

# 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

---

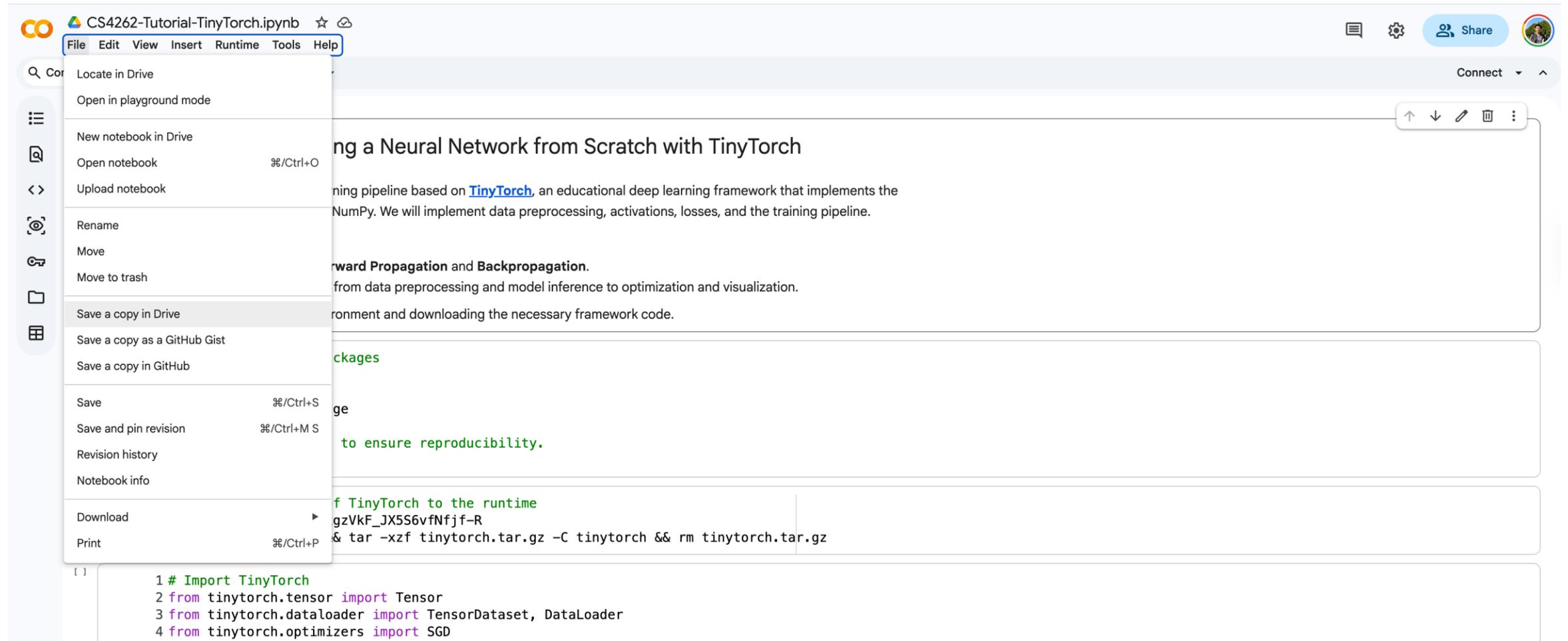
- In this tutorial, we will build an end-to-end ML pipeline for a Heart Disease binary prediction
- The Framework: **TinyTorch**
  - An educational framework built on NumPy
  - Mimics PyTorch's abstraction (Tensors, Autograd, Layers).

# 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-Sm2cg7su7bgEQ2](https://colab.research.google.com/drive/1seNYOl9emkdEqqn_-Z-Sm2cg7su7bgEQ2)

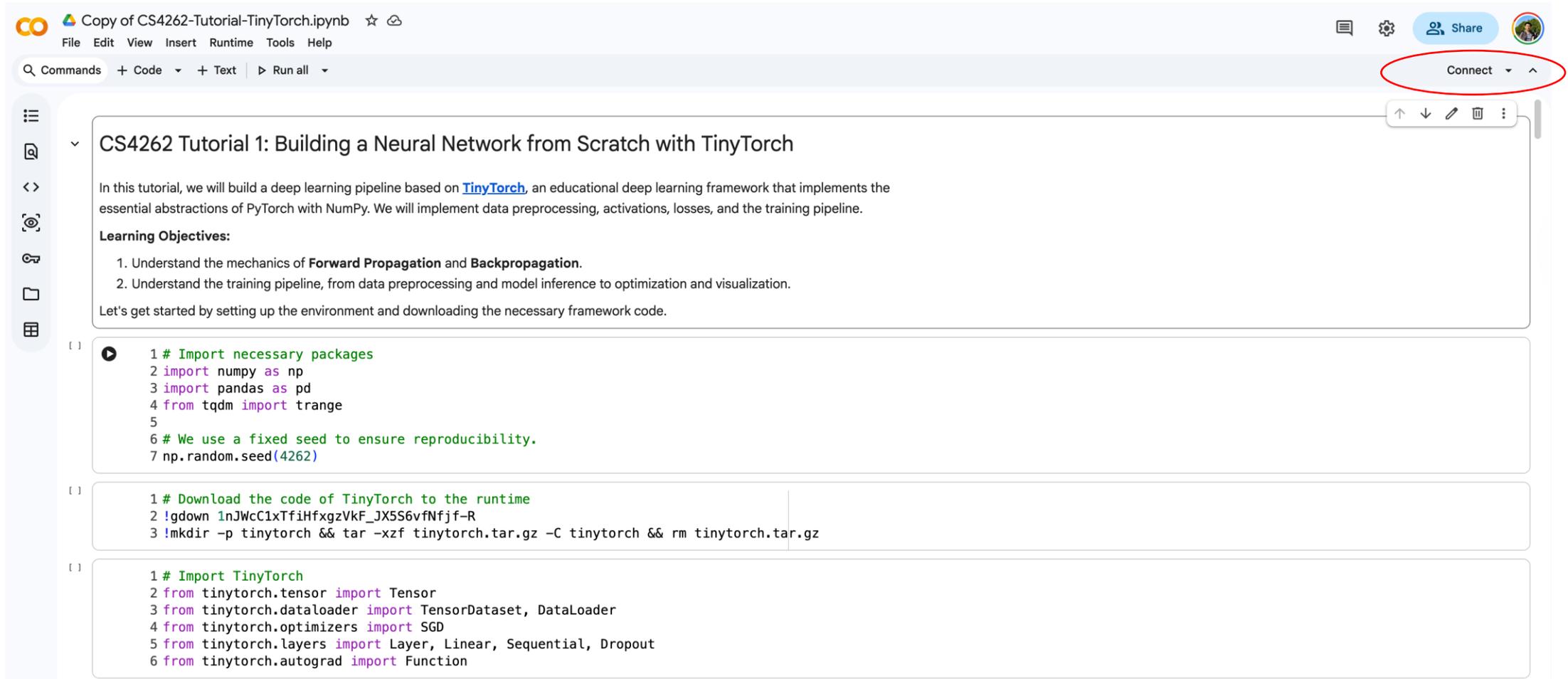
# Set up Google Colab – Step 1. Make a Copy



The screenshot shows the Google Colab interface for a notebook titled "CS4262-Tutorial-TinyTorch.ipynb". The "File" menu is open, and the option "Save a copy in Drive" is highlighted. The notebook content includes a title "Building a Neural Network from Scratch with TinyTorch", an introduction to the TinyTorch framework, and a code cell for installing the framework. The code cell contains the following Python code:

```
[ ]  
1 # Import TinyTorch  
2 from tinytorch.tensor import Tensor  
3 from tinytorch.dataloader import TensorDataset, DataLoader  
4 from tinytorch.optimizers import SGD
```

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



The screenshot shows the Google Colab interface for a notebook titled "Copy of CS4262-Tutorial-TinyTorch.ipynb". The top navigation bar includes "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". On the right side, there are icons for chat, settings, a "Share" button, and a user profile. A red circle highlights the "Connect" button in the top right corner. The notebook content is titled "CS4262 Tutorial 1: Building a Neural Network from Scratch with TinyTorch". It includes an introductory paragraph, learning objectives, and three code blocks for setting up the environment and importing packages.

Copy of CS4262-Tutorial-TinyTorch.ipynb ☆ ☁  
File Edit View Insert Runtime Tools Help

Commands + Code + Text ▶ Run all

Connect

## CS4262 Tutorial 1: Building a Neural Network from Scratch with TinyTorch

In this tutorial, we will build a deep learning pipeline based on [TinyTorch](#), an educational deep learning framework that implements the essential abstractions of PyTorch with NumPy. We will implement data preprocessing, activations, losses, and the training pipeline.

**Learning Objectives:**

1. Understand the mechanics of **Forward Propagation** and **Backpropagation**.
2. Understand the training pipeline, from data preprocessing and model inference to optimization and visualization.

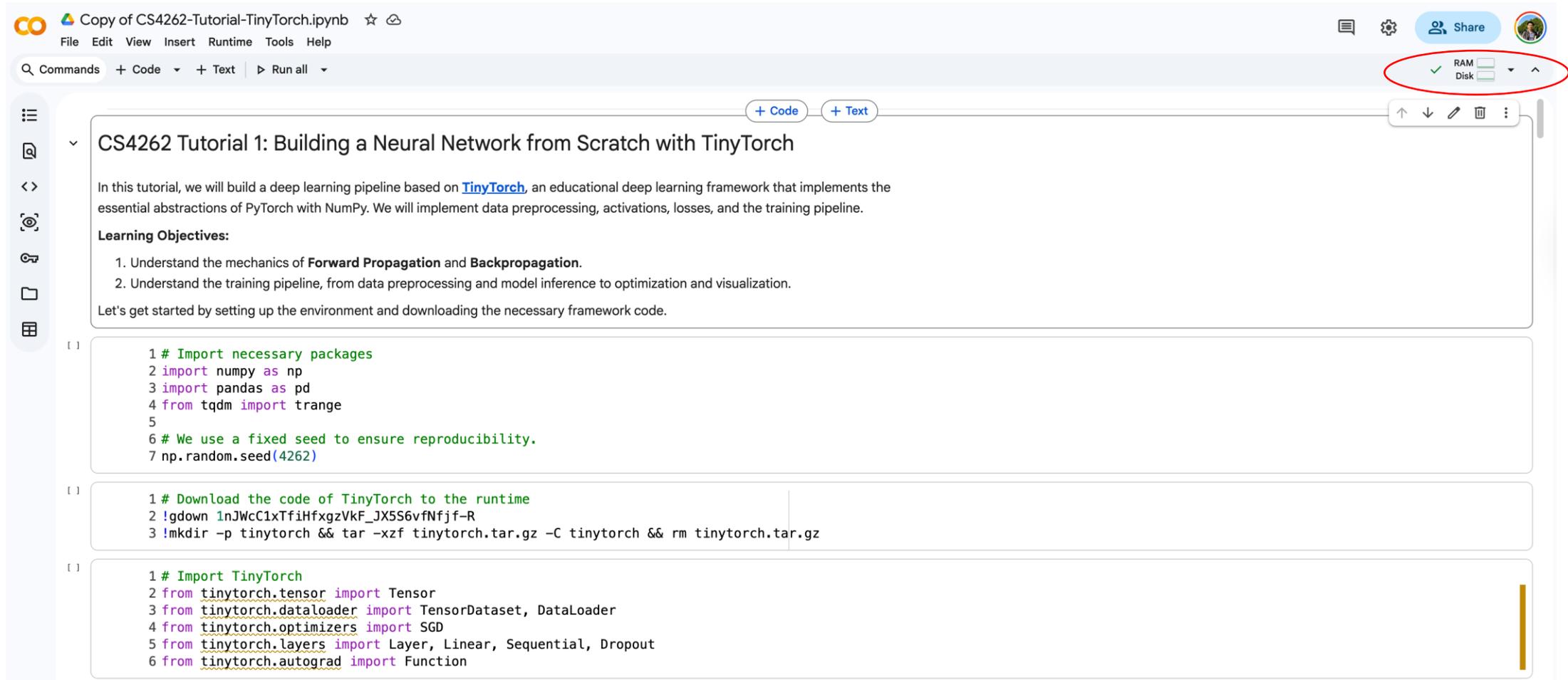
Let's get started by setting up the environment and downloading the necessary framework code.

```
1 # Import necessary packages
2 import numpy as np
3 import pandas as pd
4 from tqdm import trange
5
6 # We use a fixed seed to ensure reproducibility.
7 np.random.seed(4262)
```

```
1 # Download the code of TinyTorch to the runtime
2 !gdown 1nJwC1xTfiHfxgzVKF_JX5S6vfNfjf-R
3 !mkdir -p tinytorch && tar -xzf tinytorch.tar.gz -C tinytorch && rm tinytorch.tar.gz
```

```
1 # Import TinyTorch
2 from tinytorch.tensor import Tensor
3 from tinytorch.dataloader import TensorDataset, DataLoader
4 from tinytorch.optimizers import SGD
5 from tinytorch.layers import Layer, Linear, Sequential, Dropout
6 from tinytorch.autograd import Function
```

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



The screenshot shows the Google Colab interface for a notebook titled "Copy of CS4262-Tutorial-TinyTorch.ipynb". The top right corner features a "Share" button and a runtime status indicator. The runtime status is currently set to "RAM" and "Disk" with a green checkmark, indicating it is connected. The notebook content includes a title, an introduction, learning objectives, and three code blocks for setting up the environment and importing packages.

Copy of CS4262-Tutorial-TinyTorch.ipynb ☆ ☁

File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all

RAM ✓ Disk

## CS4262 Tutorial 1: Building a Neural Network from Scratch with TinyTorch

In this tutorial, we will build a deep learning pipeline based on [TinyTorch](#), an educational deep learning framework that implements the essential abstractions of PyTorch with NumPy. We will implement data preprocessing, activations, losses, and the training pipeline.

**Learning Objectives:**

1. Understand the mechanics of **Forward Propagation** and **Backpropagation**.
2. Understand the training pipeline, from data preprocessing and model inference to optimization and visualization.

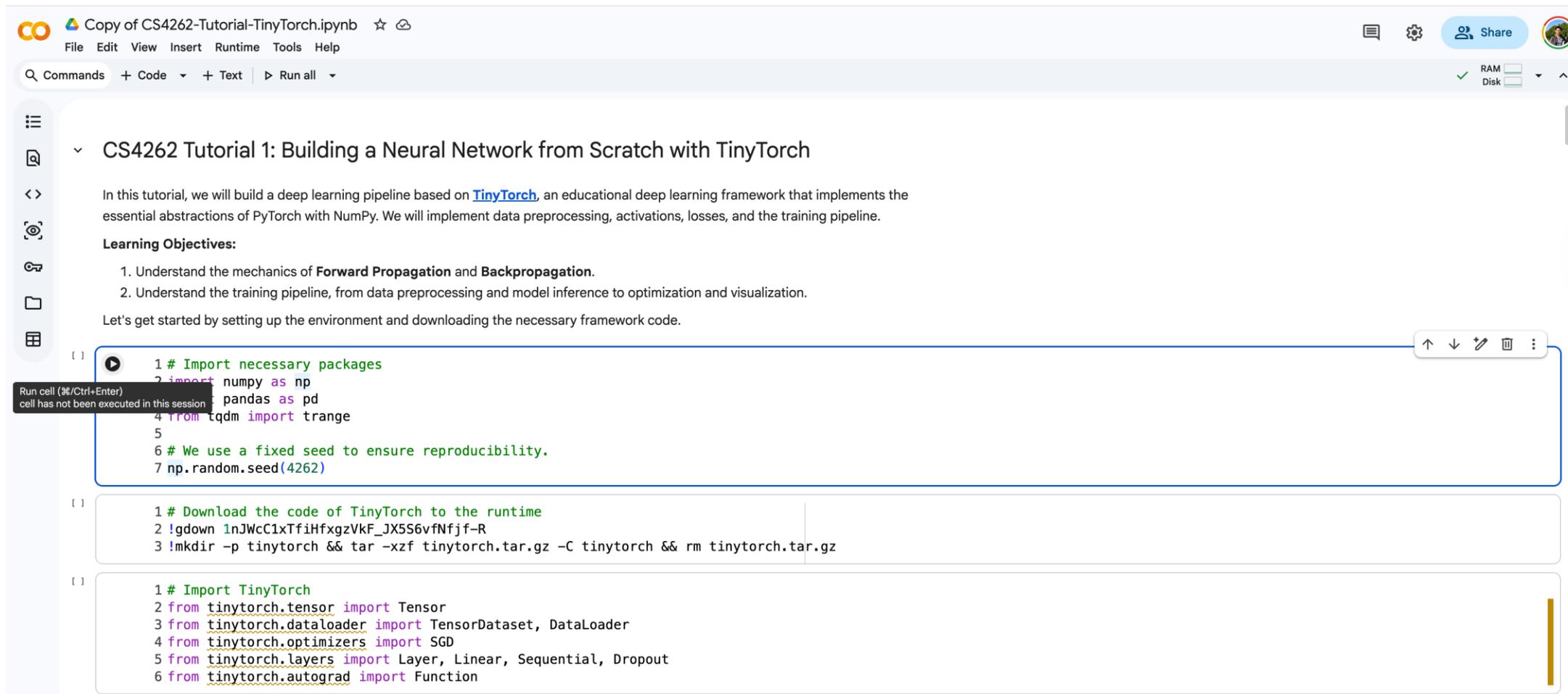
Let's get started by setting up the environment and downloading the necessary framework code.

```
[ ] 1 # Import necessary packages
2 import numpy as np
3 import pandas as pd
4 from tqdm import trange
5
6 # We use a fixed seed to ensure reproducibility.
7 np.random.seed(4262)
```

```
[ ] 1 # Download the code of TinyTorch to the runtime
2 !gdown 1nJWc1xTfiHfxgzVKF_JX5S6vfNfjf-R
3 !mkdir -p tinytorch && tar -xzf tinytorch.tar.gz -C tinytorch && rm tinytorch.tar.gz
```

```
[ ] 1 # Import TinyTorch
2 from tinytorch.tensor import Tensor
3 from tinytorch.dataloader import TensorDataset, DataLoader
4 from tinytorch.optimizers import SGD
5 from tinytorch.layers import Layer, Linear, Sequential, Dropout
6 from tinytorch.autograd import Function
```

# Set up Google Colab – Step 3. Run a Cell



The screenshot shows a Google Colab notebook titled "Copy of CS4262-Tutorial-TinyTorch.ipynb". The notebook content includes a title "CS4262 Tutorial 1: Building a Neural Network from Scratch with TinyTorch", an introductory paragraph, learning objectives, and three code cells. The first code cell is highlighted with a blue border and a play button icon, indicating it is the current cell to be run. A tooltip above the first code cell reads "Run cell (⌘/Ctrl+Enter) cell has not been executed in this session".

CS4262 Tutorial 1: Building a Neural Network from Scratch with TinyTorch

In this tutorial, we will build a deep learning pipeline based on [TinyTorch](#), an educational deep learning framework that implements the essential abstractions of PyTorch with NumPy. We will implement data preprocessing, activations, losses, and the training pipeline.

**Learning Objectives:**

1. Understand the mechanics of **Forward Propagation** and **Backpropagation**.
2. Understand the training pipeline, from data preprocessing and model inference to optimization and visualization.

Let's get started by setting up the environment and downloading the necessary framework code.

```
[ ] 1 # Import necessary packages
    2 import numpy as np
    3 import pandas as pd
    4 from tqdm import trange
    5
    6 # We use a fixed seed to ensure reproducibility.
    7 np.random.seed(4262)
```

```
[ ] 1 # Download the code of TinyTorch to the runtime
    2 !gdown 1nJWcC1xTfiHfxgzVkf_JX5S6vfnfjf-R
    3 !mkdir -p tinytorch && tar -xzf tinytorch.tar.gz -C tinytorch && rm tinytorch.tar.gz
```

```
[ ] 1 # Import TinyTorch
    2 from tinytorch.tensor import Tensor
    3 from tinytorch.dataloader import TensorDataset, DataLoader
    4 from tinytorch.optimizers import SGD
    5 from tinytorch.layers import Layer, Linear, Sequential, Dropout
    6 from tinytorch.autograd import Function
```

# Set up Google Colab – Step 4. Setup the Environment

The screenshot displays the Google Colab interface for a notebook titled "Copy of CS4262-Tutorial-TinyTorch.ipynb". The interface includes a top menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". Below the menu is a "Commands" bar with options for "+ Code", "+ Text", and "Run all". On the left side, there is a "Files" panel showing a directory structure with folders "sample\_data" and "tinytorch", and files including "\_init\_.py", "autograd.py", "dataloader.py", "layers.py", "optimizers.py", and "tensor.py". The main area contains four code cells, each with its execution status and time:

- Cell [1]:** Imports necessary packages: `import numpy as np`, `import pandas as pd`, `from tqdm import trange`, and sets a fixed seed: `np.random.seed(4262)`. Execution time: 0s.
- Cell [2]:** Downloads the code of TinyTorch to the runtime using `!wget` and `!mkdir`. Execution time: 2s. Output shows the download progress from a Google Drive link.
- Cell [3]:** Imports TinyTorch modules: `from tinytorch.tensor import Tensor`, `from tinytorch.dataloader import TensorDataset, DataLoader`, `from tinytorch.optimizers import SGD`, `from tinytorch.layers import Layer, Linear, Sequential, Dropout`, and `from tinytorch.autograd import Function`. Execution time: 0s.
- Cell [4]:** Installs the ucimlrepo package using `!pip install ucimlrepo`. Execution time: 4s. Output shows the collection and installation of the package and its dependencies.

At the bottom left, a "Disk" indicator shows "86.55 GB available".

# Tasks

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



# Task 1. Feature Normalization

---

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

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

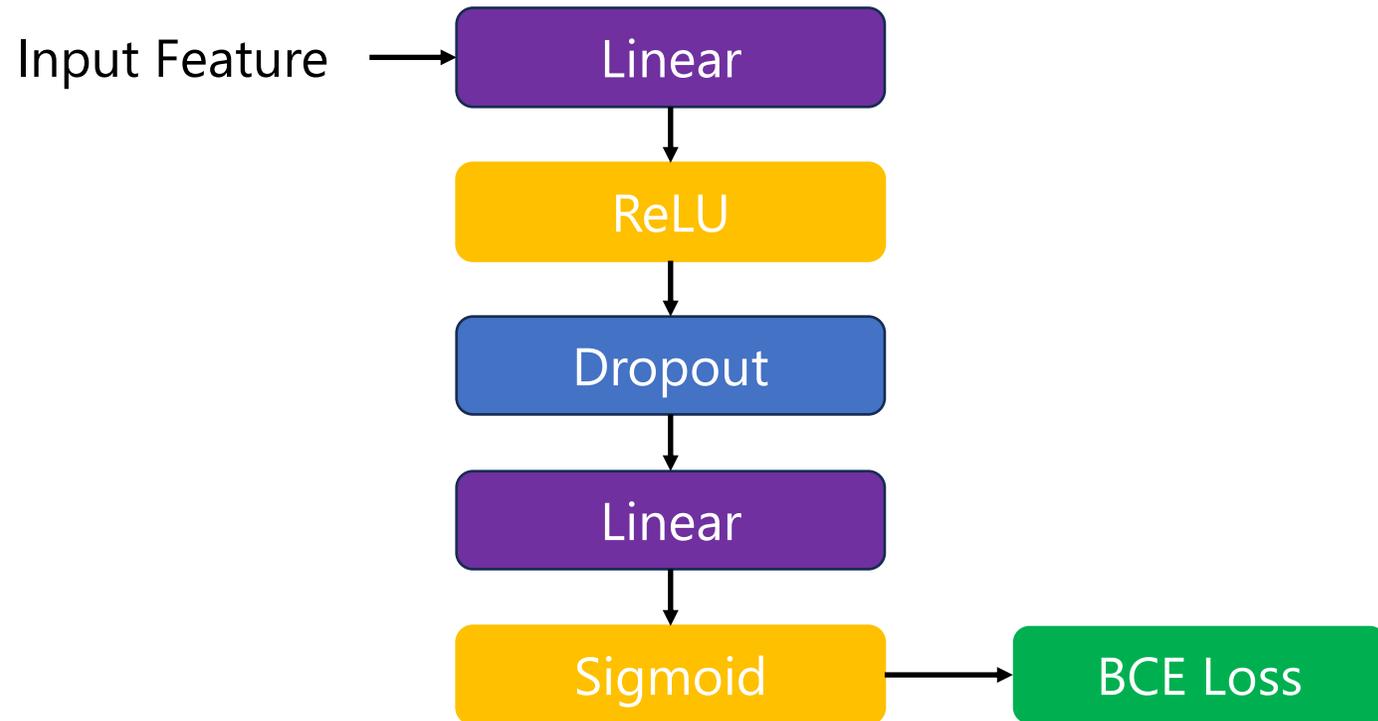
where  $\mu$  and  $\sigma$  are the mean and standard deviation respectively

- 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

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

# Task 2. Forward and Backward Propagation – Sigmoid

---

- Forward Propagation

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

- Backward Propagation

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

## 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

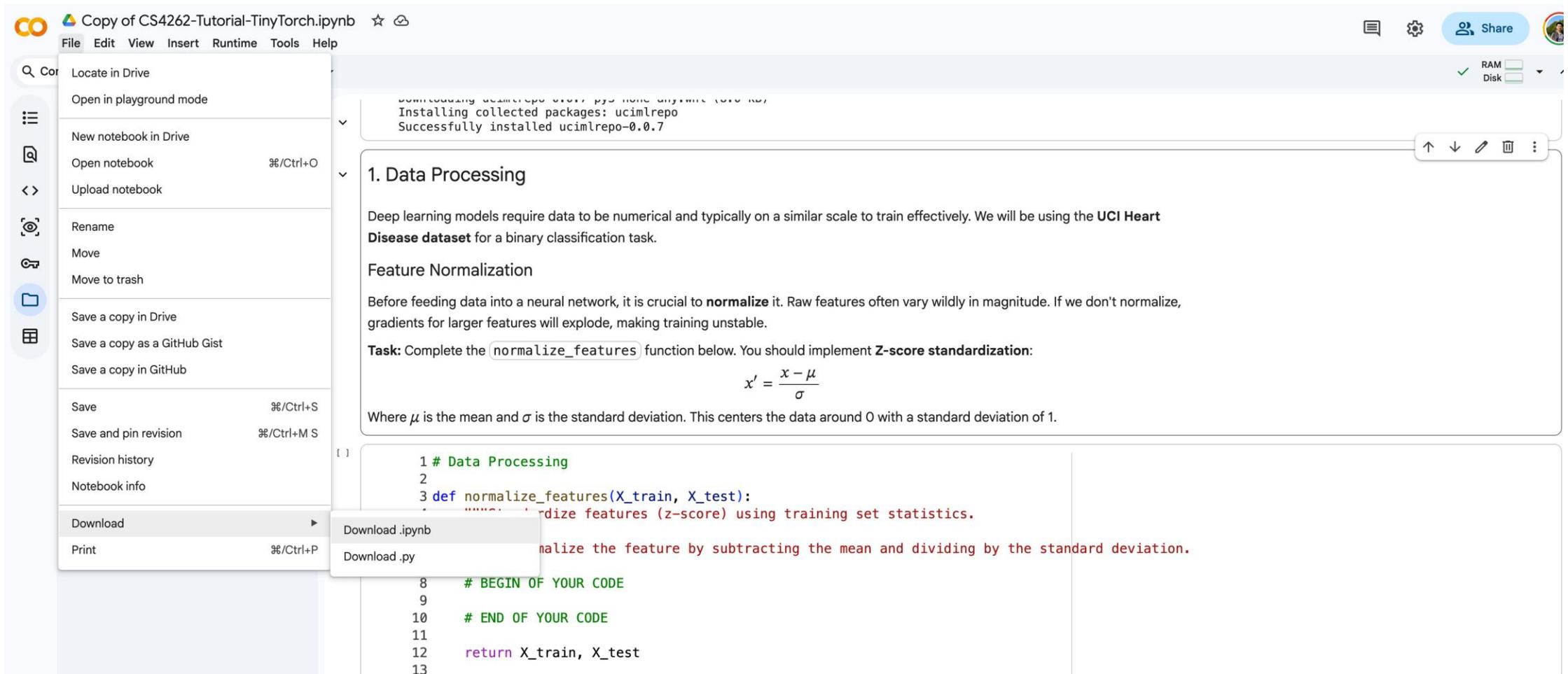
$$\frac{\partial}{\partial p_i} (L(p_i, y_i)) = \frac{p_i - y_i}{N p_i (1 - p_i)}$$

# 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



The screenshot shows a Jupyter Notebook interface. The title bar reads "Copy of CS4262-Tutorial-TinyTorch.ipynb". The menu bar includes "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". On the left, a file menu is open, listing options such as "Locate in Drive", "Open in playground mode", "New notebook in Drive", "Open notebook", "Upload notebook", "Rename", "Move", "Move to trash", "Save a copy in Drive", "Save a copy as a GitHub Gist", "Save a copy in GitHub", "Save", "Save and pin revision", "Revision history", "Notebook info", "Download", and "Print". A sub-menu for "Download" is also visible, showing "Download .ipynb" and "Download .py".

The main content area displays the following text:

```
Downloading ucimlrepo-0.0.7.pyz from https://pypi.org/packages/...  
Installing collected packages: ucimlrepo  
Successfully installed ucimlrepo-0.0.7
```

## 1. Data Processing

Deep learning models require data to be numerical and typically on a similar scale to train effectively. We will be using the **UCI Heart Disease dataset** for a binary classification task.

### Feature Normalization

Before feeding data into a neural network, it is crucial to **normalize** it. Raw features often vary wildly in magnitude. If we don't normalize, gradients for larger features will explode, making training unstable.

**Task:** Complete the `normalize_features` function below. You should implement **Z-score standardization**:

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

Where  $\mu$  is the mean and  $\sigma$  is the standard deviation. This centers the data around 0 with a standard deviation of 1.

```
1 # Data Processing  
2  
3 def normalize_features(X_train, X_test):  
4     """Normalize features (z-score) using training set statistics.  
5     """  
6     # TODO: normalize the feature by subtracting the mean and dividing by the standard deviation.  
7  
8     # BEGIN OF YOUR CODE  
9  
10    # END OF YOUR CODE  
11  
12    return X_train, X_test  
13
```

# Thanks for Listening!

Credits to Vijay Janapa Reddi