Report

1. ML Classification

- Model Architecture & Hyperparameters

Component	Value
Input Layer	4 features (Iris flower measurements)
Hidden Layer	32 neurons, ReLU activation
Output Layer	3 neurons (for 3 flower classes)
Loss Fuction	CrossEntropyLoss
Optimizer	SGD
Learning Rate	0.01
Epochs	50
Weight initialization	Xavier (Glorot uniform)
Batch Size	Full batch

- Final Accuracy and Interpretation

Metric	Result
Train Accuracy	74%
Test Accuracy	76%

  - Training and Test Accuracy Trends:
    - **Training Accuracy Progression**: Increased from ~55.83% at epoch 10 to ~74.17% at epoch 50.
    - **Test Accuracy Progression**: Improved from ~63.33% to ~76.67%.

  This indicates that the model is learning meaningful patterns, but the overall accuracy is moderate, suggesting it hasn't fully captured the complexity of the Iris dataset.

  - Loss Trend:
    - The loss decreased from ~0.91 to ~0.69, showing improvement, but it plateaued relatively early (no sharp drop near the end).
    - This suggests the model is **converging slowly**, or is unable to minimize the error further with the current architecture/hyperparameters.

2. GenAI Experiment

For the GenAI experiment, I explored how adjusting the temperature parameter influences the creativity and coherence of text generated by a language model (e.g., GPT-based text

generation). The temperature parameter in a language model controls the randomness of predictions: lower values make the output more focused and deterministic, while higher values encourage more diverse and creative word choices.

- Observations:

Temperature	Creativity	Coherence	Example Behavior
0.1	Low - repetitive and predictable	Very high - logical and precise	The model sticks closely to factual, safe completions.
0.7	Balanced - some variation, sensible	Moderate to high - mostly coherent	Sentences flow naturally but with some novel word choices.

3. Key Learnings

Working on both the ML Classification and Generative AI experiments provided valuable hands-on learning experiences. In the ML Classification task, the most challenging part was implementing the neural network without using high-level PyTorch wrappers like torch.nn.Linear or torch.optim.SGD. Manually initializing weights, writing the forward and backward passes, and updating parameters helped me gain a deeper understanding of how neural networks function internally.

For the Generative AI experiment, it was fascinating to observe how a single hyperparameter like temperature could drastically change the nature of the generated text. The lower temperatures produced highly coherent but less creative outputs, while higher temperatures resulted in more imaginative, but sometimes incoherent, results. Understanding this trade-off between creativity and coherence helped me appreciate the flexibility and limitations of text generation models.

Overall, the most interesting aspect was seeing how relatively small changes in model architecture or hyperparameters could significantly impact performance and output quality in both structured (classification) and unstructured (text generation) tasks.