UE23CS352A: MACHINE LEARNING

Week 6: Artificial Neural Networks

1. Lab Overview & Objectives

Welcome to the lab on Neural Networks for Function Approximation.
The objective of this assignment is to give you hands-on experience implementing a neural network from scratch, without relying on high-level frameworks like TensorFlow or PyTorch.

In this lab, you will:

- Generate a custom dataset
- Implement the core components of a neural network: activation functions, loss functions, forward pass, backpropagation, and weight updates.
- Train your neural network to approximate the generated polynomial curve.
- Evaluate and visualize the performance of your model.

2. Datasets

- A synthetic dataset will be generated when you run the code cell, based on the last three digits of your SRN.
- Possible dataset types:
    - Quadratic (degree 2)
    - Cubic (degree 3)
    - Quartic (degree 4)
    - Cubic + Sine term
    - Cubic + Inverse term

- Each dataset contains 100,000 samples split into training (80%) and testing (20%).
- Both input (`x`) and output (`y`) are standardized using `StandardScaler`.

3. Key Concepts

Artificial Neural Networks (ANNs) are computational models inspired by the structure and functioning of the human brain. At their core, they consist of layers of interconnected nodes (or “neurons”), where each connection has an associated weight. During computation, inputs are multiplied by weights, summed with a bias, and then passed through an activation function that introduces non-linearity. This enables neural networks to learn and represent highly complex relationships that linear models cannot capture.

Several important concepts are implemented in the code and are critical to understand:

- **Activation Functions**: These determine the output of each neuron after applying the weighted sum of inputs. You will implement the Rectified Linear Unit (ReLU), which outputs zero for negative inputs and the input itself for positive values. ReLU introduces non-linearity and helps mitigate issues such as the vanishing gradient problem. Its derivative, which you will also implement, is essential for backpropagation.

- **Loss Function**: To measure how well the network’s predictions align with the true values, we use the Mean Squared Error (MSE). This function computes the average of the squared differences between predicted and actual values, penalizing larger errors more heavily. Minimizing this loss is the central objective of training.

- **Forward Propagation**: In this process, input data passes sequentially through each layer of the network. The weighted sum and activation function are applied at every step, and the final prediction is produced at the output layer. Forward propagation is the mechanism by which the network makes predictions.

- **Backpropagation**: Training a neural network involves updating its weights and biases to minimize the loss. Backpropagation computes the gradient of the loss with respect to each parameter using the chain rule of calculus. These gradients indicate how much each parameter should change to reduce the loss.

- **Gradient Descent**: Once gradients are computed, the parameters are updated in the opposite direction of the gradient, scaled by a learning rate. This iterative optimization process allows the network to gradually reduce the loss over many epochs of training.

Your task is to implement and train a neural network with the following architecture:
Input(1) → Hidden Layer 1 → Hidden Layer 2 → Output(1)

4. Instructions and Tasks

Your main task in this lab is to complete the `TODO` sections in the provided notebook and then systematically experiment with different hyperparameter settings to study their impact on the model's performance.

Part A

Baseline Model

1. You will begin by implementing the missing components of the neural network, including the activation functions, loss function, forward propagation, and backpropagation.
2. Once these are complete, you will set up a training loop that updates weights using gradient descent and tracks the training loss over epochs.
3. After training, you must evaluate your model on the test set and generate visualizations such as the training loss curve and a plot comparing predicted outputs against the true target values.
4. Train your network with the following hyperparameters:
    - Learning rate = 0.001
    - Batch size = 32
    - Number of epochs = 10
    - Optimizer = Adam
    - Activation function = ReLU

This completes your baseline model and ensures that it is fully functional before experimentation begins.

Part B

Hyperparameter Exploration

1. Conduct 4 additional experiments where you vary one or more hyperparameters such as learning rate, batch size or number of epochs.
2. For each experiment, retrain your network, evaluate it on the test set, and generate the same visualizations as in the baseline run.
   You may, for example, try increasing the learning rate, using a larger batch size, training for more epochs, or testing a different activation function.

Result Table:
You must maintain a tabular comparison of all experiments. Your table should contain the following columns:

- Experiment
- Learning Rate
- Batch Size
- Number of Epochs
- Optimizer
- Activation Function
- Training Accuracy
- Validation Accuracy

- Test Accuracy
- Training Loss
- Validation Loss
- Test Loss
- Observations

5. Expected Deliverables

1. **Completed Jupyter Notebook (.ipynb)**
   - All `TODO` sections filled.
   - Fully executed with outputs (plots, metrics) visible.
   - Clean, well-documented, error-free.

2. **Lab Report (.pdf)**
   - Summarizes dataset, implementation details, results table, and findings.

6. Report Structure and Guidelines

Your report should include the following sections:

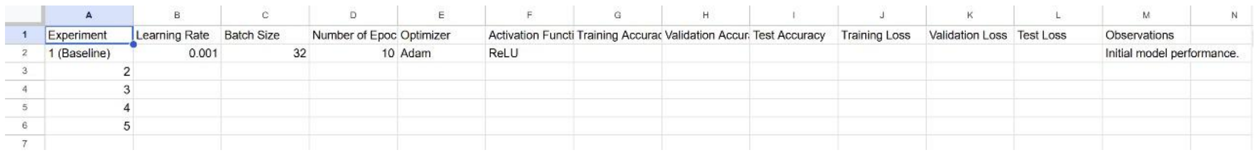
- **Title Page**
  - Project Title, Your Name, SRN, Course, Date.

1. **Introduction**
   - Purpose of the lab
   - Tasks performed.

2. **Dataset Description**
   - Type of polynomial assigned.
   - Number of samples, features, noise level.

3. **Methodology**

4. **Results and Analysis (Add screenshots of plots)**
   - Training loss curve (plot)
   - Final test MSE
   - Plot of predicted vs. actual values
   - Discussion on performance (overfitting / underfitting)
   - Results Table



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Experiment	Learning Rate	Batch Size	Number of Epoc	Optimizer	Activation Functi	Training Accurac	Validation Accur	Test Accuracy	Training Loss	Validation Loss	Test Loss	Observations	
2	1 (Baseline)	0.001	32	10	Adam	ReLU							Initial model performance.	
3	2													
4	3													
5	4													
6	5													
7														

Sample Results Table

5. **Conclusion**