Part 1: Train your First Neural Network

SUMMER 2025 STARTER-AI HANDS-ON WORKSHOP SERIES

The University of Texas Rio Grande Valley

Welcome!

- Overview of activities
  - Part 1: Introduction and Training a simple Neural Network
    - Intro to Google Colab + PyTorch
    - Losses, Optimizers, Classification
  - Part 2: Semantic Classification
    - Apply fundamental techniques to classify language as positive/negative
    - Visualize embeddings of language into vectors
  - Part 3a: Fine-tuning a Language Model
    - Use DistilBERT as a small model case to show workflow of fine-tuning
    - Intro to HuggingFace Training tools
  - Part 3b: Capstone
    - Use your skills to fine-tune a model to classify emotions expressed in language

Setup

To start, go to https://starter.utrgv.edu/workshops/summer-2025

Links are embedded in the schedule - Click on "**Intro to Google Colab + Setup**"

You will need a Google account (free) to complete the labs online

Alternatively: you can use your computer if you have a GPU available. Notebooks are available in the github repo below

Starter workshop page

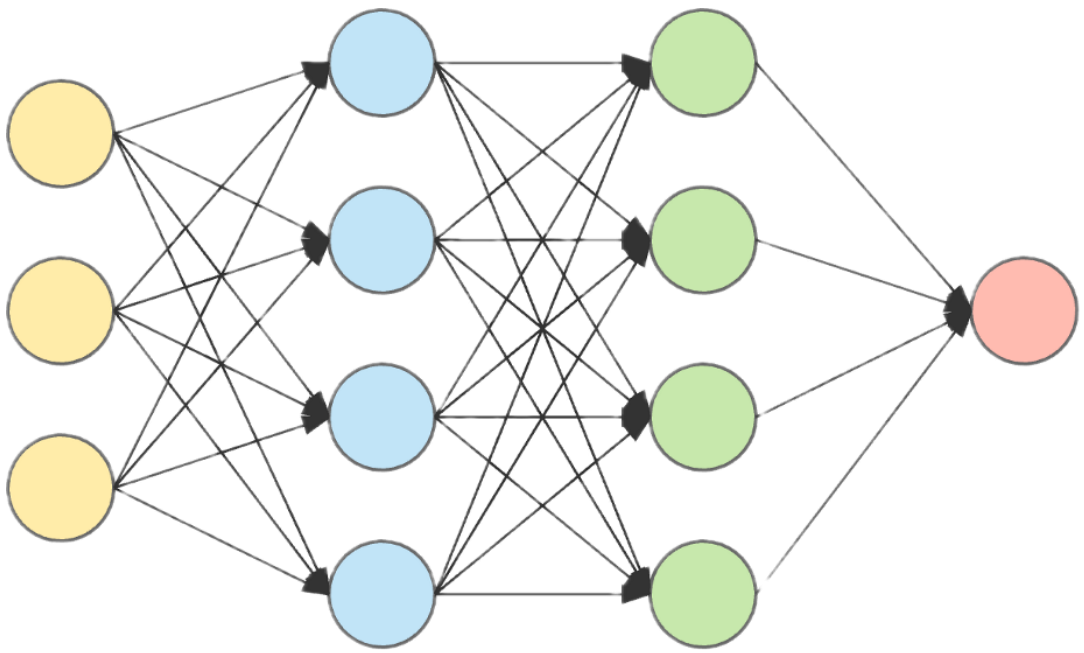
Github link to repo

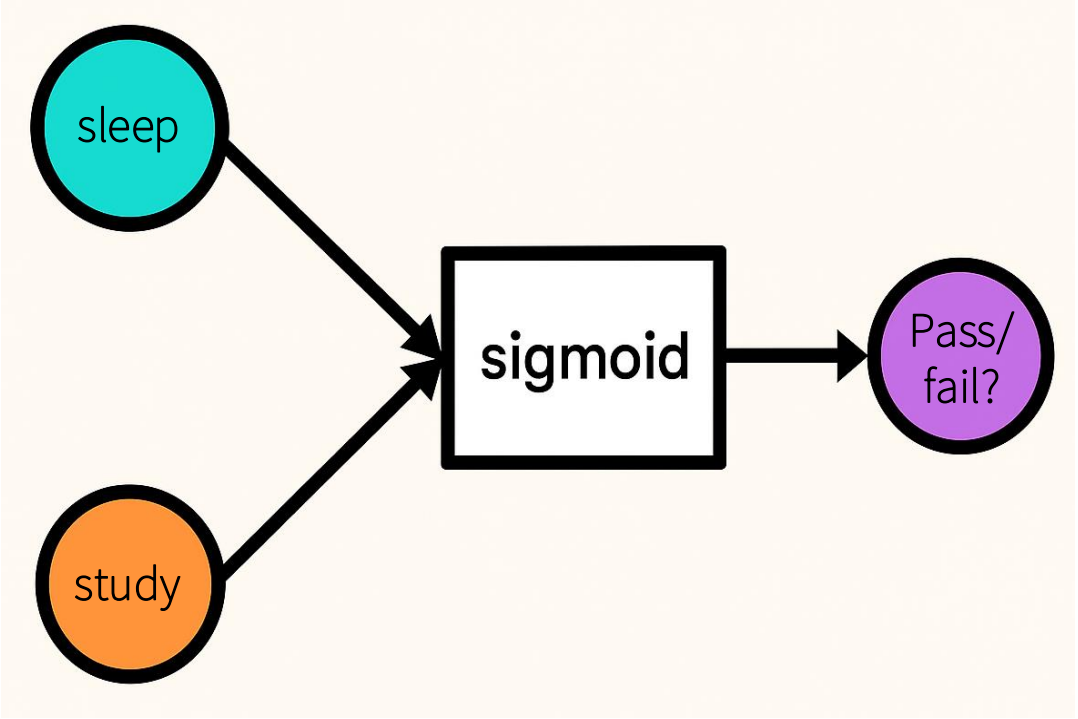
Let's hop in!

Try the notebook out to get familiar with the interface we will be working with

A Brief Intro to Neural Networks

VERY BRIEF!

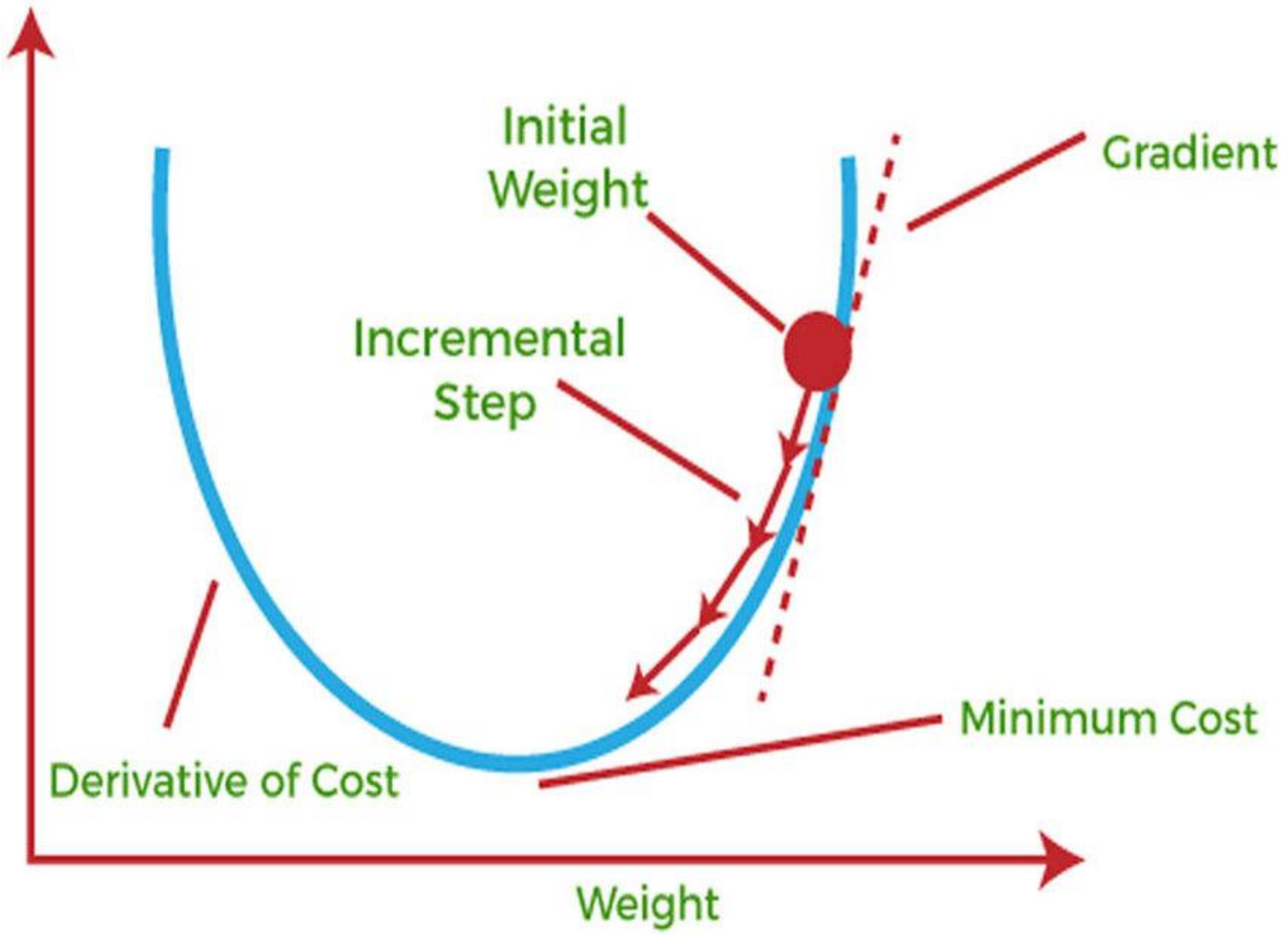




What is a Neural Network?

Tensors

- Generalization of arrays:
  - Scalar = 0D, Vector = 1D, Matrix = 2D, Tensor = nD
- PyTorch uses tensors to store and carry data through the model



Gradients and Model Training

- To learn, models need to know how wrong they are
- Training follows a general process: calculate error → compute gradients → update weights
- Gradient calculation helps to measure:
  - How much each weighted input (like sleep or study hours) contributed to the error.
  - In which direction and how much the weights should change (increase or decrease).

https://www.researchgate.net/publication/372985352/figure/fig2/AS:11431281180215918@1691514092819/Gradient-descent-works-in-the-case-of-one-dimensional-variables-7.jpg

Training Loop

Breakdown of the main training loop throughout the course

The ”training loop” you will see throughout the workshop follows this:

1. Feed input value into model
2. Get prediction
3. Use ground truth to compute loss
4. Perform backpropagation from loss to find gradients
5. Use the optimizer to adjust network weights

```
for epoch in range(101):  # Do 101 training steps
    y_pred = model(X)  # Model's predictions for our data
    loss = loss_fn(y_pred, y)  # How wrong is the model?
    optimizer.zero_grad()  # Clear previous gradients
    loss.backward()  # Compute new gradients
    optimizer.step()  # Update the model
```

Build Your First Classifier

Let’s now work on a simple classifier model to put these concepts to the test.