MNIST Classifier Break Down

- **What is MNIST?**
  - Modified National Institute of Standards and Technology, it's a dataset in ML. It contains 70k grayscale images of handwritten digits (0-9) (28 x 28)

- **What is a classifier?**
  - A ML or deep learning model

- **Project Idea:**
  - Create a MNIST classifier that can recognize which digit is shown in the given image

- **Purpose of project (what do I hope to learn)?**
  - Learn how to train a simple neural network
  - Learn the workflow of deep learning (dataset handling, model building, training and evaluation)
  - Understand how neural networks 'learn' and make predictions

- **What tools will I use to complete the project?**
  - PyTorch
  - MNIST
  - Python
  - Google Colab
  - Matplotlib (Visualize results)
  - NumPy (data manipulation)
  - Torchvision (loads MNIST dataset + preprocess images)

- **Notes:**
  - Weights are the 'knowledge' of the model, it occurs between every pair of layers, controls how much influence a piece of data has on the next layer, through training they adjust to represent patterns in the data

    $$\text{output} = (\text{input} \times \text{weight}) + \text{bias}$$, then applies ReLU

PSEUDOCODE:

1. Import all libraries
2. Load MNIST dataset (split it into training vs testing data)
3. Prepare the data
   - ↳ Normalization (scale numbers down so that it's easier for the model to learn, usually between 0 an 1 or -1 and 1.
   - ↳ Batch (split the data into smaller groups so that the model can train/learn gradually)
   - ↳ Shuffle (mix up the data order so that the model learns and not memorize the order)
4. Define the neural network
   - ↳ Input Layer (each pixel is an input)
   - ↳ (Hidden) layer 1 (learn patterns from data, first run may detect edges/simple shapes)
   - ↳ (Hidden) layer 2 (may detect loops, intersections etc)
     - ↳ Within the hidden layers use Rectified Linear Unit (ReLU) Function (this is because images are nonlinear, but each hidden layer does a linear transformation and so after ReLU should be used allowing the network to model complex patterns)
   - ↳ Output Layer (each neuron outputs a score of probability for each digit (0-9) and the highest score is the predicted digit)
5. Training Loop (for each epoch - a full pass through the entire training dataset)
   - ↳ For each batch loop
     - ↳ Forward Pass (image goes through the network and produces a prediction) (input · weights)
     - ↳ Compute Loss (how far off the prediction is)
     - ↳ Back propagate (calculates how each weight contributes to the error and adjusts weights to reduce errors (flows output → input)
     - ↳ Update Weights with an optimizer (algorithm that uses gradients) making the model more accurate
       - new weight = old weight − learning rate × gradient
6. Test Data
   - ↳ Use unseen data to see how well it generalizes
7. Print Accuracy

- Reflection/Final Thoughts after Coding (Challenges, questions, suprises, what would I do differently etc.)

I made sure my pseudocode went into a lot of detail that way I knew what I had to do for each line. I still needed to use the internet to help me, like with torch functions, and to explain more in depth the purpose of somethings.

Some examples of what I looked up:

- transforms function: used to make image a tensor
- cross entropy loss: measures the dissimilarity between the predicted probability and the true distribution
- and more

- Did I achieve / learn what I wanted to? Explain

Yes, from this project I learned a lot about neural networks and how they are used in code. Neural networks learn by passing data forward to get a prediction, then use a loss function to measure the error compared to correct answers, then working backwards to adjust the networks internal weights and biases. This process continues until the networks predicitions are more accurate. I've also learned about the workflow in deep learning by actually implementing (load data → picked loss function + optimizer → built training loop → evaluate). Because of this project I have created a foundation of knowledge of AI, deep learning, neural networks, and PyTorch. I can now use this foundation to continue learning and build more projects.