**Exploring Gradient Descent, Activation Functions, and Neural Networks**

**Author:** Arijit Bhattacharjee  
**roll : 2205452**

**Table of Contents**

1. Introduction
2. Experimental Setup
3. Plots and Visuals
4. Results Summary Table
5. Analysis & Discussion
6. Conclusion
7. Appendix (Code & Logs)

**1. Introduction**

This project investigates the effect of optimization techniques and activation functions in neural networks, specifically for binary classification using the sklearn digits dataset. The notebook covers:

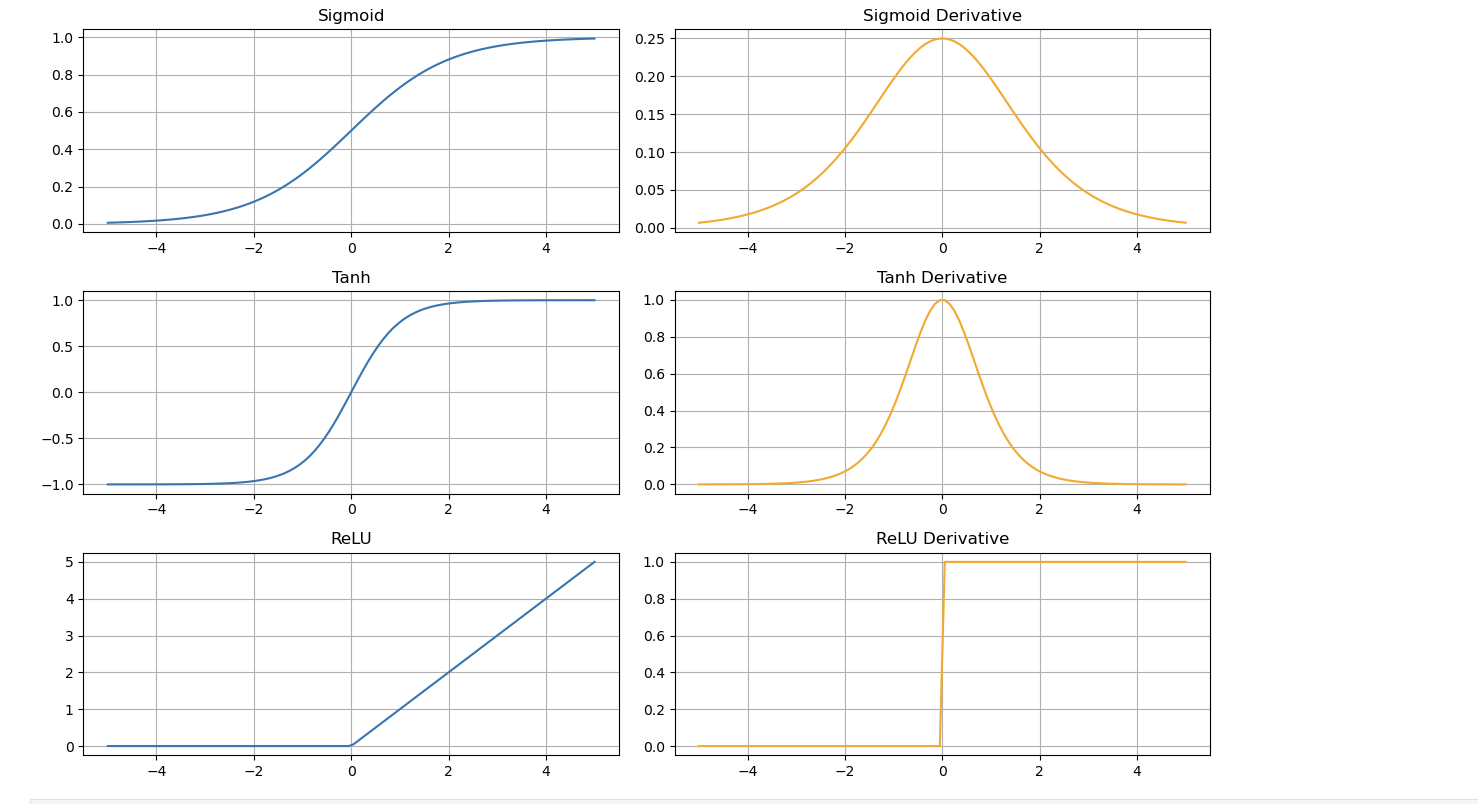
* Custom implementation of gradient descent (GD)
* Comparison of sigmoid, tanh, and ReLU activations
* Gradient checking (numerical vs. analytical)
* Neural training with MLPClassifier using SGD, Adam, and LBFGS optimizers
* Analysis of convergence, overfitting, and model capacity

**2. Experimental Setup**

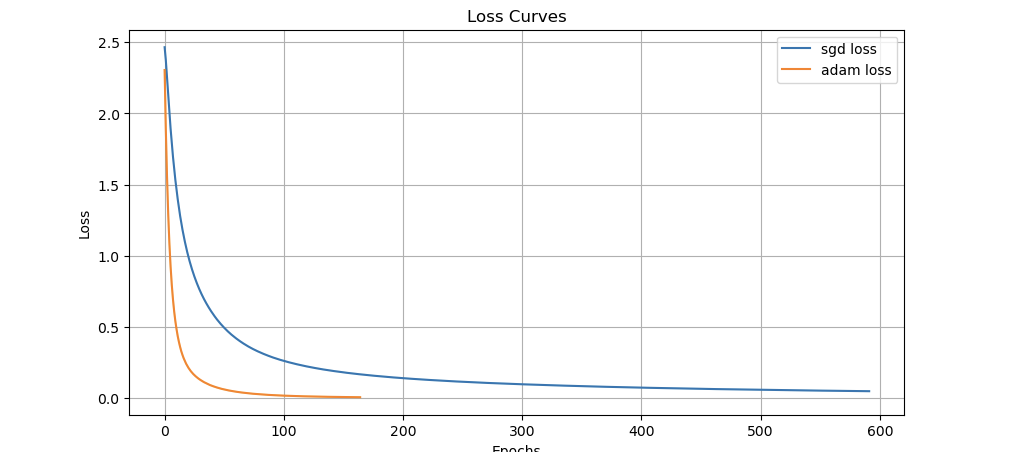
* **Dataset**: sklearn.datasets.load\_digits(), binary relabel: 0-4 versus 5-9
* **Preprocessing**: Standardization, train-test split
* **Frameworks**: NumPy, matplotlib, scikit-learn
* **Manual Methods**: Custom GD, activation/derivative implementation, and gradient checking
* **Network**: Feedforward MLPClassifier, tuned hidden layer size, tested with three solvers

**3. Plots and Visuals**

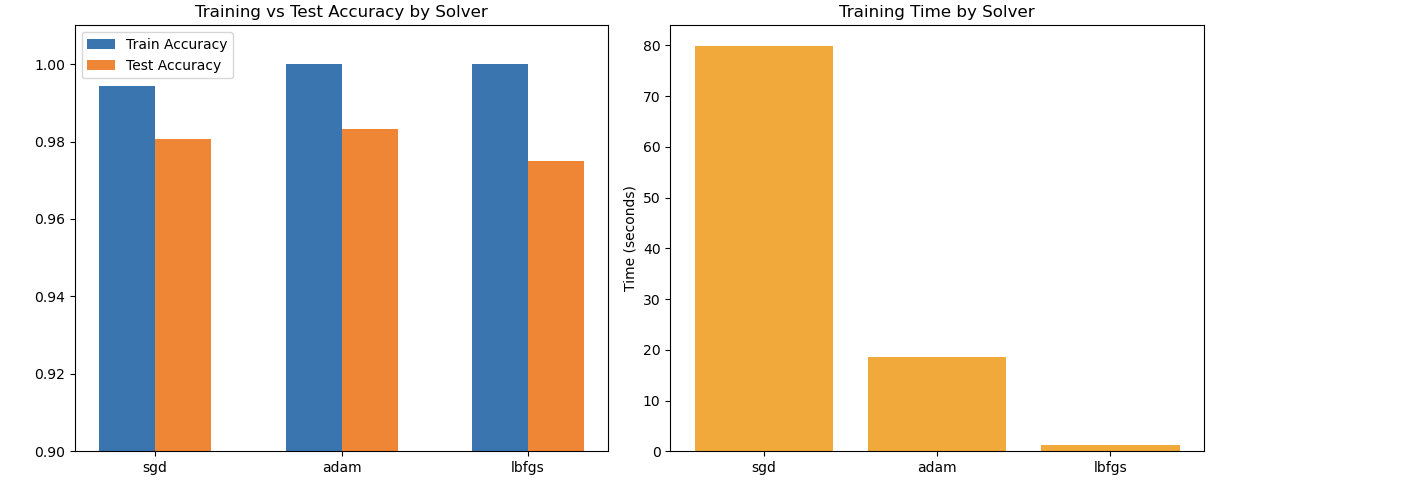
**Activation Functions & Gradients**

*Caption: Sigmoid, Tanh, and ReLU activation functions with their derivatives. ReLU provides non-saturating gradients, reducing vanishing gradient issues.*

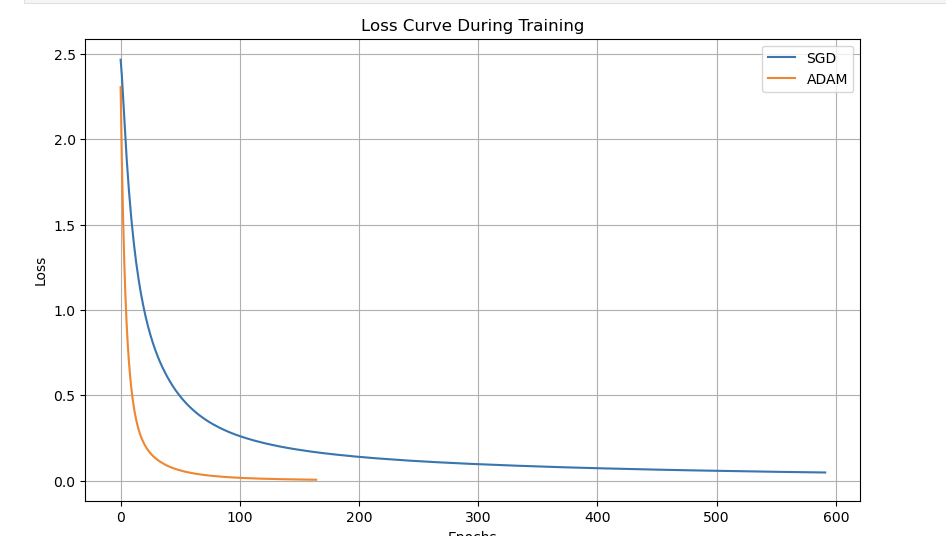
**Loss Curves (SGD vs. Adam)**

*Caption: Loss decreases faster and plateaus sooner with Adam compared to SGD, reflecting Adam’s adaptive learning rate advantage.*

**Accuracy and Training Time by Solver**

*Left: Training and test accuracy per solver. Right: Training time per solver. Adam and lbfgs reach high accuracy faster than SGD.*

**Individual Loss Curves**

*Caption: Adam consistently achieves lower loss in fewer epochs. SGD converges more slowly, LBFGS (shown in Table) was fastest overall.*

**4. Results Summary Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model Architecture** | **Optimizer** | **Training Accuracy** | **Test Accuracy** | **Training Time (s)** | **Notes/Observations** |
| 100 hidden units | sgd | 99.44% | 98.06% | 74.80 | Slowest, stable, minor overfit |
| 100 hidden units | adam | 100.0% | 98.33% | 20.52 | Fast, robust, top accuracy |
| 100 hidden units | lbfgs | 100.0% | 97.50% | 1.84 | Fastest, possible overfit |

**5. Analysis & Discussion**

* **Optimizers:**
  + **SGD:** Slowest to converge, but stable. Shows minor overfitting; suitable for small/medium datasets when memory is a concern.
  + **Adam:** Fast and robust; reaches high accuracy efficiently, less prone to get stuck in flat regions.
  + **LBFGS:** Lightning fast on tabular/structured data; not scalable to large sets but excellent for this task.
* **Activation Functions:**
  + **ReLU** outperformed sigmoid/tanh, especially in deeper models, due to strong gradients and rapid convergence.
* **Training Speed:** Adam and LBFGS required much less time to reach optimality compared to SGD.
* **Overfitting:** Slight overfitting as seen from train-test gap, particularly for LBFGS.
* **Gradient Checking:** Numerical and analytical gradients matched closely, confirming correct backpropagation.
* **Performance Bottlenecks:** Only SGD was slow; otherwise, manual gradients are bottleneck for custom code only.

**6. Conclusion**

* **Best:** Adam optimizer with ReLU for this binary classification task.
* **Overfitting Risks:** Monitor on deeper/larger nets.