Coding Assignment:

Fully Connected Neural Net (FCNN)

Deadline: February 08, 2026

Objective: Build a Deep Learning engine from the ground up, then scale to industrial frameworks.

Part 1: The "No-Framework" Challenge (NumPy Only)

Dataset: UCI Adult Census Income

Goal: Understand the "Black Box" by building it.

Question 1.1: Building the Engine

Using only **NumPy** and standard Python libraries (NO PyTorch/TensorFlow/Keras), implement a 3-layer FCNN.

- **The Task:** Manually code the following:
    - **Layer Initialization:** He-Initialization for weights and zeros for biases.
    - **The Forward Pass:** Z = W * X + b, followed by a ReLU activation.
    - **The Optimizer:** Implement the basic SGD update rule:

$$W_{new} = W_{old} - learning\_rate * \partial L/\partial W.$$

    - **The Backpropagation:** Manually derive and code the chain rule for the gradients of weights and biases.
- **Deliverable:** A training loop that outputs the loss every 100 iterations. Show that your "From-Scratch" model can achieve at least 75% accuracy on the Census test set.

Question 1.2: The Importance of "Pre-Processing"

- **The Task:** Train your NumPy model twice: once with raw data and once with **Min-Max Scaled** data.
- **Deliverable:** Compare the two results. Document what happens to the gradients when features have vastly different scales (e.g., "Age" vs. "Capital Gain").

Part 2: Vision & Feature Interpretation (Frameworks Allowed)

Dataset: MNIST Handwritten Digits (download "MINIST.zip" from Moodle)

Goal: Bridge the gap between pixels and patterns using PyTorch/TensorFlow.

Question 2.1: Weight Visualization

- **The Task:** Build an FCNN using a library (PyTorch/TesorFlow). Once trained, extract the weights of the first hidden layer.
- **Deliverable:** Reshape the weights of 10 different neurons back into "28 x 28" grids and visualize them as heatmaps.
- **Observation:** Describe the "shapes" the network is looking for. Are they dots, lines, or just noise?

Question 2.2: The "Flattening" Experiment

- **The Task:** Randomly shuffle the pixels of every image in the MNIST dataset (use the same shuffle pattern for all images). Train your FCNN on this "scrambled" data.
- **Deliverable:** Compare the accuracy of the model on "Normal MNIST" vs. "Scrambled MNIST."
- **Observation:** Why does the FCNN perform almost identically on both, while a human would find the scrambled version impossible? (This highlights the lack of spatial awareness in FCNNs).

Part 3: Stress Testing & Robustness

Dataset: Tiny ImageNet 10 - a subset of tiny-imagenet-200 available at https://www.kaggle.com/datasets/akash2sharma/tiny-imagenet (**Download "tiny-imagenet.zip" from Moodle**)

Question 3.1: Vanishing Gradients & Modern Fixes

- **The Task:** Create a "Very Deep" FCNN (8+ layers).
  - **Experiment A:** Use **Sigmoid** activations throughout.
  - **Experiment B:** Use **ReLU** activations + **Batch Normalization**.
- **Deliverable:** Plot the "Gradient Norm" (magnitude of the weights' updates) for the first layer in both experiments. Explain which version trained faster and why.

Question 3.2: The Ablation Study

Perform a "Switch-Off" test. Report how the final accuracy changes when you:

1. Remove **Dropout**.
2. Change the **Learning Rate** by a factor of 10 (too high vs. too low).

3. Switch from **Adam** to **Vanilla SGD**.

- **Deliverable:** A summary table showing which factor had the biggest impact on performance

---

Submission Requirements

1. **Code: Submit a well-documented Jupyter Notebook (`.ipynb`).**
2. **Report (`.pdf`) detailing:**
    - **Problem definition & methodology.**
    - **Hyperparameter tuning & architecture choices.**
    - **Results, visualizations, and interpretations.**
    - **Key findings from latent space analysis.**

Guidelines for the Submission:

1. **Upload the Assignment Report in PDF format with the following name: <Group_number>_Assignment1_Report.pdf**
2. **Upload the code files in a single zip file with the following name: <Group_number>_Assignment1_Code.zip**

****