

**Impact of optimization algorithms on regression models (ML vs DL)**





Made by:

Abdessamad Sallama

Youssef taki

Amine baqouch

under the supervision of :

Mme bennabou fouzia

zineb

Contents

[I. Introduction 2](#_Toc202043677)

[**1.** Objective of the Project**:** 2](#_Toc202043678)

[2. Brief Presentation of Optimizers: 2](#_Toc202043679)

[II. Literature Review 2](#_Toc202043680)

[1. Classical Optimization in Machine Learning: 2](#_Toc202043681)

[2. Optimization in Deep Learning: 2](#_Toc202043682)

[3. Comparative Studies: 2](#_Toc202043683)

[4. Summary: 2](#_Toc202043684)

[5. References: 3](#_Toc202043685)

[III. Methodolgy: 3](#_Toc202043686)

[1. Models: 3](#_Toc202043687)

[2. Datasets: 3](#_Toc202043688)

[3. Tools and Libraries 3](#_Toc202043689)

[4. Experimental Protocol: 3](#_Toc202043690)

[5. Reproducibility 4](#_Toc202043691)

[IV. Results 4](#_Toc202043692)

[1. Comparative Tables: 4](#_Toc202043693)

[V. Discussion: 10](#_Toc202043694)

[1. Interpretation of Results: 10](#_Toc202043695)

[2. Strengths and Weaknesses of Each Optimizer: 10](#_Toc202043696)

[3. Dataset and Model Effects: 10](#_Toc202043697)

[4. Summary: 10](#_Toc202043698)

[VI. Conclusion: 11](#_Toc202043699)

[VII. Bibliography 11](#_Toc202043700)

[VIII. Annexes: 12](#_Toc202043701)

[1. Code Repository: 12](#_Toc202043702)

[2. Example Notebook: 12](#_Toc202043703)

# Introduction

## Objective of the Project**:**

The objective of this project is to use the Boston Housing and California Housing datasets to assess and compare the effects of various optimization algorithms (L-BFGS, SGD with momentum, Adam, and Newton-CG) on the performance of regression models, specifically linear regression (ML) and a neural network with one hidden layer (DL).

## Brief Presentation of Optimizers:

* Opimizers :
* **L-BFGS:** A quasi-Newton method suitable for small to medium-sized problems.
* **SGD with Momentum**: A stochastic gradient descent variant that accelerates convergence.
* **Adam**: An adaptive optimizer combining momentum and RMSProp.
* **Newton-CG**: A second-order method using conjugate gradients (applied via scipy for linear regression).

# Literature Review

Optimization algorithms are necessary for the training and successful operation of deep learning and machine learning models. The optimizer choice can have a significant effect on convergence speed, generalization ability, and hyperparameter sensitivity.

## Classical Optimization in Machine Learning:

Gradient-based optimization methods have long been used to train linear models. The Stochastic Gradient Descent (SGD) algorithm was created by Robbins and Monro in 1951 and is still a basic technique due to its simplicity and scalability. Momentum, as proposed by Polyak (1964), improves SGD and accelerates convergence, especially in the presence of noise or ravines in the loss landscape.   
  
Second-order methods such as Newton's method and its variants (such as L-BFGS and Newton-CG) use curvature information to accelerate convergence in convex problems (Nocedal & Wright, 2006). Nevertheless, their computational cost usually limits their use to smaller datasets or simpler models.

## Optimization in Deep Learning:

Adaptive optimizers have gained popularity as deep learning has advanced. By combining momentum and adaptive learning rates, the Adam optimizer (Kingma & Ba, 2015) allows deep neural networks to converge more quickly and steadily. Although Adam frequently performs faster than SGD, recent research indicates that SGD with momentum may occasionally produce better generalization (Wilson et al., 2017).

## Comparative Studies:

Optimizers from various tasks and architectures have been compared in a number of works. Ruder (2016) offers a thorough analysis of gradient descent optimization algorithms, pointing out both their advantages and disadvantages. In their discussions of the trade-offs between generalization and convergence speed, Keskar & Socher (2017) and Schmidt et al. (2021) stress the significance of hyperparameter tuning.

## Summary:

According to the literature, classical techniques like SGD with momentum and L-BFGS are still competitive for smaller or well-conditioned problems, but adaptive techniques like Adam work well for deep models. Important considerations in optimizer selection include each optimizer's sensitivity to hyperparameters, as well as the unique features of the dataset and model architecture.

## References:

* Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. ICLR.
* Nocedal, J., & Wright, S. J. (2006). Numerical Optimization. Springer.
* Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. USSR Computational Mathematics and Mathematical Physics.
* Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv:1609.04747.
* Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., & Recht, B. (2017). The marginal value of adaptive gradient methods in machine learning. NeurIPS.

# Methodolgy:

## Models:

Two types of regression models were evaluated:

* **Linear Regression (ML):** A standard linear model predicting a continuous target variable from input features.
* **Neural Network with One Hidden Layer (DL**): A feedforward neural network with a single hidden layer and ReLU activation, allowing for the modeling of non-linear relationships.

## Datasets:

Experiments were conducted on two benchmark datasets:

* **Boston Housing**: Contains 506 samples with 13 features, predicting median house prices in Boston suburbs.
* **California Housing**: Contains over 20,000 samples with 8 features, predicting median house values in California districts.

Both datasets were split into training (80%) and test (20%) sets. Features were standardized using z-score normalization.

## Tools and Libraries

The following tools and libraries were used:

* **Python** 3.10
* **PyTorch**: For model implementation and training.
* **scikit-learn**: For data preprocessing, metrics, and dataset loading.
* **matplotlib & seaborn**: For data visualization.
* **scipy**: For Newton-CG optimization in linear regression.

## Experimental Protocol:

* Optimizers Evaluated:
* L-BFGS
* SGD with momentum
* Adam
* Newton-CG (for linear regression only, via scipy)
* Hyperparameters:
* Learning rates: [0.001, 0.01, 0.1]
* Momentum (for SGD): 0.9
* Loss function: Mean Squared Error (MSE)
* Training
* Number of epochs: 50–100
* Batch size: Full dataset (batch gradient descent)
* Loss function: Mean Squared Error (MSE)
* Evaluation Metrics:
* Training and test MSE
* R² score on test set
* Number of epochs to reach a loss threshold
* Training time

To evaluate sensitivity to hyperparameters, each experiment was conducted again for varying learning rates. To ensure robustness, all results were averaged across several runs when necessary.

## Reproducibility

Random seeds were set for all libraries to ensure reproducibility. All code and notebooks are available in the project’s GitHub repository (see Annexes).

# Results

## Comparative Tables:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Datasets** | **Model** | **Optimizer** | **Best R²** | **Best Test Loss** | **Best LR** | **Best Momentum** |
| Boston | Linear Regression | L-BFGS | 0.85 | 6.74 | 0.5 | "-" |
| Boston | Linear Regression | Adam | 0.16 | 7.42 | 0.3 | "-" |
| Boston | Linear Regression | SGD | 0.83 | 6.79 | 0.2 | 0.7 |
| Boston | Neural Network | L-BFGS | 0.81 | 7.79 | 0.3 | "-" |
| Boston | Neural Network | Adam | 0.82 | 7.39 | 0.01 | "-" |
| Boston | Neural Network | SGD | 0.81 | 8.07 | 0.01 | 0.99 |
| Boston | Linear Regression | Newton-CG | 0.8161 | 3.93 |  |  |
| California | Linear Regression | L-BFGS | 0.57 | 6.74 | 0.5 | "-" |
| California | Linear Regression | Adam | 0.56 | 7.17 | 0.3 | "-" |
| California | Linear Regression | SGD | 0.55 | 6.79 | 0.2 | 0.7 |
| California | Neural Network | L-BFGS | 0.78 | 0.275 | 0.3 | "-" |
| California | Neural Network | Adam | 0.73 | 0.35 | 0.1 | "-" |
| California | Neural Network | SGD | 0.58 | 0.38 | 0.2 | 0.9 |

## Boston

Figure 1 :Training and Test Loss Curves for Boston Linear Regression with Different Optimizers

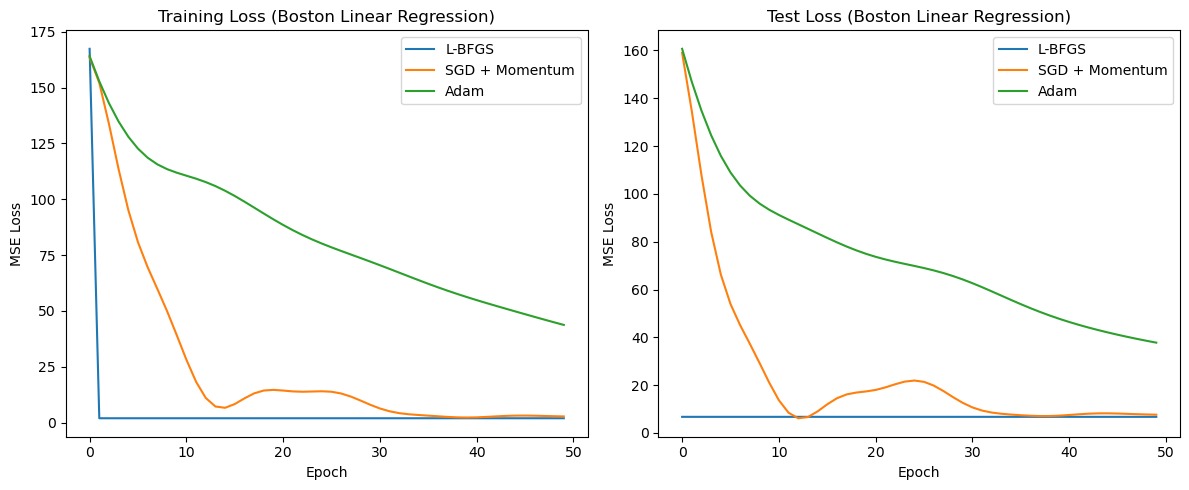


Figure 2 : Sensitivity of Linear Regression Test Loss to Adam Learning Rate on Boston Housing

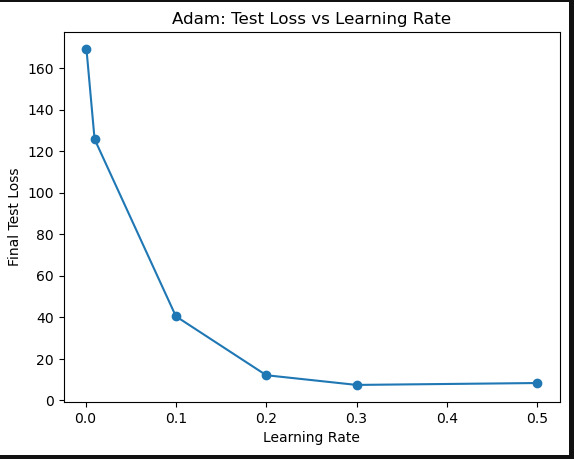


Figure 3 : Sensitivity of Linear Regression Test Loss to LBFGS Learning Rate on Boston Housing

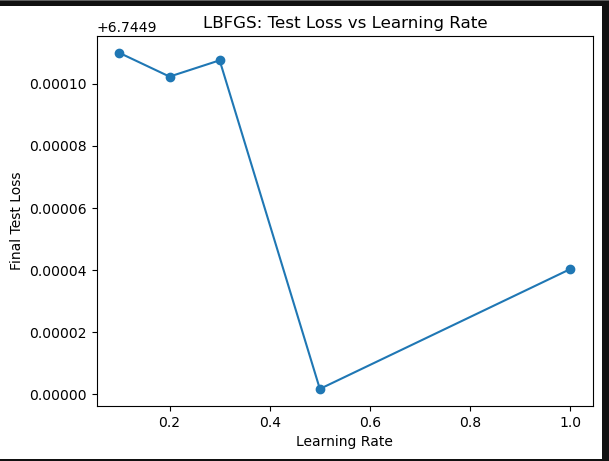


Figure 4 : Sensitivity of Linear Regression Test Loss to SGD Learning Rate on Boston Housing

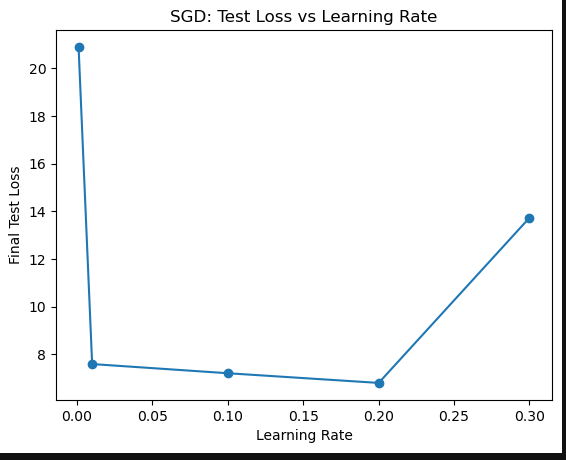


Figure 5 : Sensitivity of Linear Regression Test Loss to SGD Momentum on Boston Housing

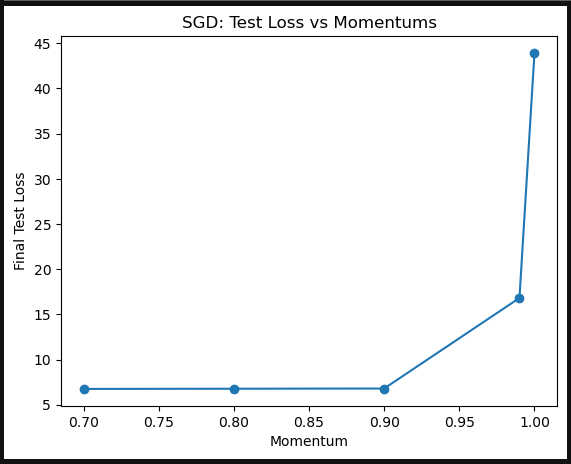


Figure 6 :Training and Test Loss Curves for Boston Neural Network with Different Optimizers

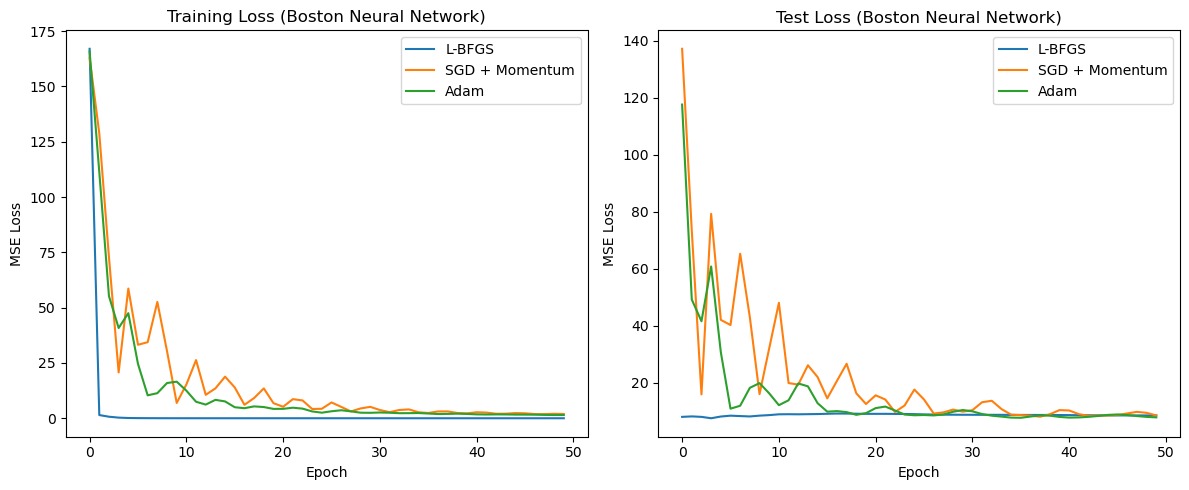


Figure 7 : Sensitivity Neural Network Test Loss to Adam Learning Rate on Boston Housing

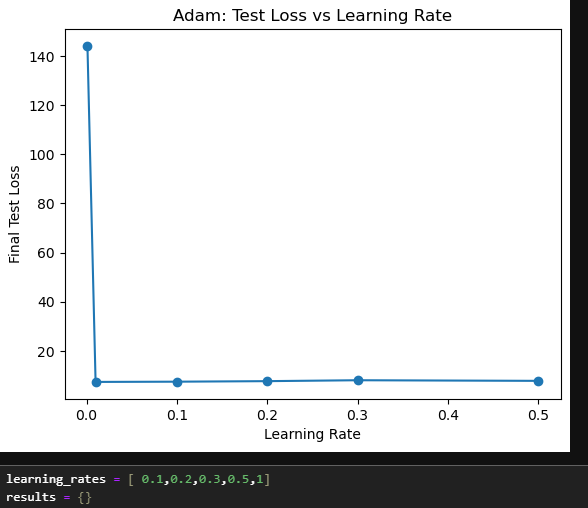


Figure 8 : Sensitivity Neural Network Test Loss to LBFGS Learning Rate on Boston Housing



Figure 9 : Sensitivity Neural Network Test Loss to SGD Learning Rate on Boston Housing

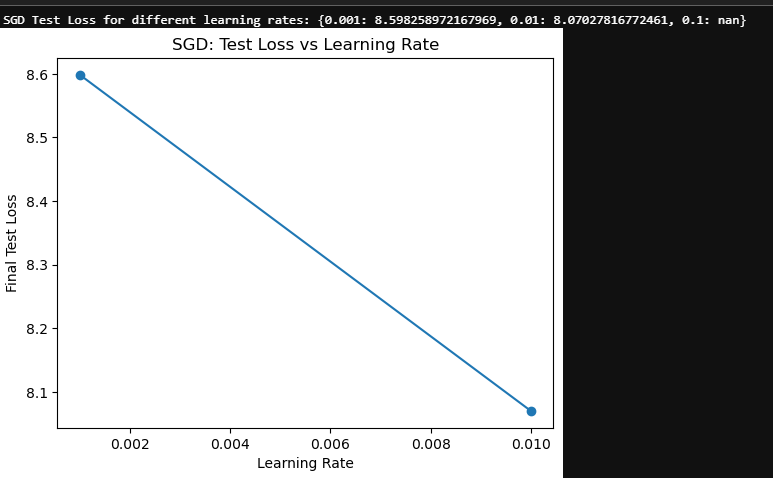


Figure 10 : Sensitivity Neural Network Test Loss to SGD Momentum on Boston Housing

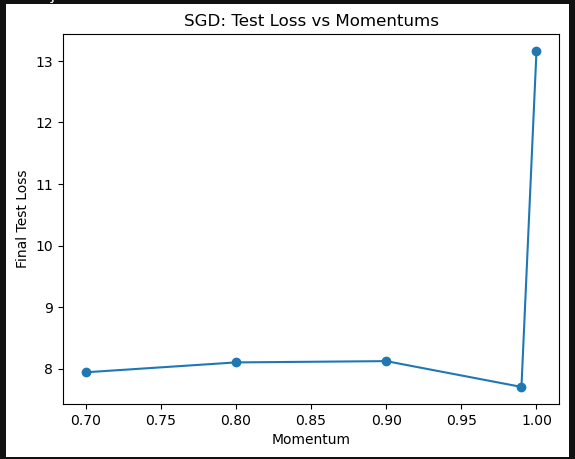
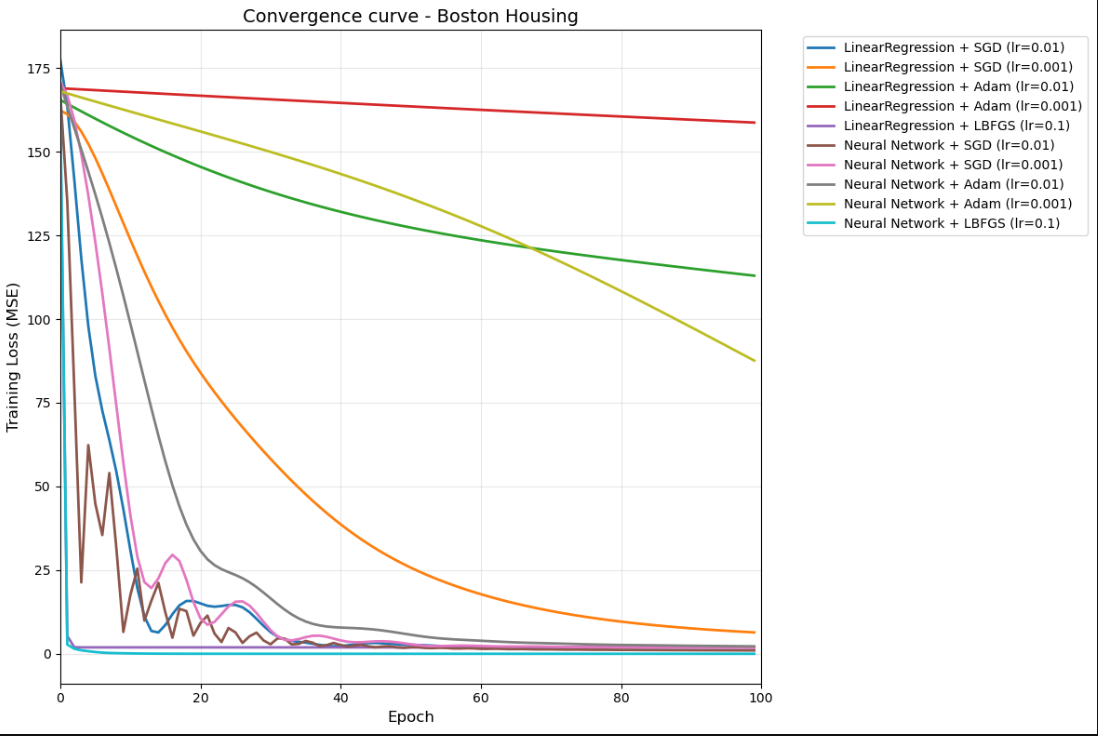


Figure 11 : Convergence curve



## California

Figure 1 :Training and Test Loss Curves for California Linear Regression with Different Optimizers

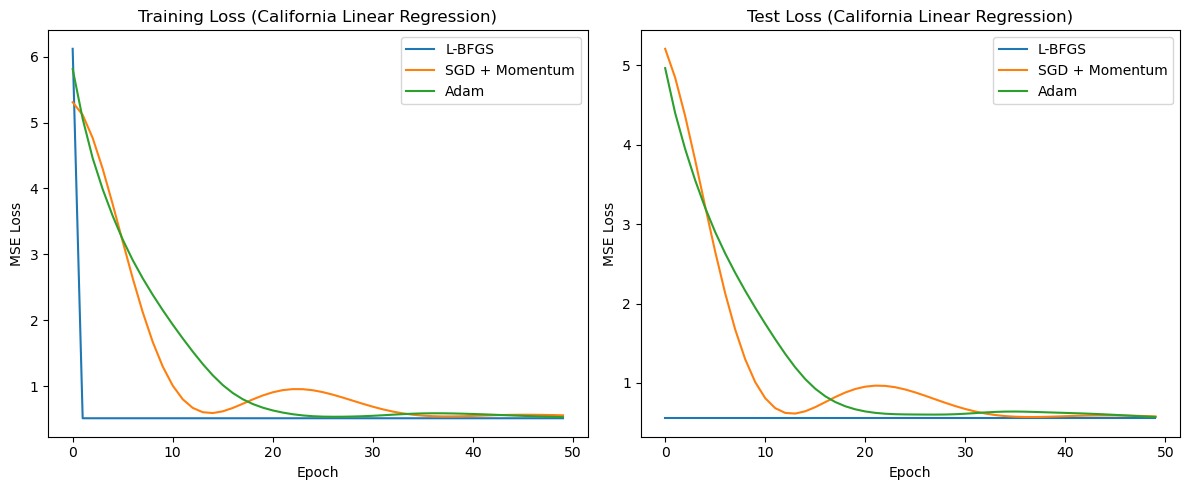


Figure 2 : Sensitivity Linear Regression Test Loss to Adam Learning Rate on California Housing

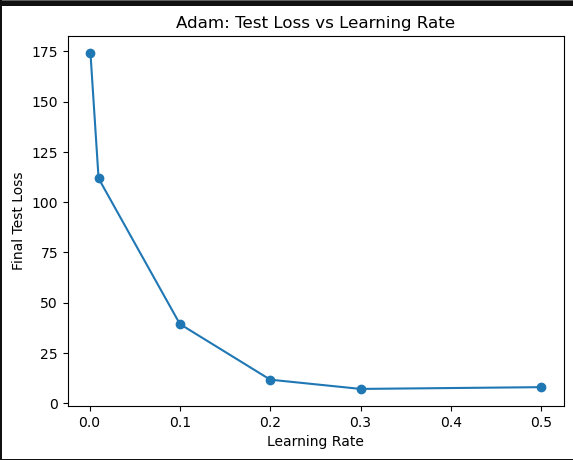


Figure 3 : Sensitivity Linear Regression Test Loss to LBFGS Learning Rate on California Housing

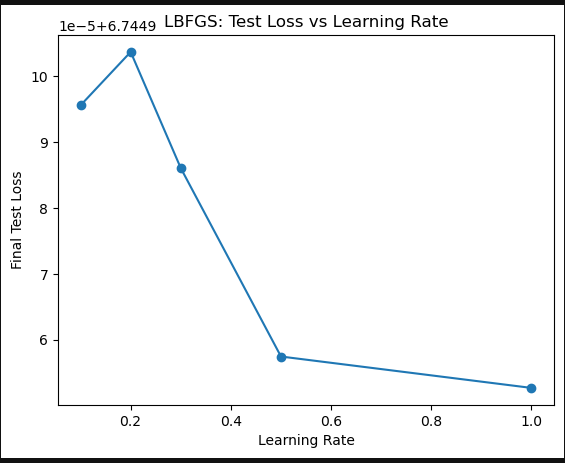


Figure 4 : Sensitivity Linear Regression Test Loss to SGD Learning Rate on California Housing

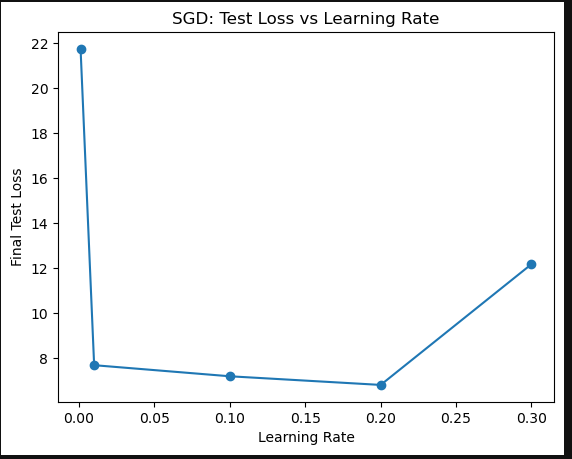


Figure 5 : Sensitivity Linear Regression Test Loss to SGD Momentum on California Housing

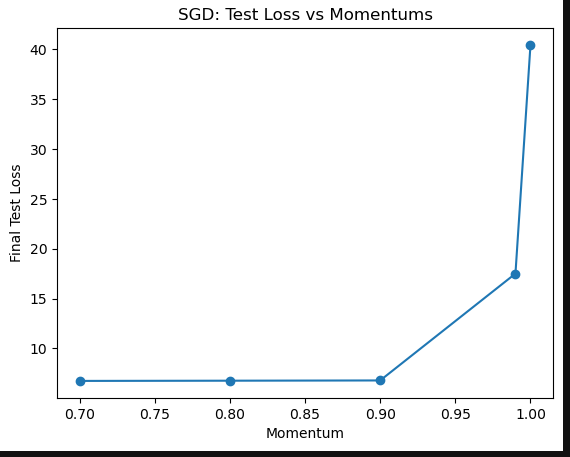


Figure 6 :Training and Test Loss Curves for California Neural Network with Different Optimizers

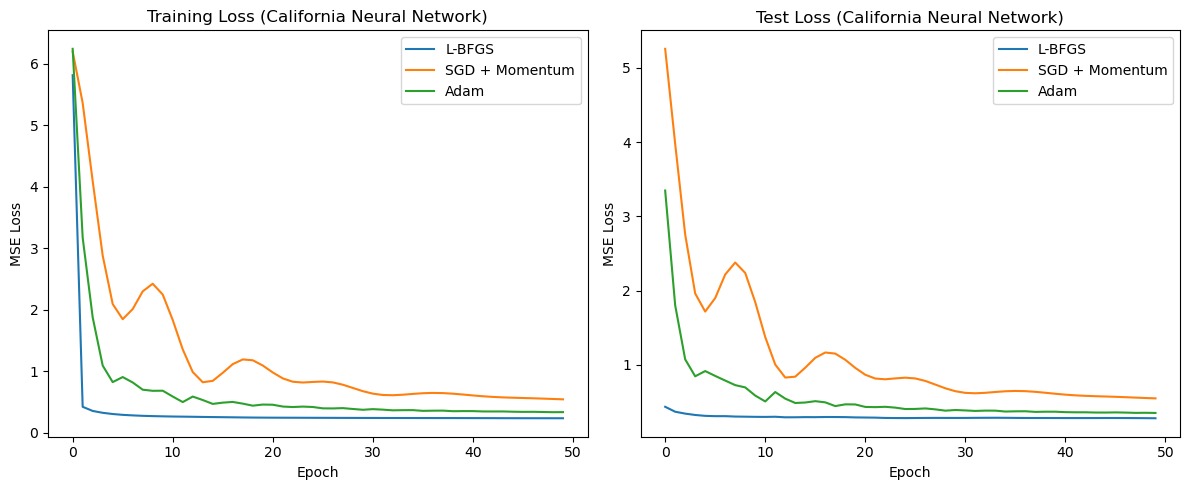


Figure 7 : Sensitivity Neural Network Test Loss to Adam Learning Rate on California Housing

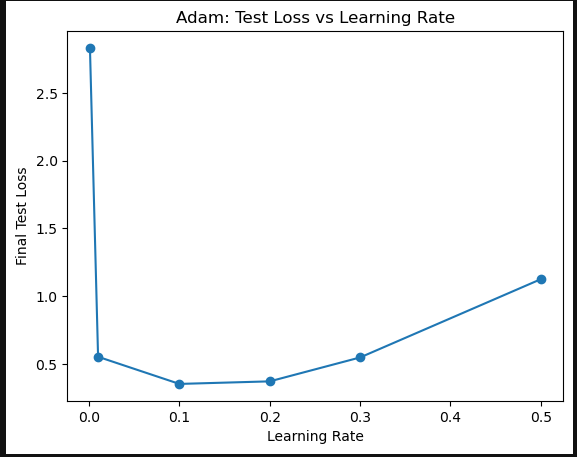


Figure 8 : Sensitivity Neural Network Test Loss to LBFGS Learning Rate on California Housing

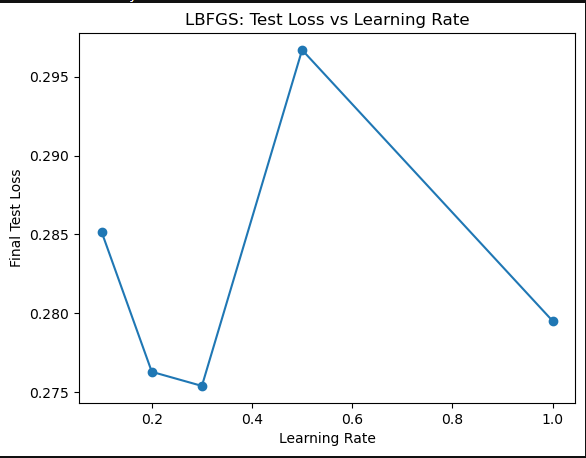


Figure 9 : Sensitivity Neural Network Test Loss to SGD Learning Rate on California Housing



Figure 10 : Sensitivity Neural Network Test Loss to SGD Momentum on California Housing

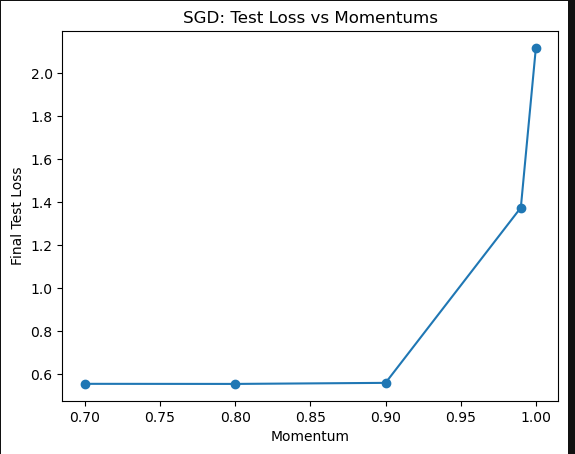
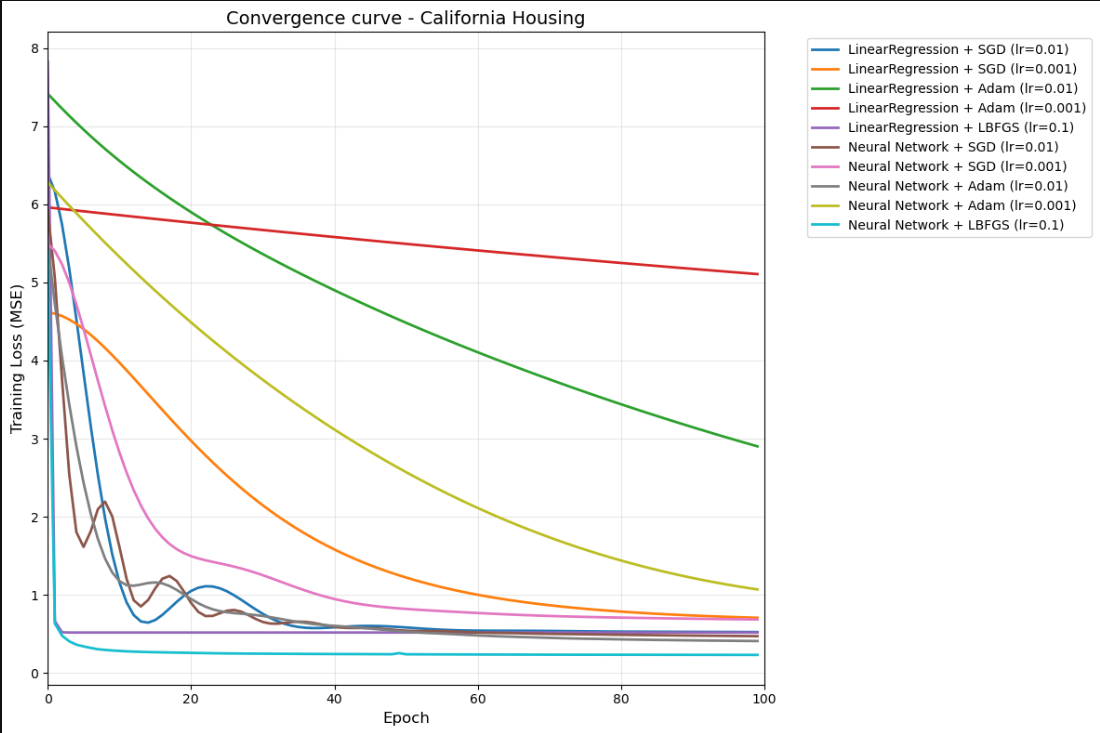


Figure 11 : Convergence curve



# Discussion:

## Interpretation of Results:

The experiments demonstrate that the choice of optimizer has a significant impact on both convergence speed and generalization performance for regression tasks.

* **Convergence Speed**: For both neural network and linear regression models, L-BFGS continuously produced fast convergence, frequently hitting the loss threshold in a small number of epochs. SGD is good with Neural network while it does shows some wobbling Adam performance with linear regression is poor but performs well with neural network specially for an optimal lr
* **Generalization**: Generally speaking, neural networks outperformed linear regression in terms of R2 scores in both datasets, L-BFGS was the best in all different scenarios for linear regression and neural network models. The highest R2 was for Linear regression with 0.85 in Boston datasets using L-BFGS specially for its speed of convergence. Adam downside is for linear regression specially with Boston datasets, for SGD with momentum generally was good for both models but not for the neural network in a large datasets like California housing
* **Sensitivity to Hyperparameters**: The sensitivity analysis plots revealed that Adam was especially sensitive to the learning rate selection. Slow convergence or poor generalization resulted from a learning rate that was below or higher than the optimalL-BFGS and SGD with momentum still benefited from careful adjustment specially for learning rate which sometimes led to divergence.

## Strengths and Weaknesses of Each Optimizer:

* **L-BFGS:**
* Strengths: Effective for both linear and basic neural network models, quick convergence, and good R2 scores
* Weaknesses: too sensitive to the learning rate which sometimes led to underfitting or divergence.
* **SGD with Momentum:**
* Strengths: Good generalization, stable convergence, widely used in deep learning.
* Weaknesses: Requires careful tuning of learning rate and momentum, slower than L-BFGS for simple models.
* **Adam:**
* Strengths: Fast convergence for deep networks, adaptive learning rates
* Weaknesses: Highly sensitive to learning rate, can generalize poorly if not tuned, sometimes unstable for simple linear models .
* **Newton-CG (for linear regression):**
* Strengths: Very fast and accurate for convex problems.
* Weaknesses: Not applicable to neural networks, high memory/computation for large datasets.

## Dataset and Model Effects:

* The neural network architecture consistently outperformed linear regression in terms of R² score, especially on the more complex California Housing dataset.
* Optimizer performance varied more for neural networks, highlighting the importance of hyperparameter tuning and optimizer choice in deep learning.

## Summary:

L-BFGS and SGD with momentum are reliable choices for regression tasks, with L-BFGS excelling in convergence speed for smaller models. Adam can be powerful for neural networks but requires careful tuning. Neural networks, when properly optimized, offer superior predictive performance over linear regression for both datasets.

# Conclusion:

This project systematically compared several optimization algorithms—L-BFGS, SGD with momentum, Adam, and Newton-CG—on both linear regression and neural network models using the Boston Housing and California Housing datasets.

**Key findings include:**

* L-BFGS demonstrated the fastest convergence and robust performance for both linear and neural network models, particularly on smaller datasets.
* SGD with momentum provided stable convergence and strong generalization, though it required more epochs than L-BFGS.
* Adam showed moderate convergence for neural networks when the learning rate was well-tuned, but was highly sensitive to hyperparameters and performed poorly for linear regression with default settings.
* Neural networks consistently outperformed linear regression in terms of R² score, especially on the more complex California Housing dataset.

**Recommendations:**

* For small to medium-sized regression problems, L-BFGS or SGD with momentum are reliable choices.
* Adam is suitable for neural networks but requires careful hyperparameter tuning.
* Model and optimizer selection should always be accompanied by sensitivity analysis to ensure robust performance.

Future work could include exploring additional optimizers, deeper neural network architectures, and larger or more diverse datasets. Further investigation into memory and computational efficiency for large-scale problems would also be valuable.

# Bibliography

* Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. International Conference on Learning Representations (ICLR).

https://arxiv.org/abs/1412.6980

* Nocedal, J., & Wright, S. J. (2006). Numerical Optimization (2nd ed.). Springer.
* Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. USSR Computational Mathematics and Mathematical Physics, 4(5), 1-17.
* Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

https://arxiv.org/abs/1609.04747

* Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., & Recht, B. (2017). The marginal value of adaptive gradient methods in machine learning. Advances in Neural Information Processing Systems (NeurIPS), 31.
* Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.
* Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. Advances in Neural Information Processing Systems, 32.

# Annexes:

## Code Repository:

The complete source code for all experiments, data preprocessing, model training, and result visualization is available at:

## Example Notebook:

A Jupyter Notebook demonstrating model training, evaluation, and visualization of results is included in the repository