CS4262/5462 Machine Learning Systems

Tutorial 1: Intro to TinyTorch

29 Jan 2026

National University of Singapore
School of Computing

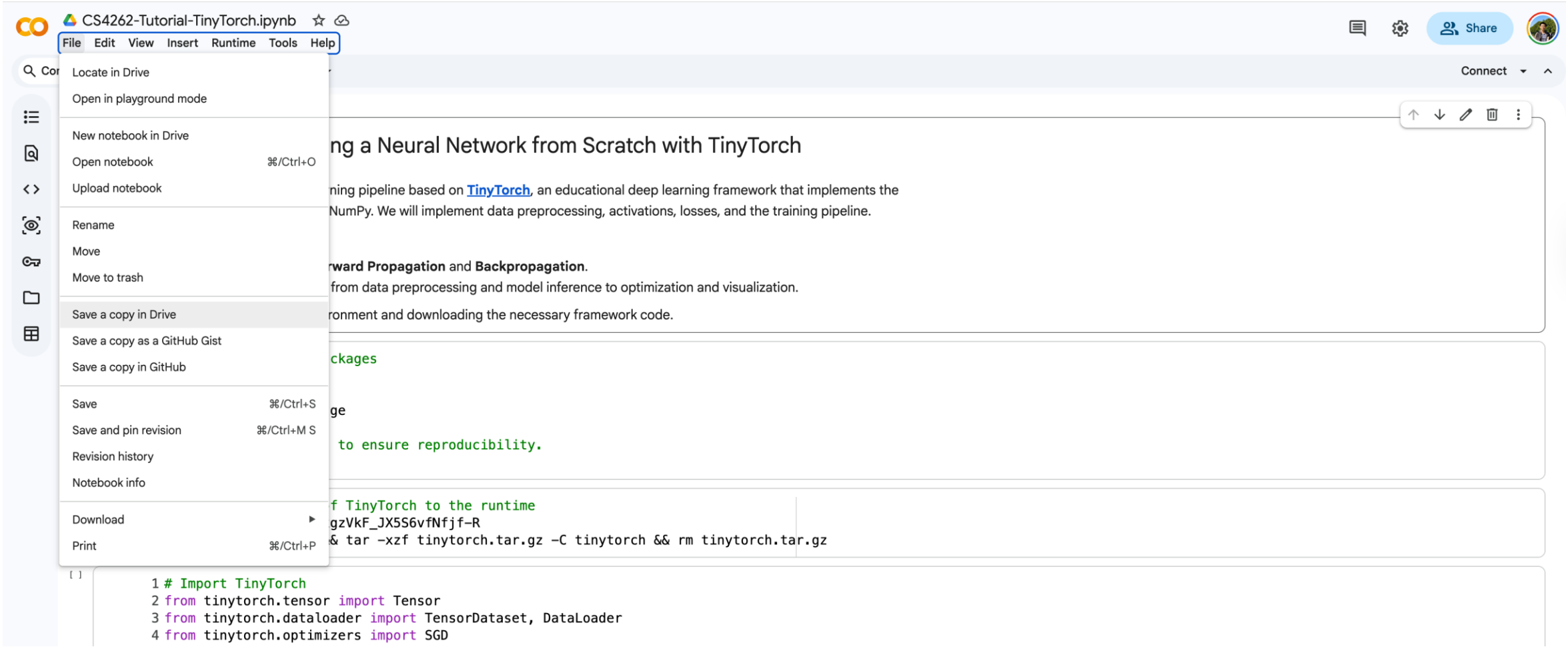
Building a ML Pipeline from Scratch

- In this tutorial, we will build an end-to-end ML pipeline for a Heart Disease binary prediction
- The Framework: **TinyTorch**
  - An educational framework built on NumPy
  - Mimics PyTorch's abstraction (Tensors, Autograd, Layers).

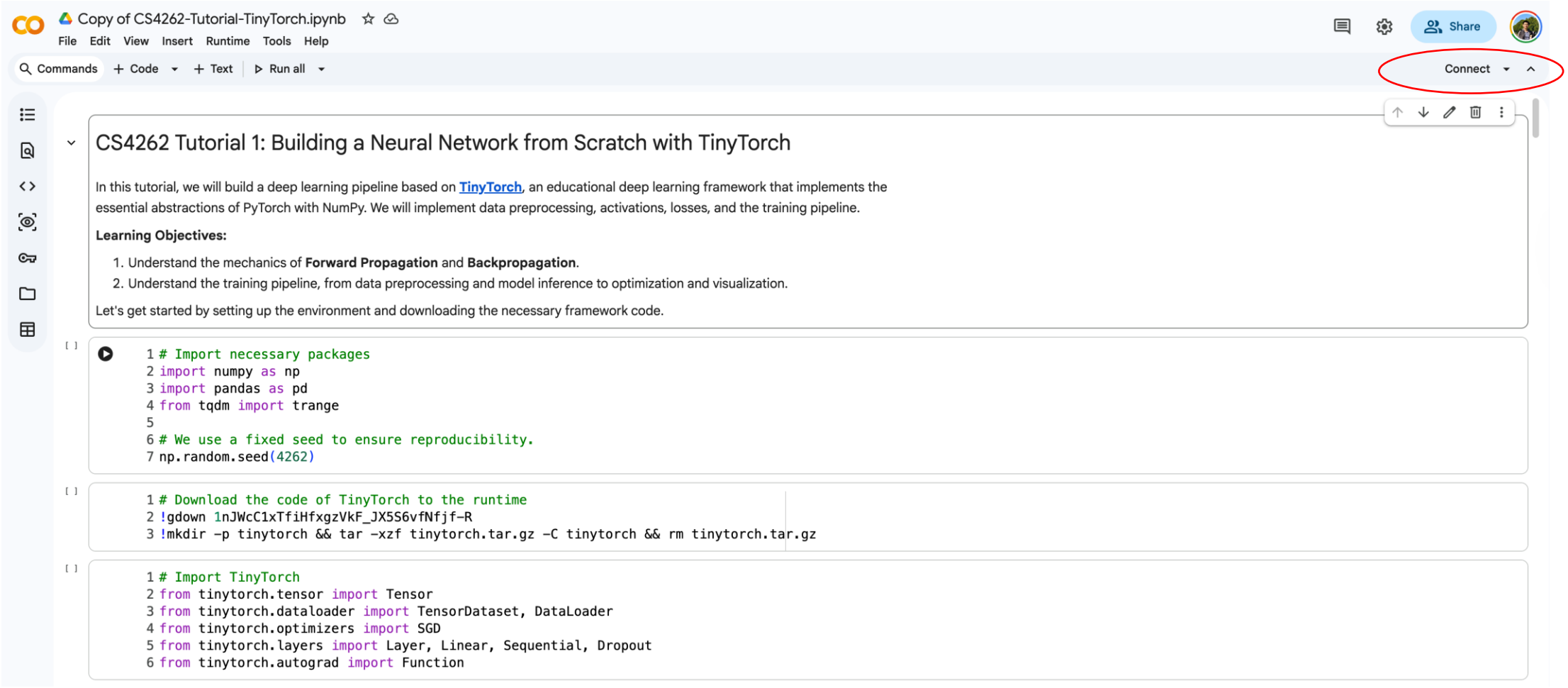
Set up Google Colab

- We will use **Google Colab** for this tutorial
  - A platform that provides free cloud CPU and GPU runtime for jupyter notebook
- Our notebook: https://colab.research.google.com/drive/1seNYOl9emkdEqqn_-Z-Sm2cgsu7bgEQ2

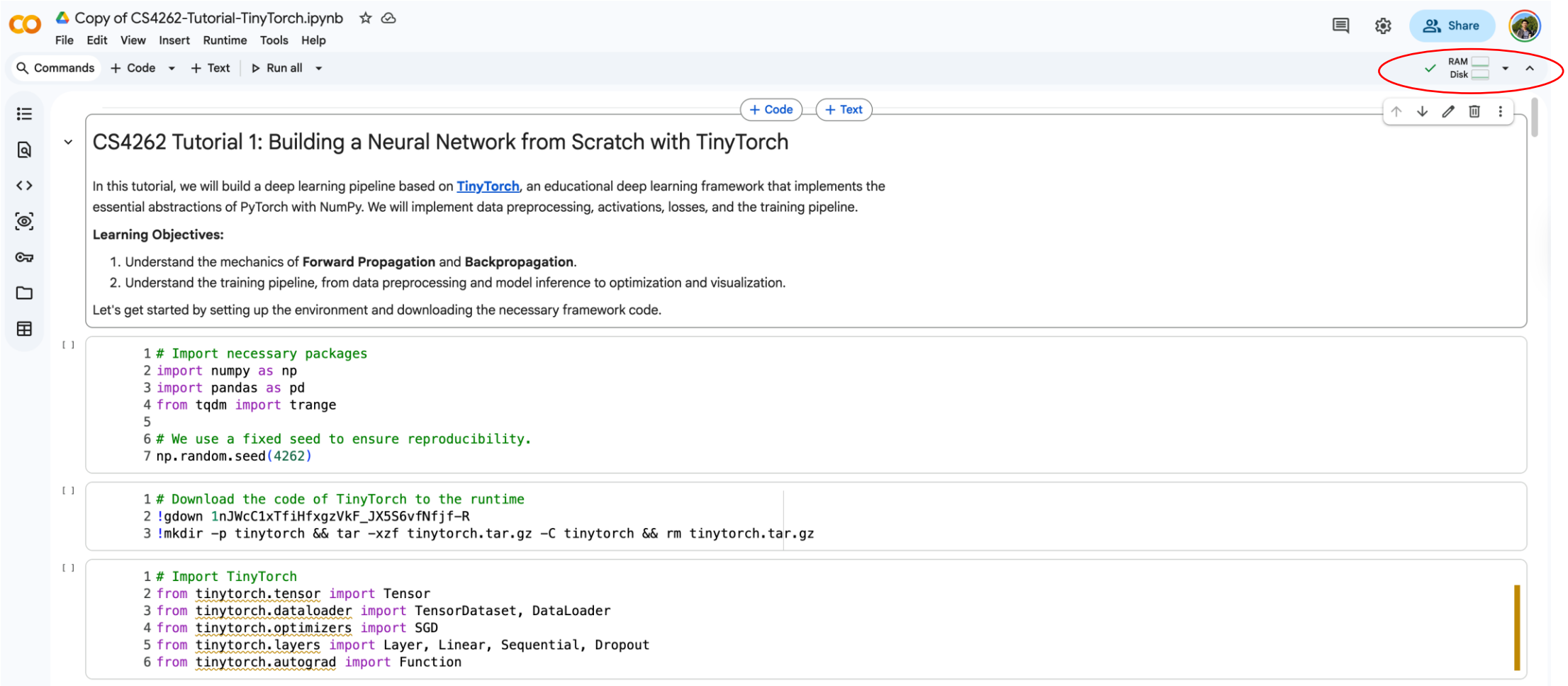
Set up Google Colab – Step 1. Make a Copy



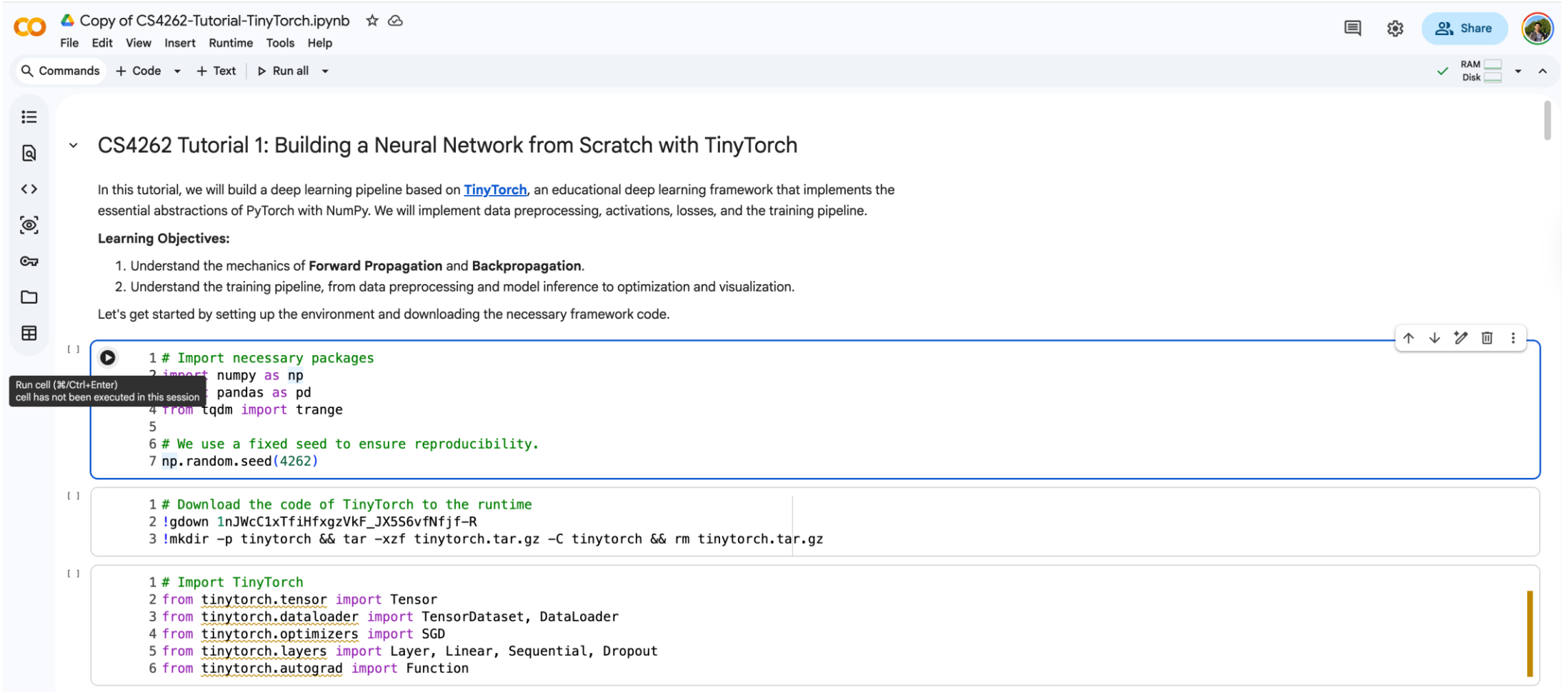
Set up Google Colab – Step 2. Connect to a Runtime



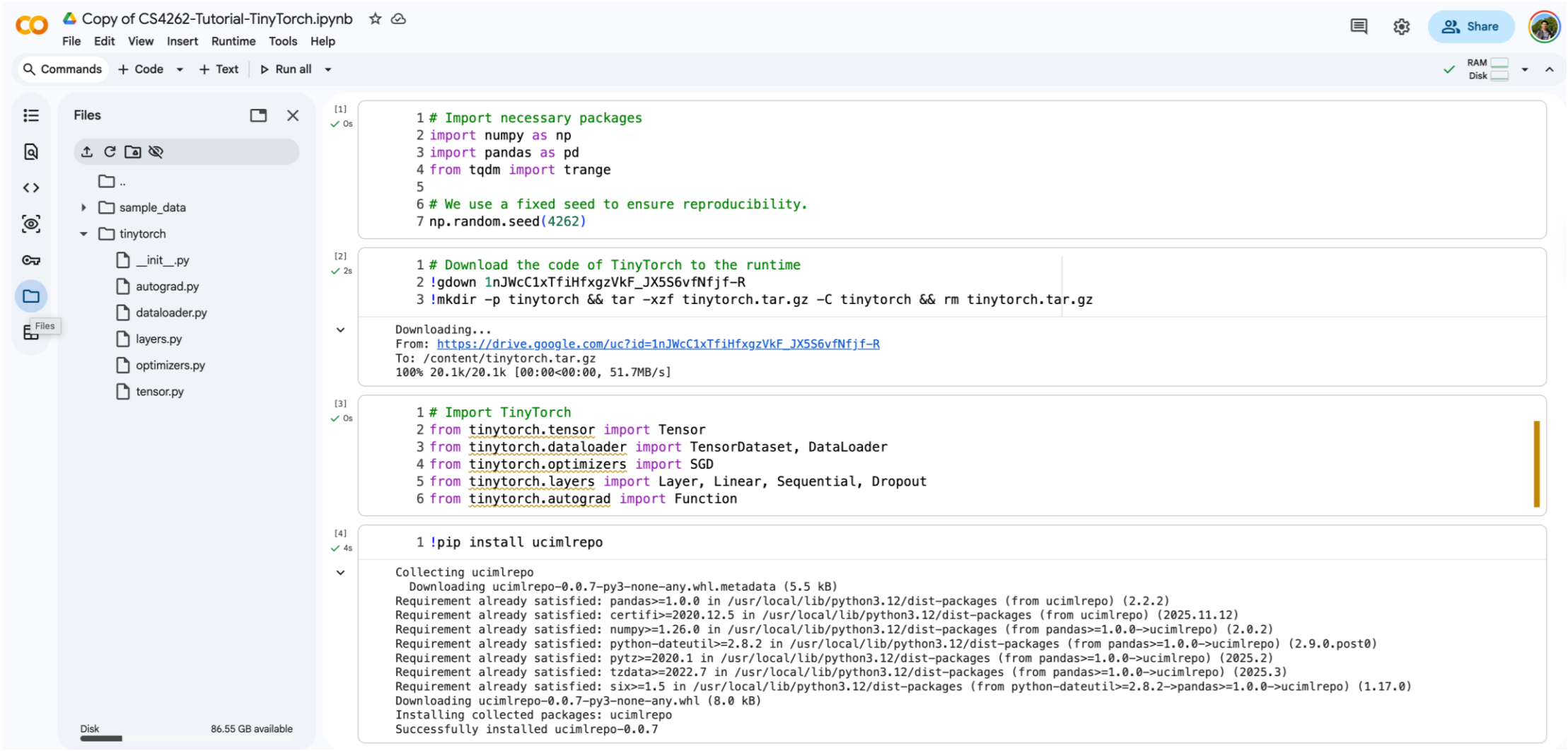
Set up Google Colab – Step 2. Connect to a Runtime



Set up Google Colab – Step 3. Run a Cell



Set up Google Colab – Step 4. Setup the Environment

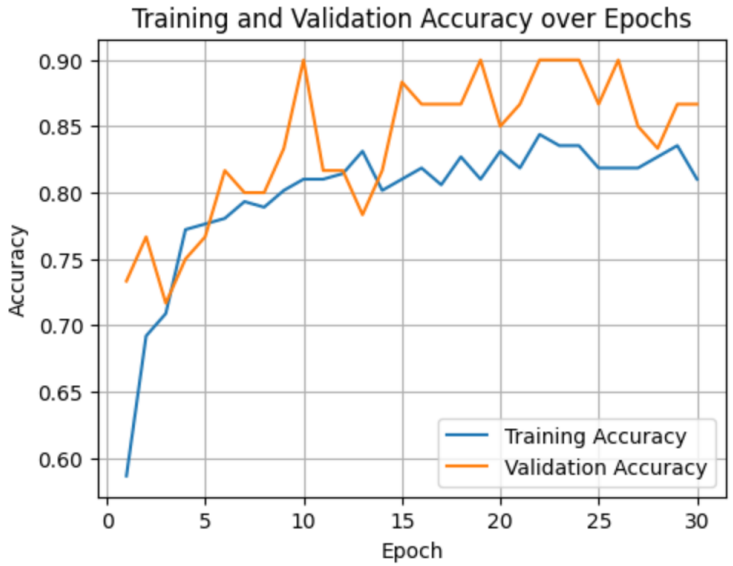


Tasks

- You are expected to fill in all the blanks in the notebook
- Expected outcome after finishing all tasks:

```
ax[1].legend()
ax[1].grid(True)

fig.show()
```



Task 1. Feature Normalization

- Z-score normalization: normalize the features to have mean 0 and std 1

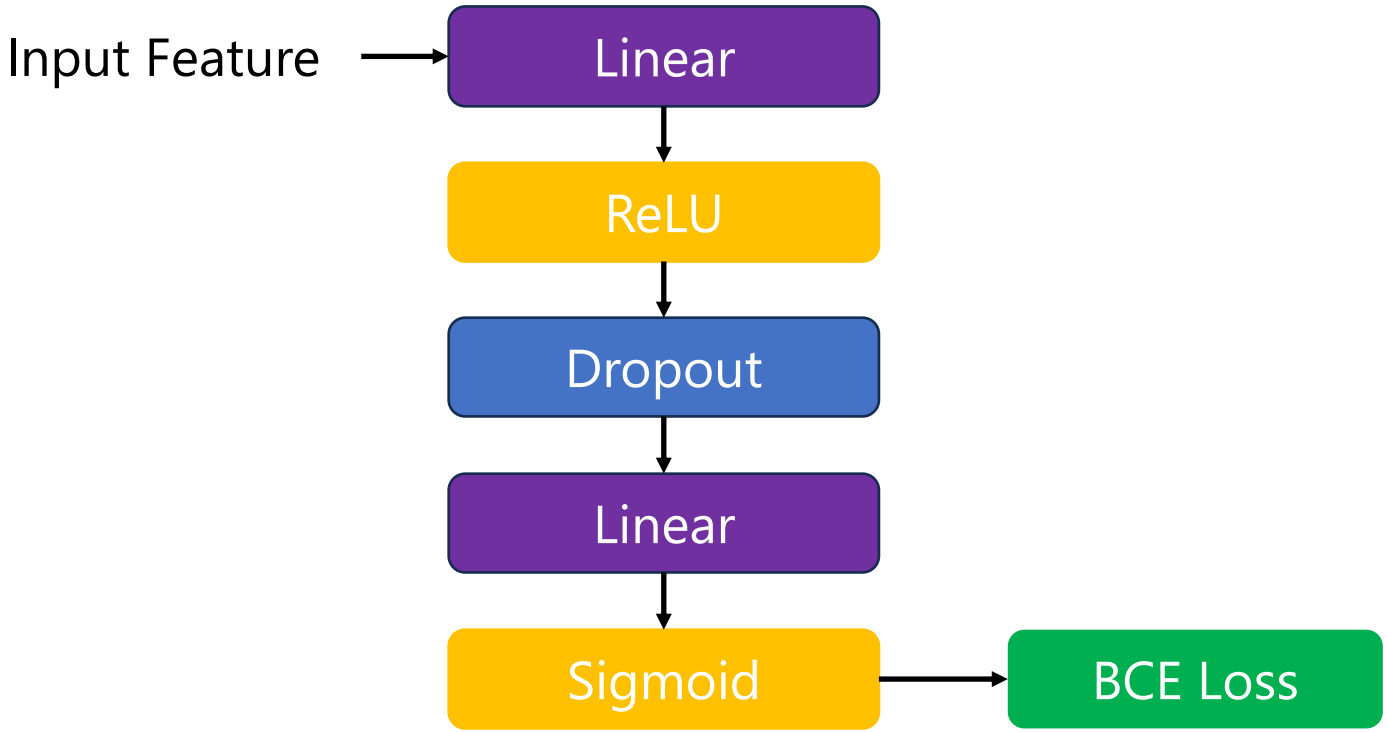
$$x_i' = \frac{x_i - \mu}{\sigma}$$

where μ and σ are the mean and standard deviation respectively

- Normalized data ensure the effectiveness of activation functions and prevents feature dominance

Task 2. Forward and Backward Propagation

- Implement the forward and backward pass of activations (ReLU & Sigmoid) and Binary Cross Entropy Loss
- Model Pipeline



Task 2. Forward and Backward Propagation – ReLU

- Forward Propagation

$$f(x) = \max(0, x)$$

- Backward Propagation

$$\frac{d}{dx}\big(f(x)\big) = \begin{cases} 1, x > 0 \\ 0, \ o.w. \end{cases}$$

Task 2. Forward and Backward Propagation – Sigmoid

- Forward Propagation

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- Backward Propagation

$$\frac{d}{dx}\big(\sigma(x)\big) = \sigma(x)\big(1 - \sigma(x)\big)$$

Task 2. Forward and Backward Propagation – BCE Loss

- Forward Propagation (p_i is model output; y_i is label)

$$L(p_i, y_i) = -\frac{1}{N}\sum_{i=0}^{N}(y_i \log p_i + (1 - y_i)\log(1 - p_i))$$

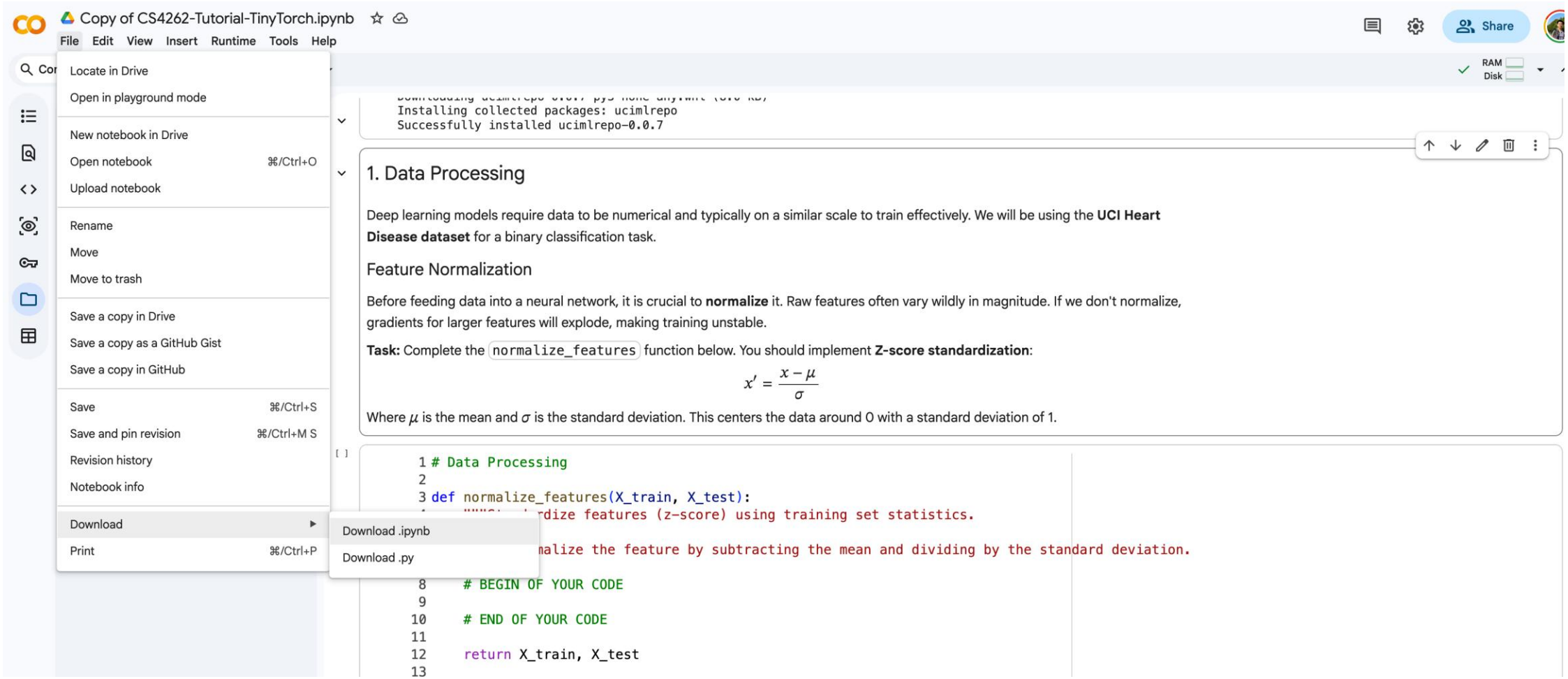
- Backward Propagation

$$\frac{\partial}{\partial p_i}\big(L(p_i, y_i)\big) = \frac{p_i - y_i}{N p_i(1 - p_i)}$$

Task 3. Training Loop

1. Run an inference to get the output
2. Compute the loss using the output and the label
3. Wipe the gradient of the optimizer
4. Compute the new gradients by backward propagation
5. Update the weights with the optimizer

Submit Your Code – Upload the Notebook to Canvas



Thanks for Listening!

Credits to Vijay Janapa Reddi