Task completion report

Name: Mudit Bhojak
University: Jecrc University
Reg no. : 21BCON315
Course: B.Tech,CSE(AI-ML)

Task:- 1

Basic ML Classification (Fundamentals)
Objective: Build and train a simple classifier from scratch to prove i understand data flow, model training and evaluation.

Task:- 2

Intro to Generative AI(GenAI)
Objective: Show you can load a pretrained transformers and generate text.

Task 1: Iris Flower Classification using PyTorch (Manual Neural Network)

Objective:

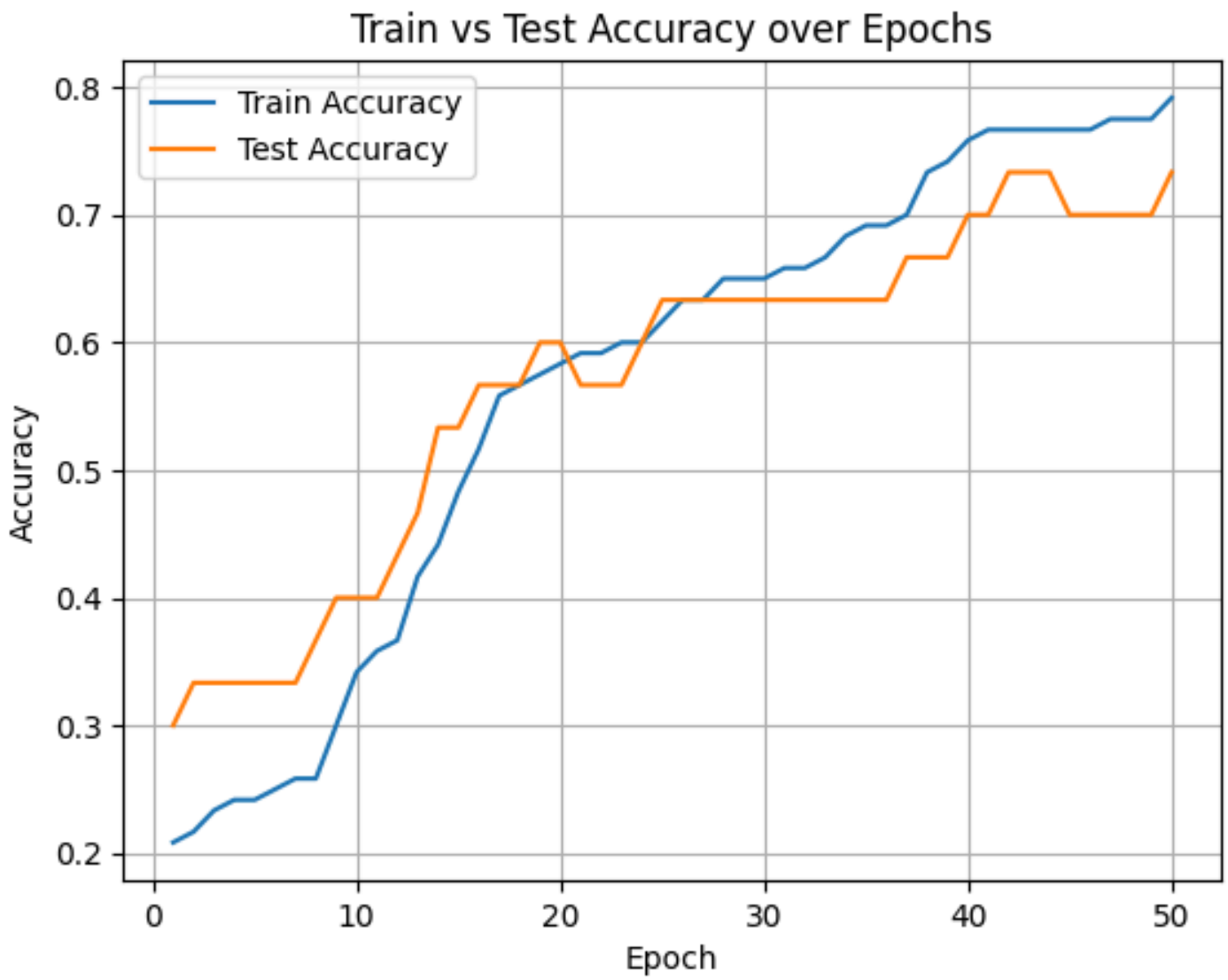
The goal was to build a neural network from scratch using low-level PyTorch constructs (without using nn.Sequential or subclassing nn.Module) to classify flowers in the Iris dataset into three species.

Implementation Details:

- **Data Loading and Preprocessing:**
  - Loaded the iris.csv file containing 150 flower samples with 4 numerical features (sepal length, sepal width, petal length, petal width) and one categorical label (species).
  - Applied label encoding to convert species names into integer class labels (0, 1, 2).
  - Used standardization (z-score normalization) to normalize input features, improving training stability.
- **Train-Test Split:**
  - Divided the dataset into 80% training and 20% testing sets using stratified sampling to maintain class balance across both sets.
- **Neural Network Architecture:**
  - Built a fully connected two-layer neural network manually:
    - Input Layer: 4 nodes (for 4 features)
    - Hidden Layer: 16 neurons with ReLU activation
    - Output Layer: 3 nodes (for 3 classes)
  - Weights and biases for both layers were initialized manually using torch.randn and updated using gradient descent.
- **Training Loop:**
  - Manually implemented:
    - Forward pass using matrix operations and ReLU
    - Loss computation using CrossEntropyLoss
    - Backward pass using loss.backward()
    - Parameter updates with SGD (Stochastic Gradient Descent)
  - Trained the model over 50 epochs, and calculated accuracy on both training and test sets after each epoch.
- **Evaluation and Visualization:**
  - Tracked training and test accuracy across epochs.
  - Plotted a line graph comparing train and test accuracy to visualize learning progress and generalization.

📈 Result:

A consistent increase in accuracy was observed throughout training, indicating effective learning. The final model demonstrated strong classification performance on both train and test sets. The plot below shows the training vs testing accuracy over 50 epochs:



Task 2: Text Generation using Hugging Face GPT-2

Objective:

The task was to explore text generation using GPT-2 (small) from the Hugging Face transformers library. The aim was to:

Accept a text prompt

Generate 50 tokens using top-k sampling (k=50)

Run generation using two temperature values (0.7 and 1.0) to observe variation in creativity and randomness

Save both outputs in separate files for comparison

Implementation Details:

Model Used: gpt2 (small), which is a 12-layer transformer with 117M parameters.

Tokenizer: GPT2Tokenizer from Hugging Face, used to tokenize the input prompt and decode generated tokens.

Sampling Strategy:

Top-k Sampling (k=50): Limits next token choices to the top 50 probable ones at each step.

Temperature:

0.7: Makes the model more focused and deterministic.

1.0: Introduces more randomness and creativity.

Prompt Used:

"The future of artificial intelligence is"

Max New Tokens: 50

Output Format:

Saved both outputs to:

output_temp_0.7.txt

output_temp_1.0.txt