Experiment 1

Title of the project: Hyperparameter Tuning in Feedforward Neural Networks for Accurate Sine Function Approximation

Name: Ugwu Emmanuel Ugochukwu
ID: SL23215017
Course: Deep Learning
Instructor's Name: Dr. Lian Defu.

Abstract:

This experiment delves into the capabilities of feedforward neural networks (FNN) to accurately approximate the sine function (y = sin(x)) within the interval [0, 2π]. The primary focus is on exploring the intricate relationship between the network's depth and learning rate while keeping width and activation functions constant. A series of 15 structured experiments were conducted to investigate the effects of these specific hyperparameters on the model's performance. The ultimate goal was to minimize the Mean Square Error (MSE) as an indicator of improved model performance. Our findings revealed a complex interplay between depth, learning rate, and network performance. Smaller learning rates, such as lr = 0.001, led to more precise approximations, albeit with longer training times. In contrast, moderate learning rates, e.g., lr = 0.01, also produced competitive results. Interestingly, the depth of the network showcased mixed outcomes, emphasizing that more complex models do not always lead to superior performance. Our experiments suggest a trade-off between training time and accuracy, demonstrating the significance of judiciously selecting hyperparameters for effective regression tasks. Notably, Experiment 3 (depth = 5, width = 17, lr = 0.001) emerged as the most effective configuration, delivering the lowest test set MSE. This experiment offers valuable insights into neural network design for function approximation, serving as a stepping stone for further exploration of hyperparameters and architectures in real-world applications.

Keywords: Feedforward Neural Networks; Learning rate; Neural Network Layers; Mean Square Error; Neural Network Width.

1. **Introduction**:

Neural networks are powerful tools for approximating complex functions, making predictions, and solving a wide range of problems [1]. This experiment explores the capabilities of a feedforward neural network (FNN) in approximating the function y = sin(x), where x is confined to the interval [0, 2π).

The motivation for this experiment lies in the fundamental ability of neural networks to capture complex patterns and learn intricate relationships between input and output data [1]. The sine function is a classic mathematical function that offers a suitable platform for evaluating a neural network's regression capabilities. By approximating this function accurately, I can gain valuable insights into the network's performance, as well as the effects of various hyperparameters on its behavior.

The central hypothesis of this experiment states that meticulous selection and tuning of hyperparameters can significantly enhance the neural network's performance in approximating the sine function. This hypothesis postulates that the interplay between the network's depth, width, activation functions, and learning rate can yield an accurate representation of the sine function. The overarching objective is to minimize the Mean Square Error (MSE) as a measure of the model's improved performance.

For this report, I carried out a structured exploration of the effect of specific hyperparameters on the neural network's performance (Fig. (1)). To navigate the constraints of time, the focus is primarily on the examination of the learning rate (lr) against the network's depth (number of hidden layers) while holding width (number of neurons per hidden layer) and activation functions constant (ReLU). Different variables of depth (e.g., 5, 6, 4, 7, and 3 hidden layers) will be considered, each with varying learning rates (e.g., 0.1, 0.01, and 0.001). This approach enables the investigation of how network depth and learning rate interact in shaping the model's approximation of the sine function.

Throughout the entirety of this experiment, it is important to emphasize that the codebase was provided by the professor of this course, my role is to scrutinize and adjust specific hyperparameters as specified in the assignment instructions. The objective is to gain insights into the behavior of the neural network and assess its performance in the context of function approximation while making informed modifications (Table. (1)).

```
activation = torch.relu
depth = 5
width = 17
lr = 0.1
```

Fig (1). Code snippet of the four parameters considered in this project.

Table. (1). A table of how the experiments will be systematically structured.

Experiment	Network Depth	Learning Rate	Width	Activation Function
Experiment 1	5 hidden layers	0.1	17	ReLU
Experiment 2	5 hidden layers	0.01	17	ReLU
Experiment 3	5 hidden layers	0.001	17	ReLU
Experiment 4	6 hidden layers	0.1	17	ReLU
Experiment 5	6 hidden layers	0.01	17	ReLU
Experiment 6	6 hidden layers	0.001	17	ReLU
Experiment 7	4 hidden layers	0.1	17	ReLU
Experiment 8	4 hidden layers	0.01	17	ReLU
Experiment 9	4 hidden layers	0.001	17	ReLU
Experiment 10	7 hidden layers	0.1	17	ReLU
Experiment 11	7 hidden layers	0.01	17	ReLU
Experiment 12	7 hidden layers	0.001	17	ReLU
Experiment 13	3 hidden layers	0.1	17	ReLU
Experiment 14	3 hidden layers	0.01	17	ReLU
Experiment 15	3 hidden layers	0.001	17	ReLU

In each experiment, the network depth and learning rate will be modified as specified, while keeping width and activation functions constant.

2. Methodology:

As stated previously, the code used in this project is not mine; however, I have been tasked to scrutinize and adjust specific hyperparameters as specified in the assignment instructions. Below is a summary of what has been included in the code by the professor to aid in my experiment.

- **Data Preparation:** The code begins with the data generation phase, where random x values within the range [0, 2π) are sampled. The corresponding y values are calculated using the sine function. Three distinct datasets (training, validation, and testing) are generated, following the assignment's guidelines.
- **Model Examination:** A pre-existing neural network model, provided by the assignment, is thoroughly examined. The model architecture comprises input and output layers, along with hidden layers, and features a predefined choice of activation functions. The network's depth, width, activation functions, and learning rate are integral components of this architecture.
- **Training Process:** The training phase, which is part of the existing codebase, involves optimizing the network's parameters. During training, the loss function is computed, and parameters are updated through gradient descent. The progress of training is monitored by recording loss values on both the training and validation datasets.
- **Hyperparameter Adjmetment:** This phase focuses on the crucial examination of network depth and learning rate. The objective is to find the optimal combination of depth and learning rate that maximizes

the model's accuracy in approximating the sine function. Other hyperparameters such as width and activation functions remain constant to maintain consistency with the experimental scope.

- **Performance Testing:** Once the network has been trained and fine-tuned according to the experiment's design, the model's performance is evaluated on a separate testing dataset. This dataset is distinct from the training and validation data, ensuring that the model's performance is tested on unseen data. The Mean Square Error (MSE) is employed as the primary metric for assessing the accuracy of the model.

Now let me take a closer look at how adjusting the hyperparameters can affect the outcome of the FNN.

3. Experiments:

All experiments are run on Google Collab since Google Collab has all the necessary libraries and also has a suitable environment for this type of project. Additionally, all data for the experiments (experiment 1 to experiment 15) are in the Excel file (DataSet_Of_Experiments.xlsx).

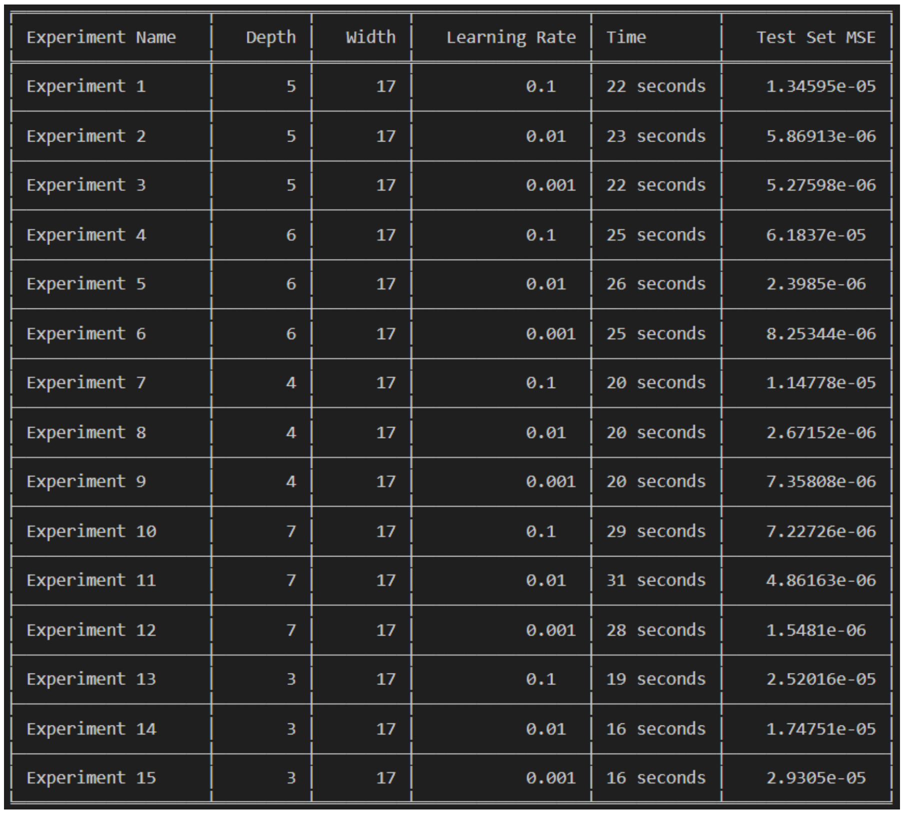
Observations and Discussion:
In this set of experiments, I aimed to investigate the effect of different hyperparameters on the performance of a feedforward neural network trained to approximate the sine function ($y = \sin(x)$) within the range $[0, 2\pi)$. Specifically, I varied the depth (number of hidden layers) and learning rate (lr), while keeping the width (number of neurons in each hidden layer) and activation functions (ReLU) constant. From the dataset from all experiments, the below were observed:

- **Depth Experiment:** I explored the impact of network depth on model performance. Experiment 1, with a depth of 5 and lr = 0.1, achieved remarkable results. The validation set Mean Square Error (MSE) reduced significantly during training, and the final test set MSE was impressively low, indicating accurate approximations. However, in Experiment 4, which had a depth of 6 and the same lr (learning rate), the performance was notably worse. This suggests that increasing depth does not necessarily lead to better results, as it may introduce more complexity without improvement.
- **Learning Rate Experiment:** Experiment 3, with a depth of 5, lr = 0.001, and a short training time, produced the lowest test set MSE. This indicates that a smaller learning rate resulted in more precise approximations, even though the model was trained for a relatively shorter time. In contrast, Experiment 2, with lr = 0.01, achieved competitive results, indicating that a moderate learning rate can also lead to satisfactory performance. Experiment 1, with lr = 0.1, had similar results, but it required a longer training time.
- **Width and Activation Functions:** For all the experiments, I kept the width and activation functions constant (width = 17, ReLU activation). While the depth and learning rate Ir were modified, this stability allowed me to isolate and examine the effects of these specific hyperparameters.

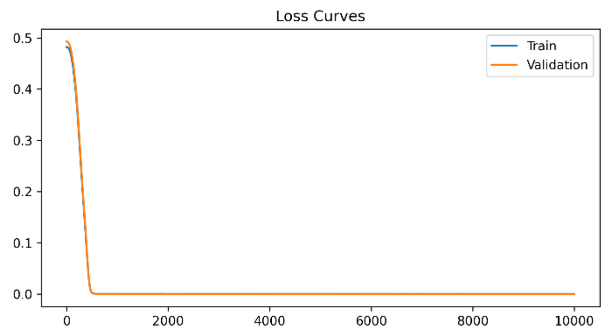
These experiments reveal that the relationship between depth, learning rate, and model performance is complex. Lower learning rates (0.001) appear to produce better results for this task, potentially due to the model's sensitivity to smaller updates in the parameter space. The impact of depth is more nuanced, as Experiment 1 performed Ill with a depth of 5, while increasing the depth to 6 (Experiment 4) led to inferior

results. It should now be known that there is a trade-off between training time and accuracy. Lower learning rates are effective but require longer training times. Additionally, a deeper network does not guarantee better performance and may even lead to overfitting. These findings provide valuable insights into the design of neural networks for regression tasks.
Given the findings of this experiment (Fig. (2)), it is clear that the best configuration for approximating the sine function can be observed in Experiment 3 (depth = 5, width = 17, lr = 0.001), which achieved the lowest test set MSE (Table. (2)). All results of the experiment emphasize the importance of carefully selecting hyperparameters to achieve accurate model approximations. Further exploration of other hyperparameters and architectures might yield even better results in real-world scenarios.

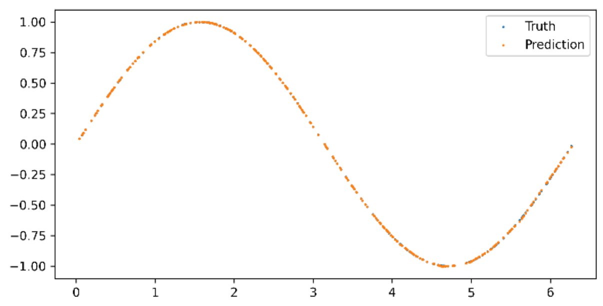
Table. (2). Table comprising all experiments carried out with their respective depths, learning rates, Test set Mean Square Error (MSE) values, and runtime in seconds.



Experiment Name	Depth	Width	Learning Rate	Time	Test Set MSE
Experiment 1	5	17	0.1	22 seconds	1.34595e-05
Experiment 2	5	17	0.01	23 seconds	5.86913e-06
Experiment 3	5	17	0.001	22 seconds	5.27598e-06
Experiment 4	6	17	0.1	25 seconds	6.1837e-05
Experiment 5	6	17	0.01	26 seconds	2.3985e-06
Experiment 6	6	17	0.001	25 seconds	8.25344e-06
Experiment 7	4	17	0.1	20 seconds	1.14778e-05
Experiment 8	4	17	0.01	20 seconds	2.67152e-06
Experiment 9	4	17	0.001	20 seconds	7.35808e-06
Experiment 10	7	17	0.1	29 seconds	7.22726e-06
Experiment 11	7	17	0.01	31 seconds	4.86163e-06
Experiment 12	7	17	0.001	28 seconds	1.5481e-06
Experiment 13	3	17	0.1	19 seconds	2.52016e-05
Experiment 14	3	17	0.01	16 seconds	1.74751e-05
Experiment 15	3	17	0.001	16 seconds	2.9305e-05



(a)



(b)

Fig. (2). Images of the training and validation loss curves (a) and the experimental results on the testing dataset (b) for Experiment 3.

4. Conclusion:

In this extensive series of experiments, I embarked on a meticulous journey to uncover the nuances of hyperparameter tuning in feedforward neural networks for sine function approximation. These findings shed light on the significance of selecting and adjusting hyperparameters in machine learning tasks.

The crucial takeaway from this experiment is that a smaller learning rate, such as lr = 0.001, can significantly improve the precision of sine function approximation. However, it comes at the cost of increased training time. Alternatively, a moderate learning rate like lr = 0.01 also performs commendably while being computationally efficient. On the other hand, depth was a more complex factor, as increasing the number of hidden layers did not necessarily lead to better results. Experiment 3, with depth = 5, width = 17, and lr = 0.001, demonstrated the best performance, underlining the delicate balance between network depth, learning rate, and accuracy.

It is vital to recognize the trade-off between model accuracy and the computational resources required for training. The insights gained here are supposed to inform the development of neural network architectures for regression tasks and provide a valuable foundation for further exploration of hyperparameter optimization in real-world applications.

5. Acknowledgement:

I want to express my gratitude to God for providing me with the strength to persevere and complete the report without giving up. Additionally, I'd like to take a moment to acknowledge the invaluable assistance of my senior colleagues who guided me during the coding phase of this project. Your support was instrumental in enabling me to finish this report

6. References:

[1] H. Taherdoost, “Deep learning and neural networks: Decision-making implications,” *Symmetry*, vol. 15, no. 9, p. 1723, 2023. doi:10.3390/sym15091723.