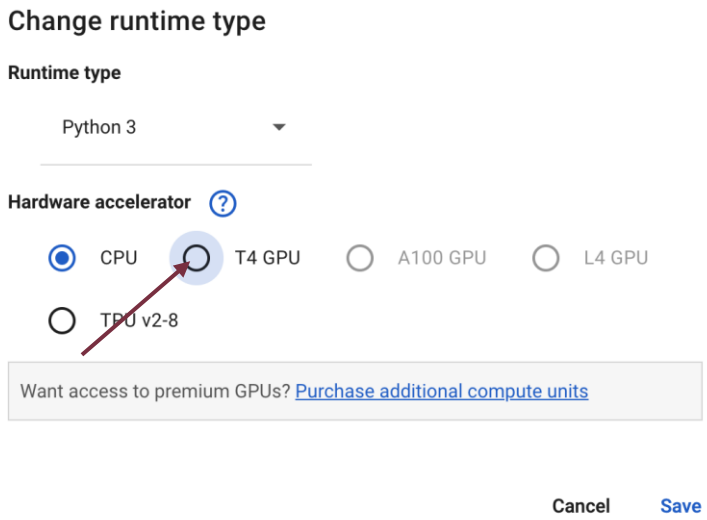


Select GPU environment and Verify

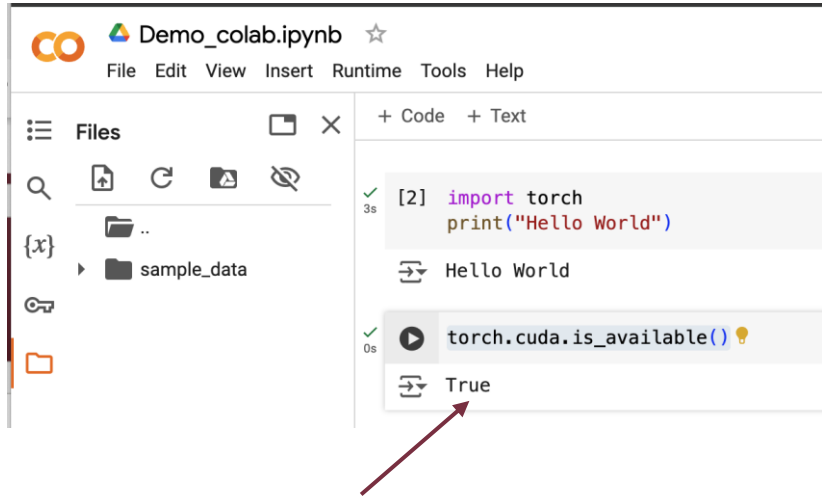
Select the button next to resource monitor icon and choose "Change runtime type"



Choose "T4 GPU" which is free to use and save



You can run the `torch.cuda.is_available() again to verify your configuration

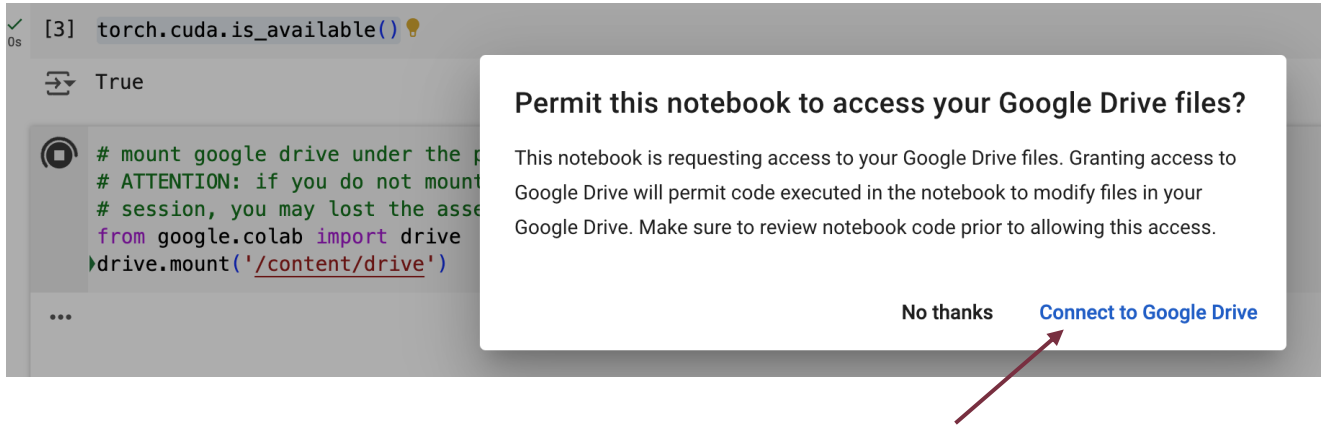


Mount Google Drive

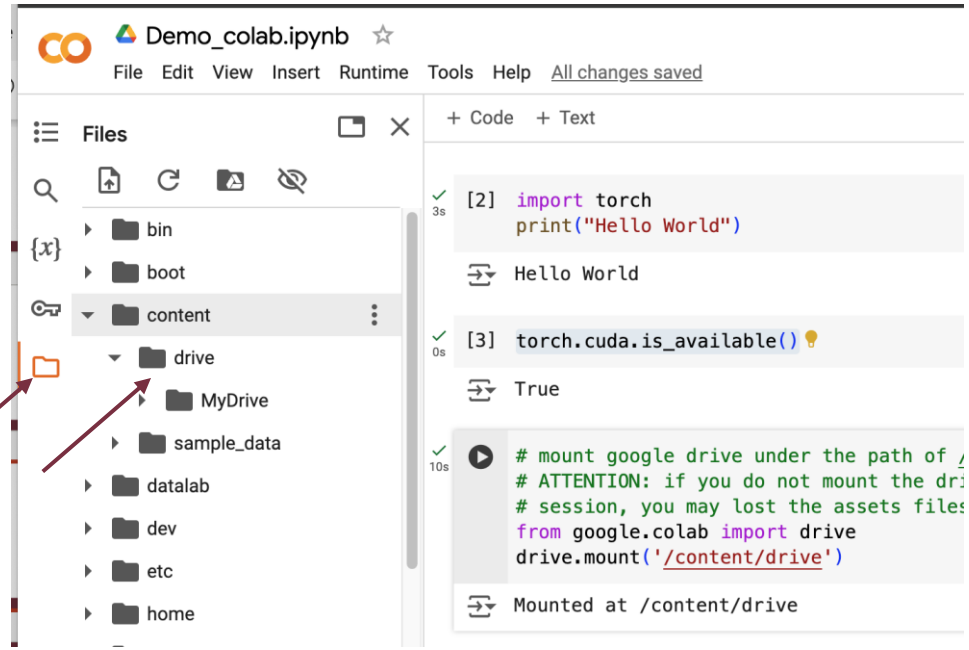
ALERT: the files in your temporary session will be lost after you restart your notebook

Use the code below to mount your google drive(15GB free) and grant the permission

```
from google.colab import drive
drive.mount('/content/drive')
```



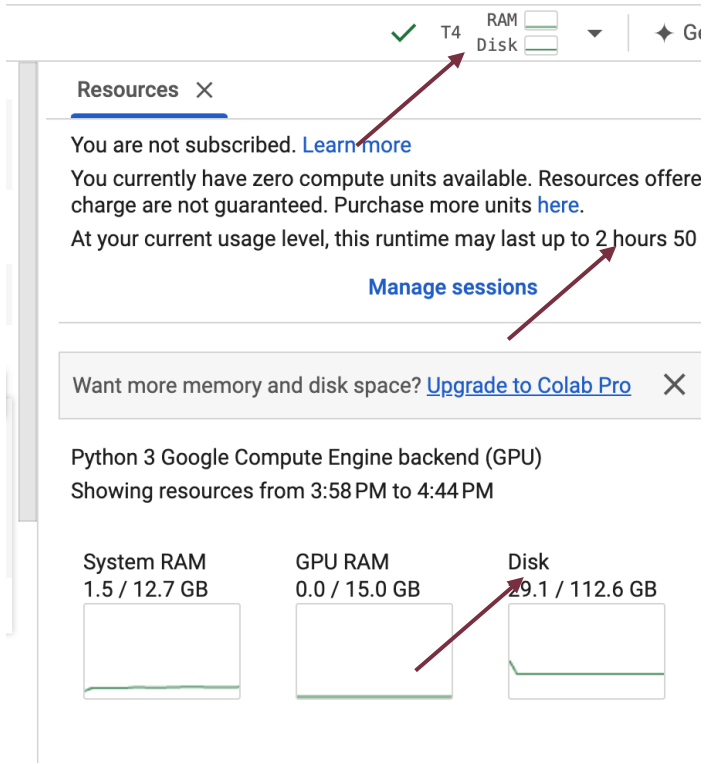
You can find the mounted folder in the side bar, where you can upload your files



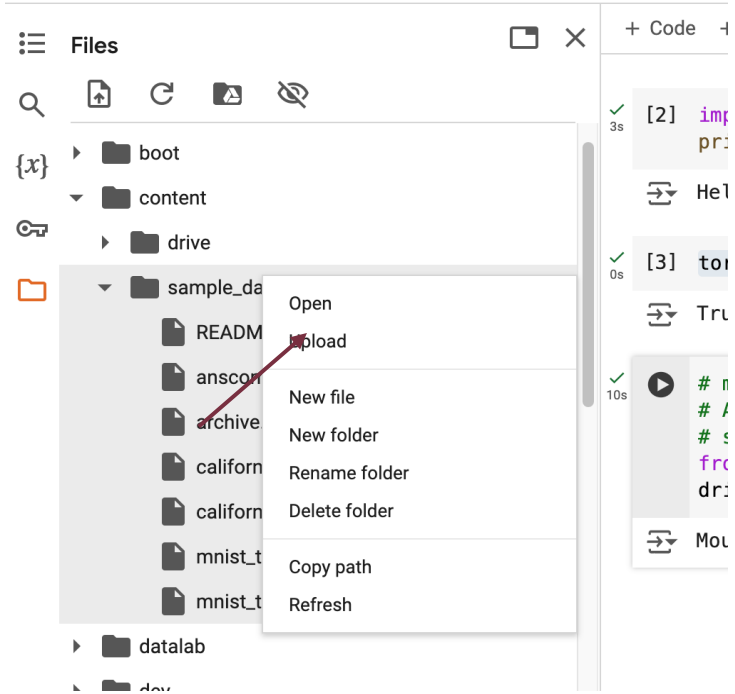
Upload or Download dataset

If you have a large dataset, you can also upload or download it directly to your temporary session

View your usage here (Notice: you do have a time limitation to use the GPU session, after that, you will lost all your temporary data)



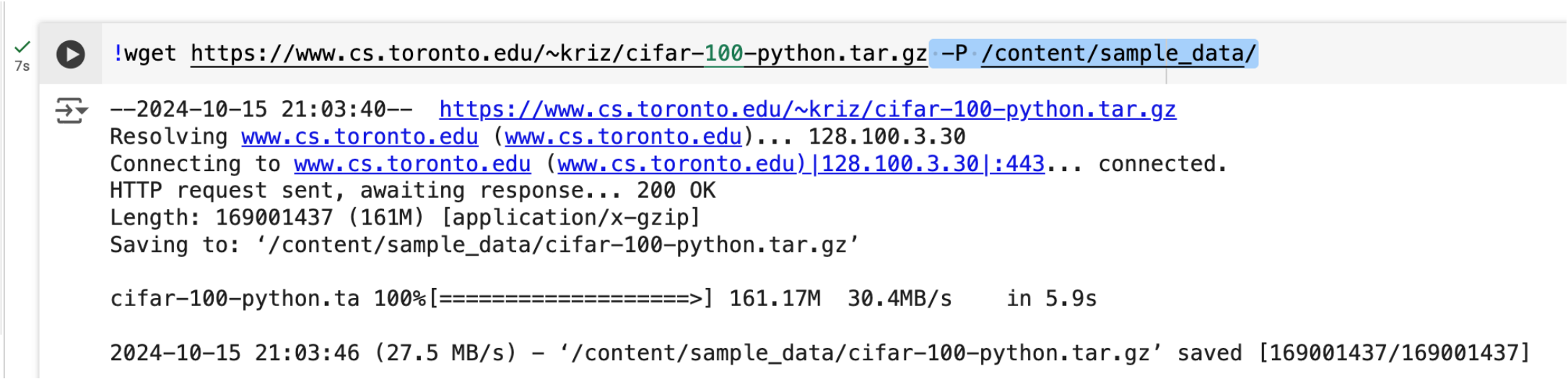
Right click the folder and select upload



Upload or Download dataset

- Or, you can download your dataset from URL to your session.
- Google Colab supports linux commands in the code block with a suffix "!" (use "%cd" to change directory), for example, use "!wget" to download files

!wget https://www.cs.toronto.edu/~kriz/cifar-100-python.tar.gz -P /content/sample_data/

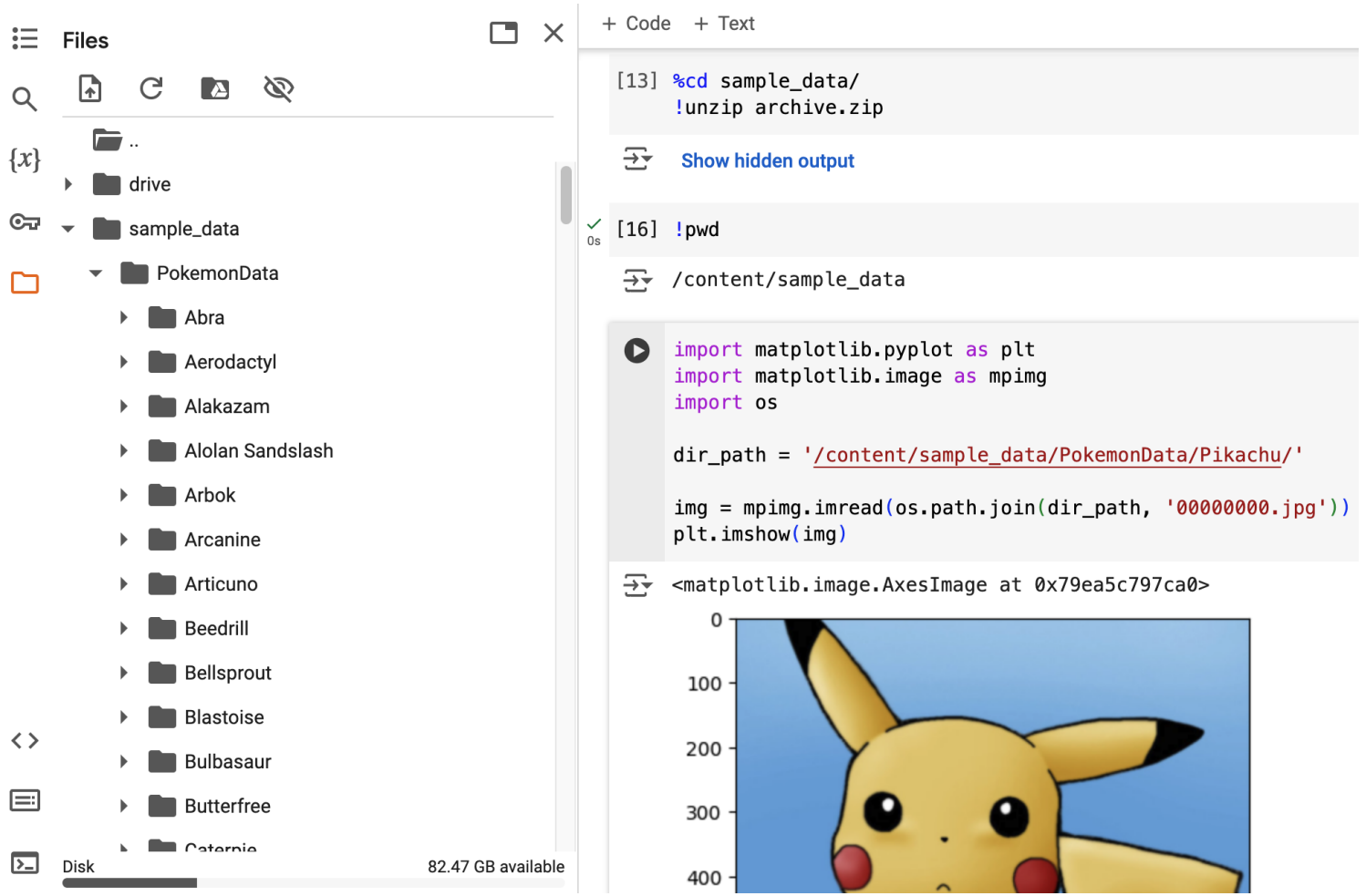


Read files

Use "%cd" enter into the folder contains your dataset,

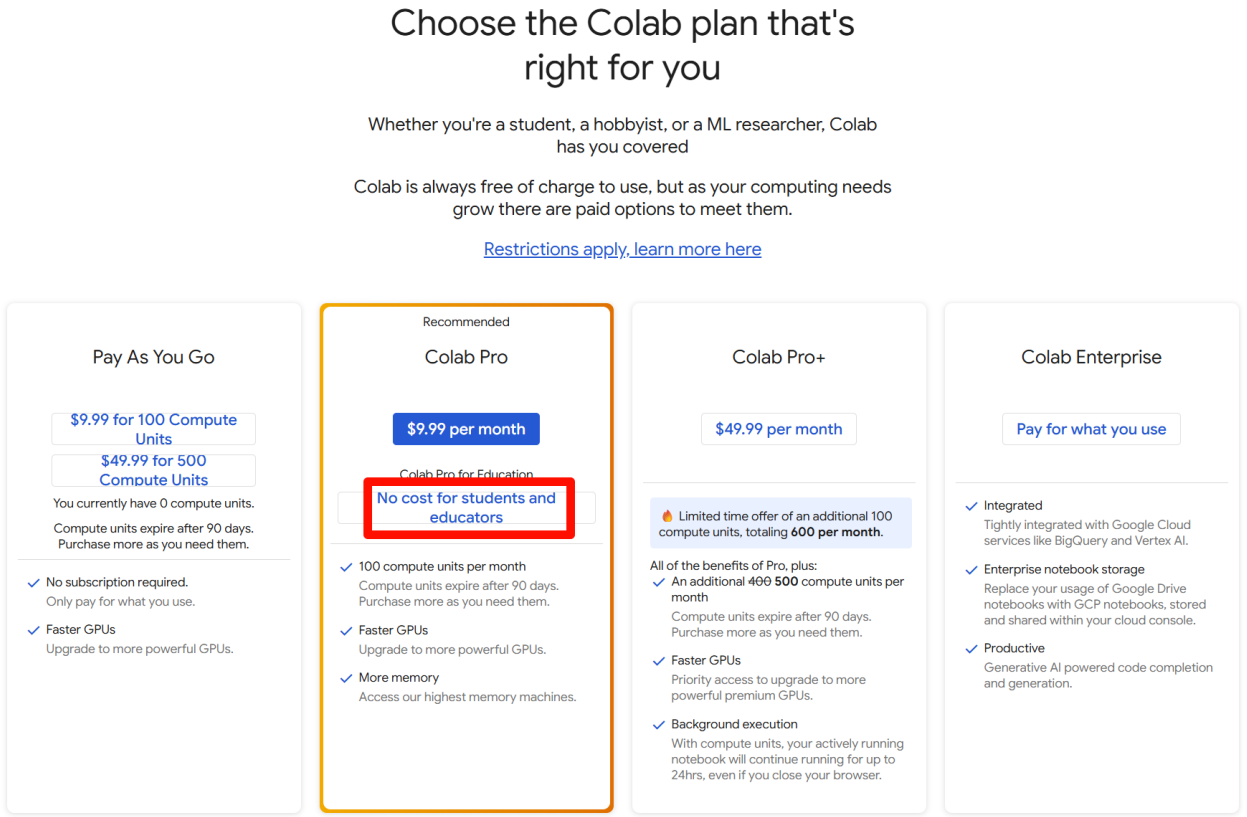
Use "!unzip" to unzip the file

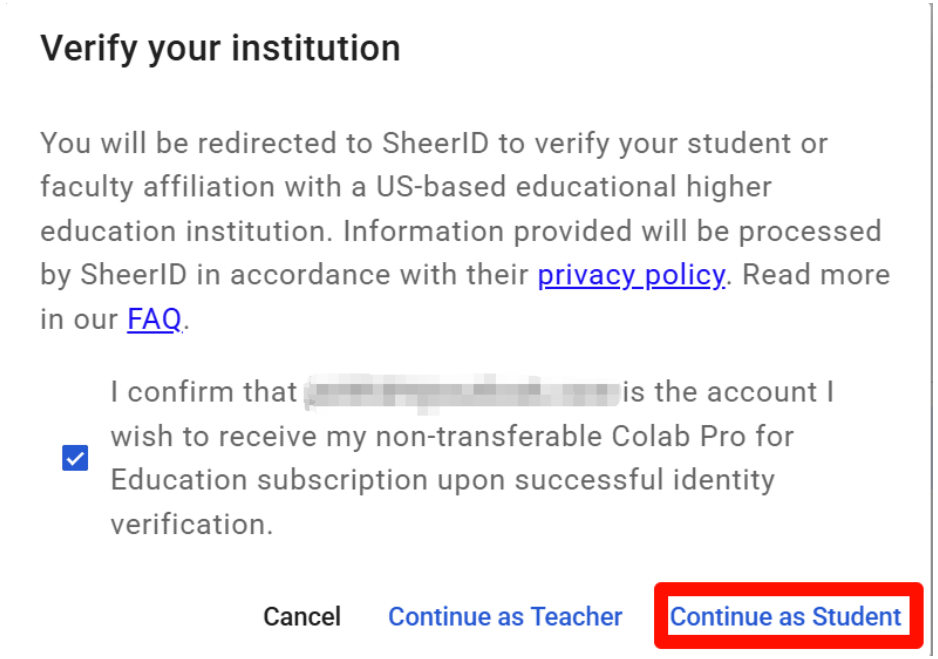
Use matplotlib to read and show image



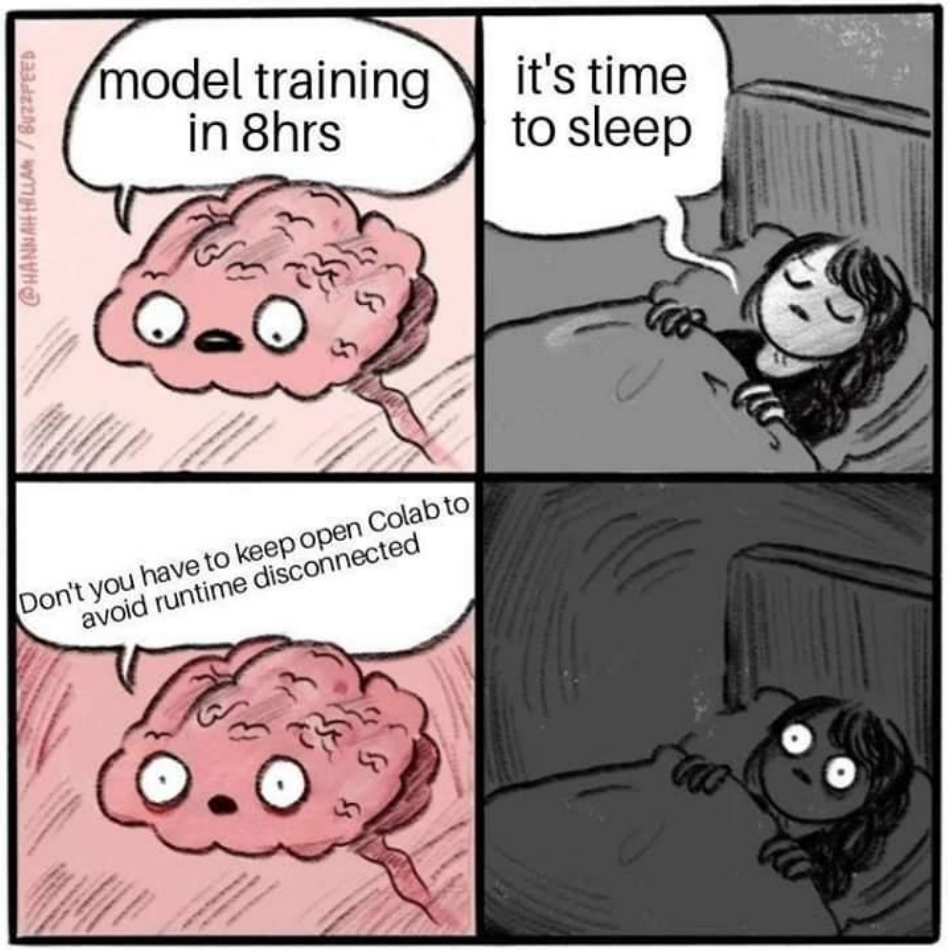
Getting Free one-year Pro Subscription

- Visit https://colab.research.google.com/signup





Attention:



- Google Colab notebooks have an idle timeout of 90 minutes and absolute timeout of 12 hours.
- This means, if user does not interact with his Google Colab notebook for more than 90 minutes, its instance is automatically terminated.
- Maximum lifetime of a Colab instance:12 hours.
- To keep your colab active: try https://stackoverflow.com/questions/57113226/how-can-i-prevent-google-colab-from-disconnecting
- There is a cooldown period, so you may have to use multiple free accounts

Alternatives

- Paperspace (Plan or on-demand instance): for example, free plan or $8 month for faster free GPU: https://www.paperspace.com
- VastAI - Similar to above: https://vast.ai
- Google Cloud, PyTorch lighting, or AWS, etc...
- Notice: If you have a good gaming PC that have RTX 4090, etc. You can always use your own PC for training. (For environment setup, please refer to the pre-requisite part)

FSU

PyTorch Basic

1. Introduction:
   - Basic Concepts, Build your network, Pipelines for training/testing/saving/loading model
2. PyTorch Documentation

PyTorch Introduction

- https://thenewstack.io/the-ultimate-guide-to-machine-learning-frameworks/

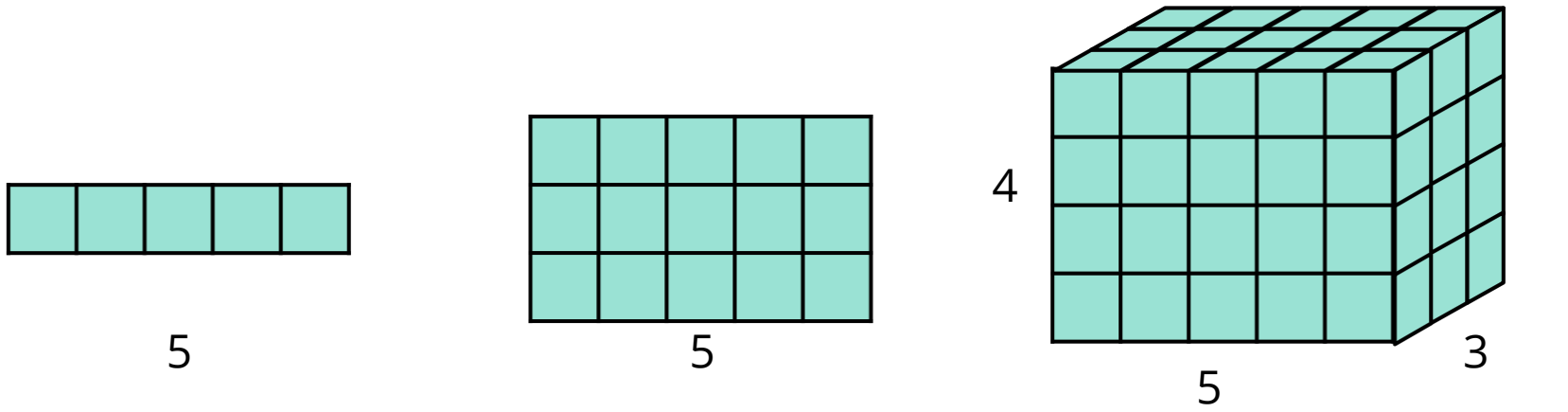
A popular machine learning framework

- For training machine learning model
- Developed at Facebook AI and Research lab
- Implementing a neural network in PyTorch is simpler and intuitive than other frameworks
- **Converting NumPy objects to tensors is natively integrated with PyTorch's core data structures**

```
✓ 8 秒  [1]  import torch
```

Key Concepts:

Tensor: A tensor is a mathematical object that generalizes scalars, vectors, and matrices into higher-dimensional spaces.



1-D tensor
e.g. audio

2-D tensor
e.g. black&white images

3-D tensor
e.g. RGB images

(5,) -> dim0

(3, 5) -> dim0, dim1

(4, 5, 3) -> dim0, dim1, dim2

Create tensors

```
tensor_0 = torch.zeros([2,3])
tensor_0
```

```
tensor([[0., 0., 0.],
        [0., 0., 0.]])
```

```
tensor_1 = torch.tensor([[0,1], [1,0]])
tensor_1
```

```
tensor([[0, 1],
        [1, 0]])
```

Check tensor's shape: .shape

```
tensor_1.shape
```

```
torch.Size([2, 2])
```

Key Concepts:

Tensor operations example:

Transpose:

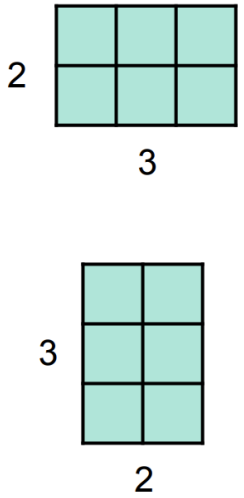
```
tensor_0.shape
```

```
torch.Size([2, 3])
```

```
tensor_0 = tensor_0.transpose(0, 1)
```

```
tensor_0.shape
```

```
torch.Size([3, 2])
```



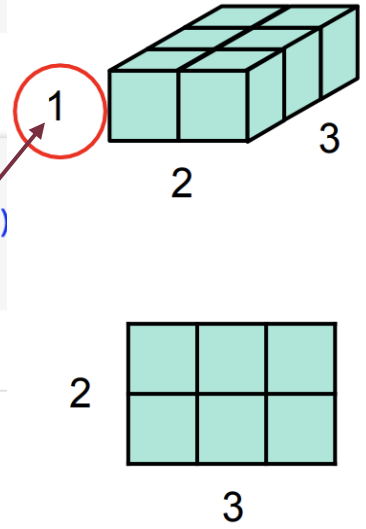
Remove Dim:

```
tensor3.shape
```

```
torch.Size([1, 2, 3])
```

```
# remove first dim
tensor3 = tensor3.squeeze(0)
tensor3.shape
```

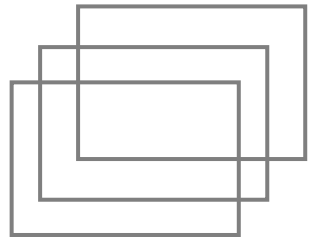
```
torch.Size([2, 3])
```



Expand Dim:

```
tensor3 = tensor3.unsqueeze(0)
tensor3.shape
```

```
torch.Size([1, 2, 3])
```



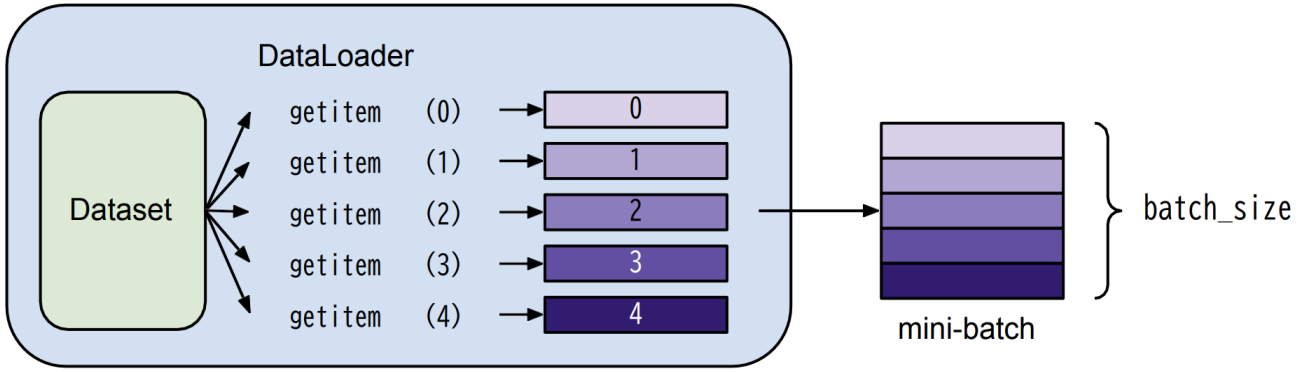
ML Steps in PyTorch


```
# Prepare data
xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=np.float32)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=np.float32)

# Convert numpy arrays to torch tensors
xs = torch.tensor(xs).view(-1, 1)  # Reshape to (n_samples, n_features)
ys = torch.tensor(ys).view(-1, 1)

print(xs)
print(ys)
```

```
tensor([[-1.],
        [ 0.],
        [ 1.],
        [ 2.],
        [ 3.],
        [ 4.]])
tensor([[-3.],
        [-1.],
        [ 1.],
        [ 3.],
        [ 5.],
        [ 7.]])
```

In a real example, you may use the code below to load data:

```
dataset = MyDataset('data.csv')
dataloader = DataLoader(dataset, batch_size=4, shuffle=True)
```



ML Steps in PyTorch

nn.Linear (fully connected layer)

```
# Define the model
class SimpleModel(nn.Module):
    def __init__(self):
        super(SimpleModel, self).__init__()
        self.linear = nn.Linear(1, 1)  # 1 input, 1 output

    def forward(self, x):
        return self.linear(x)

model = SimpleModel()
```

Under the hood, you may define your own layer manually

```
class SimpleLinearLayer(nn.Module):
    def __init__(self, in_features, out_features):
        super(SimpleLinearLayer, self).__init__()
        self.in_features = in_features
        self.out_features = out_features

        # Initialize weights and bias
        self.weights = nn.Parameter(torch.randn(out_features, in_features) * 0.01)  # Small random weights
        self.bias = nn.Parameter(torch.zeros(out_features))  # Bias initialized to zeros

    def forward(self, x):
        return x @ self.weights.t() + self.bias
```

Output = (Weight * Input) + Bias
Y = AX + B (Linear)

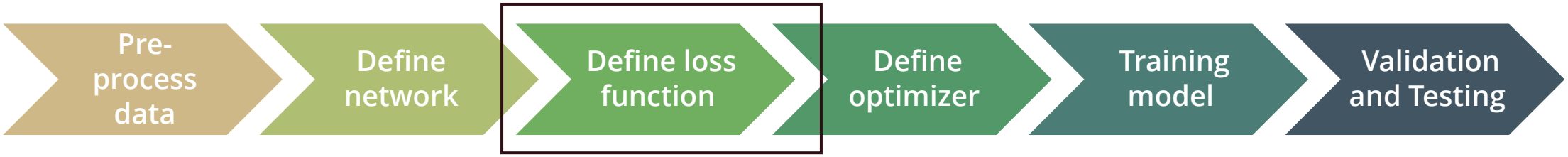
ML Steps in PyTorch

More layers in a container

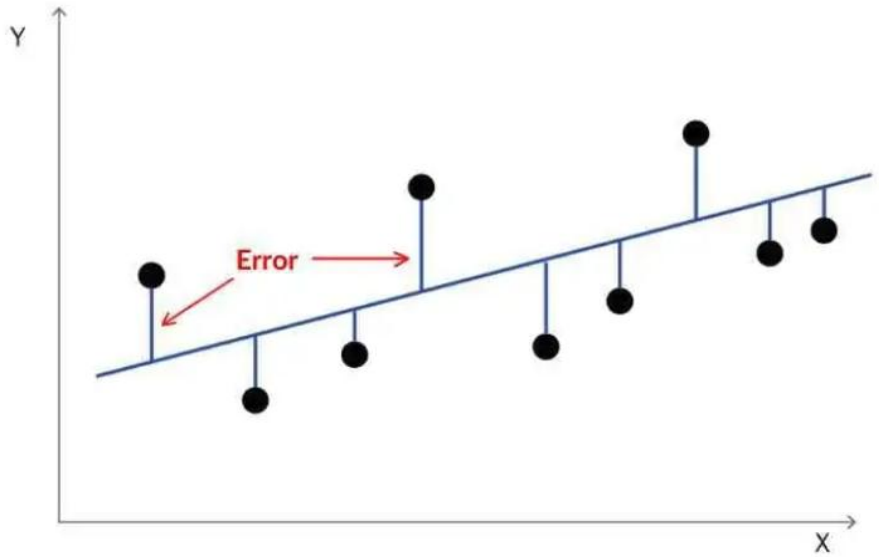
```
# Define the model
class SimpleModel2(nn.Module):
    def __init__(self):
        super(SimpleModel2, self).__init__()
        self.net = nn.Sequential(
            nn.Linear(1, 32), # input layer, 1 input, 32 neurons (32 features)
            nn.Sigmoid(), # hidden layer, sigmoid function
            nn.Linear(32, 1) # output layer, output 1
        )

    def forward(self, x):
        return self.net(x)
```

Whole Example: https://colab.research.google.com/drive/1_ru4CdNgz_MrtiPs5qUOaRoW_5q1R6cl?usp=sharing

ML Steps in PyTorch




```
# Define loss
criterion = nn.MSELoss()
```

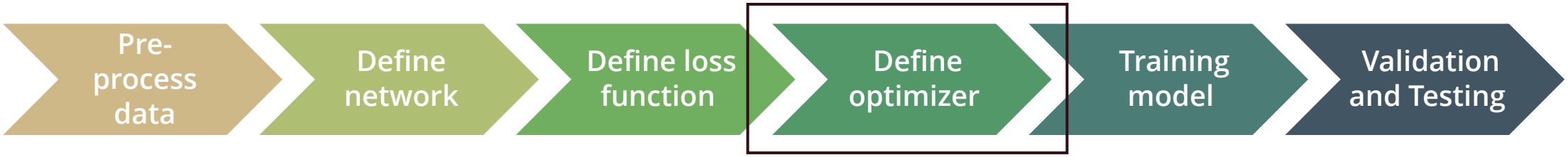


Under the hood, it just something like this:

```
def loss(predictions, targets):
    return ((predictions - targets) ** 2).mean()
```

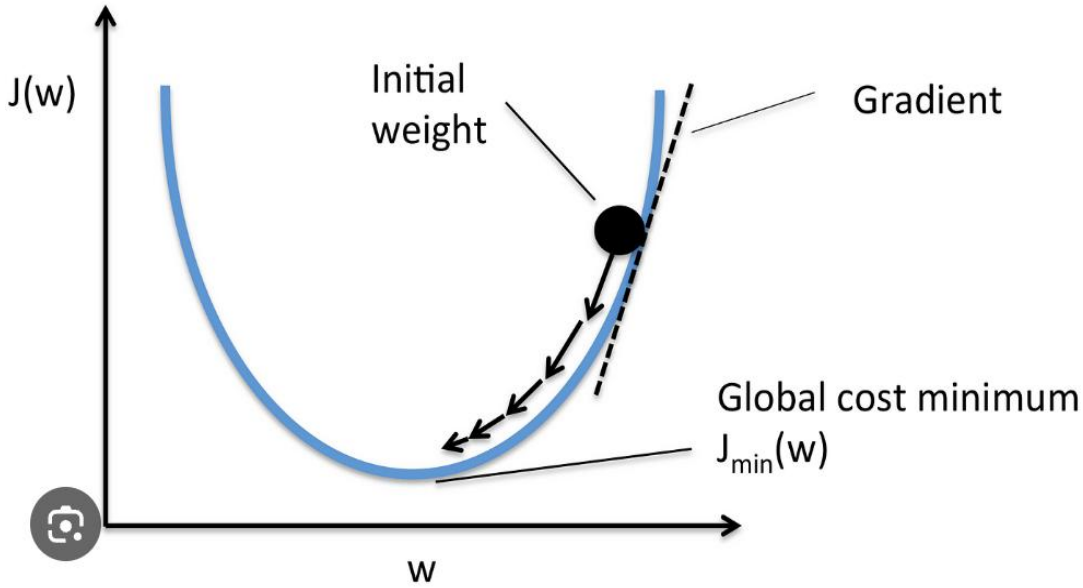
Other Example:
# Cross Entropy (for classification tasks)
criterion = nn.CrossEntropyLoss()

ML Steps in PyTorch



Here we use Stochastic Gradient Descent Optimizer

```
# Define optimizer
optimizer = optim.SGD(model.parameters(), lr=0.01)
```



ML Steps in PyTorch


```
# Train the model
epochs = 10
for epoch in range(epochs):
    # Forward pass
    outputs = model(xs)
    loss = criterion(outputs, ys)

    # Backward pass and optimization
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

- Take the input: put xs to model,
- Calculate the loss
- Zero the gradients before the backward pass in current iteration
  optimizer.zero_grad()
- loss.backward() # Compute gradients
- Optimizer.step() # # Update parameters based on gradients

ML Steps in PyTorch


```
# Predict
with torch.no_grad():
    prediction = model(torch.tensor([[10.0]]))
    print(prediction.item())
```

```
18.317289352416992
```

torch.no_grad() :
Prevents calculations from being added into gradient computation graph. Usually used to prevent accidental training on validation/testing data.

model.eval(): put model in evaluation mode

ML Steps in PyTorch

"mps" If mac

Use GPU: move input and model to GPU

```
[1] !nvidia-smi

Thu Oct 17 18:16:48 2024
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla T4                       Off | 00000000:00:04.0 Off |                    0 |
| N/A   49C    P8              10W /  70W |      0MiB / 15360MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
|  No running processes found                                                           |
+---------------------------------------------------------------------------------------+
```

```python
# Check if GPU is available, else fallback to CPU
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

```python
xs = torch.tensor(xs).view(-1, 1).to(device)  # Move to GPU (if available)
ys = torch.tensor(ys).view(-1, 1).to(device)
```

```python
# Initialize the model and move it to the selected device
model = SimpleModel().to(device)
```

```python
with torch.no_grad():
    # Move input tensor to GPU
    prediction = model(torch.tensor([[10.0]], device=device))
    print(prediction.item())
```

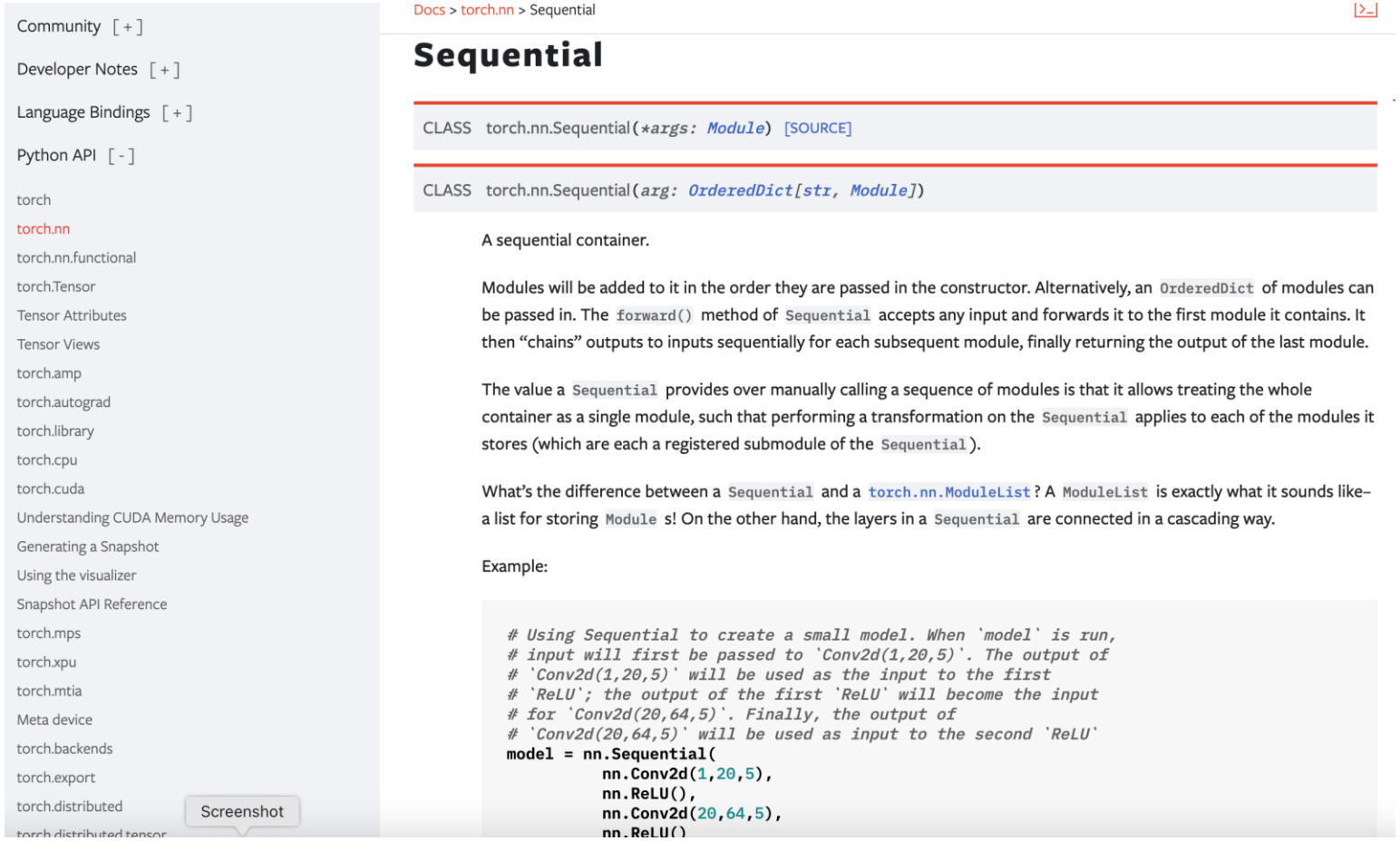
ML Steps in PyTorch

Save and load model

```python
# Save the model state
torch.save(model.state_dict(), 'model_state.pth')

# Load the model state
loaded_model = SimpleModel().to(device)
loaded_model.load_state_dict(torch.load('model_state.pth'))
loaded_model.eval()  # Set to evaluation mode

# Predict using the loaded model
with torch.no_grad():
    prediction = loaded_model(torch.tensor([[10.0]], device=device))
    print(f"Prediction: {prediction.item()}")
```

PyTorch document

https://pytorch.org/docs/stable/



FSU

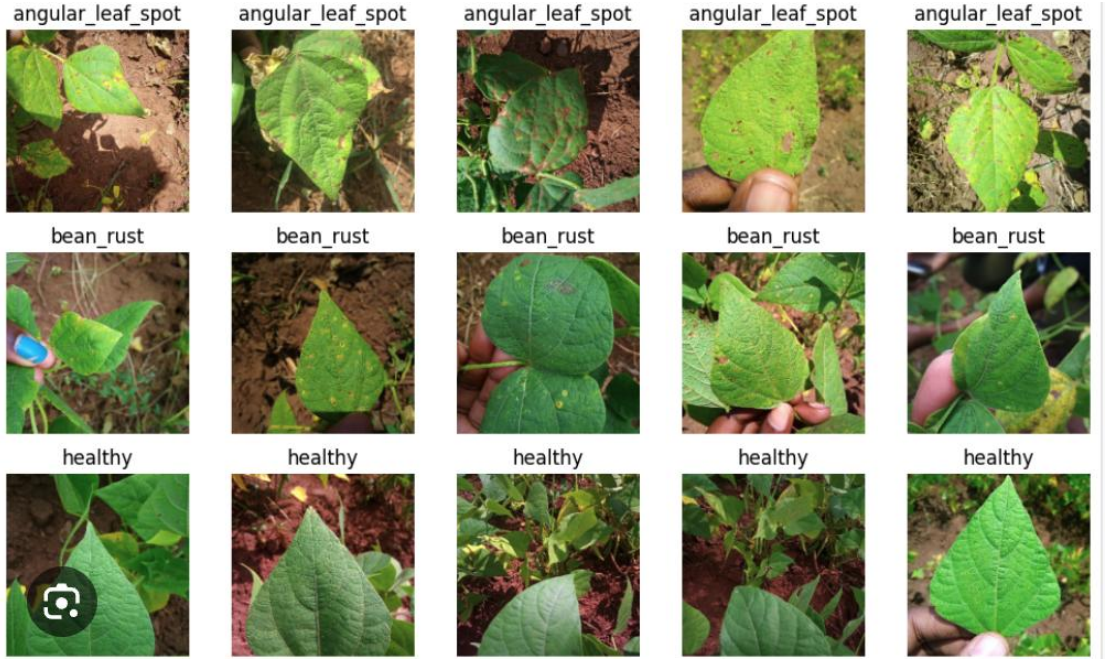
CNN Example

A simple image classification. From load data to testing model

- (Optional) Transfer Learning Example

CNN Example – Image classification

- Classify the bean's category
- https://www.kaggle.com/datasets/therealoise/bean-disease-dataset

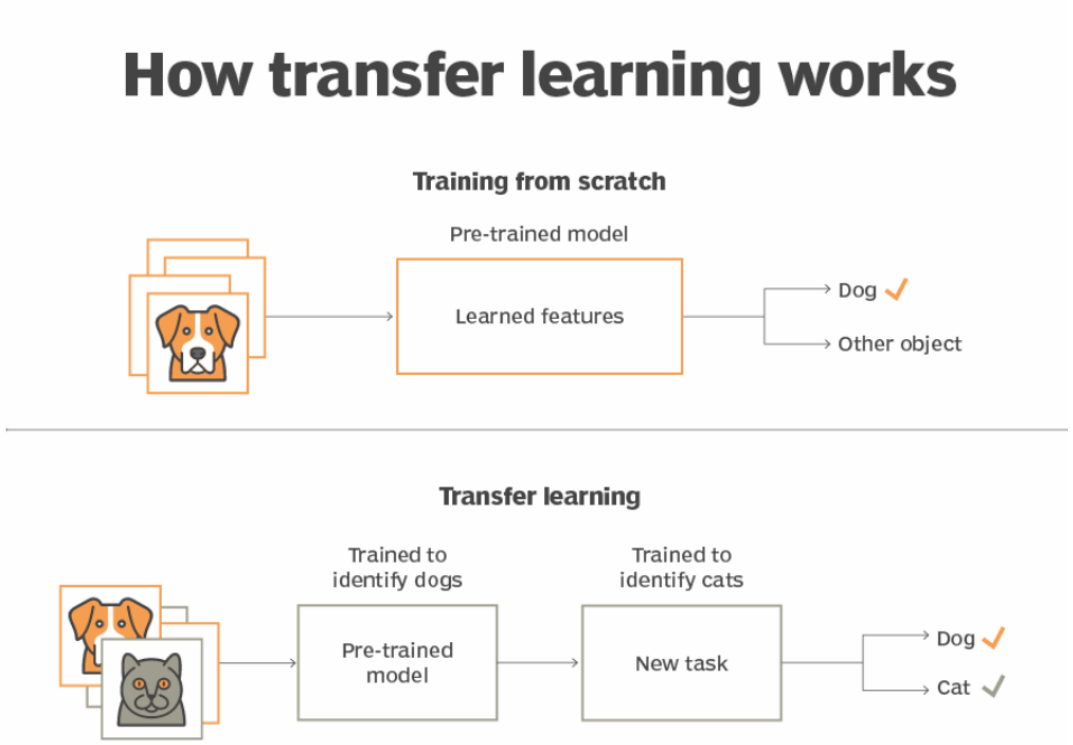


https://colab.research.google.com/drive/1i_77s0GqXkcqnqBeuvb5A0u-REQ4X6bD?usp=sharing

Transfer Learning

- **Transfer learning** is a machine learning technique where model developed for one task is reused as the starting poir for a model on a second, related task.

- Useful links:
- Tutorials: https://www.scaler.com/topics/pytorch/transfer-learning-pytorch/
- Models Repository: https://huggingface.co/



Presenting your model

(Optional, just some suggestions)

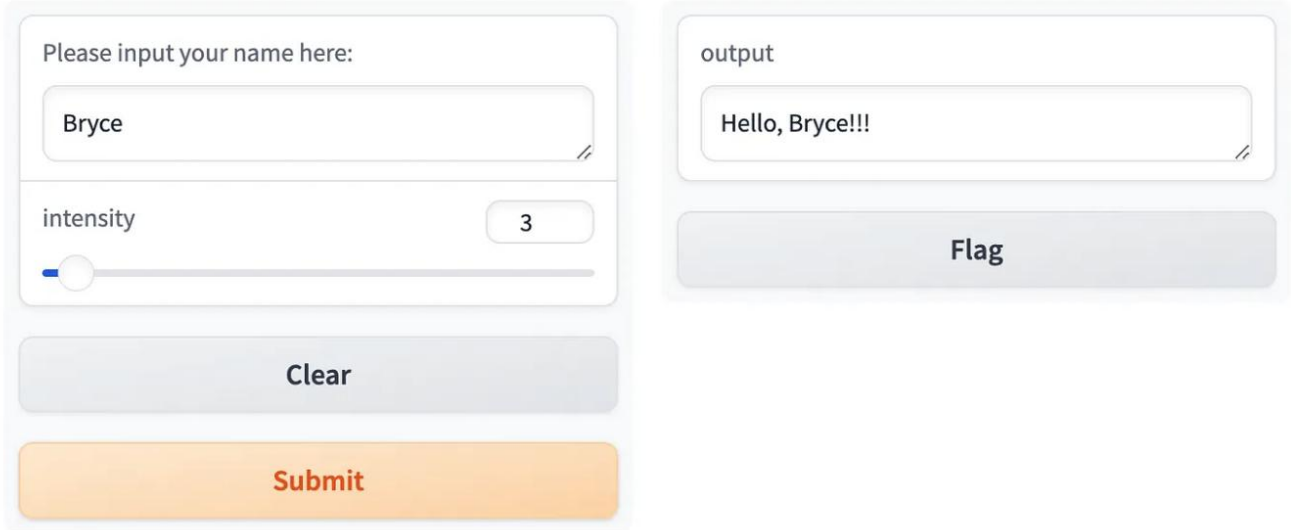
1. Gradio (others: Streamlit)

Gradio

https://www.gradio.app/guides/quickstart

Gradio is an open-source Python package that allows you to quickly build a demo or web application for your machine learning model, API, or any arbitrary Python function. You can then share your demo with a a public link in seconds using Gradio's built-in sharing features. No JavaScript, CSS, or web hosting experience needed!

```python
## Source: https://www.gradio.app/guides/quickstart
import gradio as gr

def greet(name, intensity):
    return "Hello, " + name + "!" * int(intensity)

demo = gr.Interface(
    fn=greet,
    inputs=["text", "slider"],
    outputs=["text"],
)

demo.launch()
```



Tutorial: https://www.youtube.com/watch?v=eE7CamOE-PA&t=395s

Thank you

FSU | FLORIDA STATE UNIVERSITY