Heart Disease Prediction: A Comparative Study of Optimizers Performance in Deep Neural Network

Chisom Chibuike[1] ,Adeyinka Ogunsanya[2]
University of Nigeria[1] ,SAIL Innovation Lab[2]

NEURAL INFORMATION PROCESSING SYSTEMS

Introduction

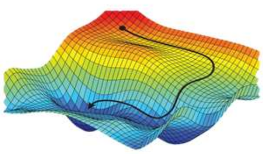
Heart disease, a leading global cause of death, necessitates effective predictive models to enable early diagnosis and targeted prevention. While deep learning models have demonstrated significant potential in heart disease prediction, the critical role of optimizers, a key component that directly affect model performance, convergence, and generalization remains underexplored. This work systematically evaluates the performance of ten widely-used optimizers on a heart disease dataset, using metrics such as convergence speed, stability, and other Machine learning metric such as AUC, precision, recall. By addressing this gap, we aim to advance the understanding of optimizer selection and uncover the trade-offs involved, thereby offering actionable insights to improve the robustness and reliability of deep learning in healthcare.

Related Work

Heart disease prediction has been a significant area of research, with numerous studies leveraging deep learning and other machine learning techniques to improve diagnosis and risk prediction. Parmar (2020) proposed a heart disease prediction model using the UCI Heart Disease dataset and demonstrated the effectiveness of Talos hyperparameter optimization in enhancing model performance. Similarly, García-Ordás et al. (2023) utilized deep learning methods with feature augmentation, achieving a 4.4% improvement over state-of-the-art approaches and attaining a precision of 90%. These studies underscore the potential of deep learning in advancing heart disease prediction. However, recent works have primarily focused on model architectures and feature engineering, overlooking the role of optimization algorithms. Our work addresses this gap by systematically analyzing the performance of various optimizers, highlighting their impact on convergence speed, stability, and overall model performance.

Optimization Algorithms

Optimization is at the heart of deep learning, guiding neural networks to achieve optimal predictive performance by minimizing the loss function. This process involves iteratively adjusting model parameters (θ) through gradient-based updates.



Fid 1. Illustration of optimization

Mathematically, the gradient-based Optimization Process can be expressed as:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J(\theta)$$

Where:
θ_t is the model parameter at step t
η is the learning rate,
$\nabla_\theta J(\theta)$ is the gradient of the loss function $J(\theta)$ wrt to the parameters.

While the foundational idea remains consistent, optimizers vary in how they handle learning rates, momentum, and gradient accumulation, e.g., SGD (Stochastic Gradient Descent) uses a fixed learning rate and stochastic updates for faster computation. While optimization algorithms play a pivotal role in balancing faster convergence and generalization of deep learning models, certain factors influence the choice of optimizers in a neural network training pipeline, such as the size and structure of the dataset involved. In this work, we evaluate the 10 different optimizers stated below in terms of:

- Convergence Speed: The number of epochs required to minimize the training loss.
- Stability: The standard deviation of the loss across training epochs
- Predictive Performance: Assessed through metrics like AUC-ROC, precision, and recall.
- Generalization Ability: The model's performance on unseen test data, balancing underfitting and overfitting.

Optimizers

- Adaptive Moment Estimation (Adam)
- Adam with Decoupled Weight Decay Regularization (AdamW)
- Adaptive Moment Estimation Max (Adamax)
- Adaptive Moment Estimation with Stabilized Updates (AMSGrad)
- Root Mean Square Propagation (RMSprop)
- Adaptive Gradient Algorithm (Adagrad)
- Adaptive Delta Update (Adadelta)
- Stochastic Gradient Descent (SGD) Stochastic
- Stochastic Gradient Descent with Nesterov Accelerated Momentum
- Nesterov-accelerated Adaptive Moment Estimation (Nadam)

Heart Disease Dataset

The dataset for this study, sourced from a public Kaggle repository, comprises 1,190 patient records and includes 12 features representing key demographic, clinical, and lifestyle factors related to heart disease. The target variable is binary, indicating the presence (1) or absence (0) of heart disease. Notable features include age, sex, chest pain type, resting blood pressure, cholesterol levels, and maximum heart rate.

Fig. 2 A sample view of the heart

Before analysis, the dataset was preprocessing to ensure it was clean, consistent, and suitable for training the deep learning model. Exploratory Data Analysis (EDA) revealed no significant imbalance in the target variable, confirming a well-distributed dataset. However, certain issues required intervention. For instance, the "Fasting Blood Sugar" feature was excluded due to limited variability, as 75% of its values were zero. Additionally, medically invalid zero values in "Cholesterol" and "Resting Blood Pressure" were replaced with their median values, while duplicate entries were removed to maintain data integrity.

To prepare the data for modeling, numerical features were normalized using RobustScaler to mitigate the influence of outliers. Categorical variables, such as "Chest Pain Type," were pre-encoded, and their consistency was verified. Finally, the dataset was split into training (70%) and testing (30%) subsets, with 20% of the training data further allocated for validation. These preprocessing steps ensured the dataset was optimized for robust and reliable model performance.

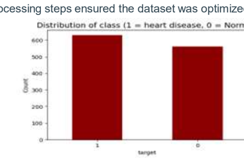


Fig. 3 An overview of the Distribution of classes in the Dataset

Methodology

The methodology involved investigates the application of a Deep Neural Network (DNN) in predicting heart disease, with a focus on comparing the performance of ten optimizers: Adam, AdamW, Adamax, Nadam, AMSGrad, SGD, SGD with Nesterov Momentum, Adagrad, Adadelta, and RMSprop. The training was carried out in steps structured as follows:

1. DNN Architecture:
- The model includes an input layer, six fully connected hidden layers with ReLU activation, and a sigmoid-activated output layer for binary classification.
- Dropout regularization is applied in later stages to reduce overfitting.

2. Training Procedure: The model is trained for up to 50 epochs using binary cross-entropy loss function and two training phases are implemented:
- Phase 1: Initial evaluation of optimizers without hyperparameter tuning to assess raw performance.
- Phase 2: The best optimizer from Phase 1 undergoes additional training with dropout, hyperparameter tuning and early stopping to improve generalization.

3. Evaluation Metrics: The performance of the optimizers were evaluated using the following metrics
- Convergence Speed: Number of epochs required for the training loss to stabilize.
- Stability: Fluctuations in validation loss during training. (evaluated based on the standard deviation of the validation loss
- Performance: Classification metrics including precision, recall, and Area Under the Curve (AUC).

4. Selection and Refinement:
- The optimizer achieving the best balance of speed, stability, and predictive accuracy is selected for hyperparameter tuning to further improve performance.

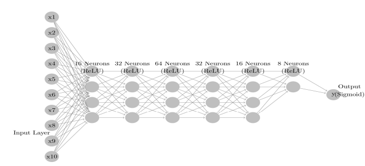


Fig. 4 An overview of the DNN Architecture

Experiments and Results

Optimizer	*Final Training Loss*	*Final Validation Loss*	*Convergence Epoch*	*Stability (Validation Loss Std Dev)*	*Final Precision*	*Final Recall*	*Final AUC*
SGD	0.3686	0.5464	48.0000	0.0568	0.7436	0.7733	0.8235
ADAM	0.0977	0.8974	10.0000	0.1247	0.7613	0.8933	0.8680
RMSProp	0.1675	0.8041	10.0000	0.1117	0.7303	0.8667	0.8414
Adagrad	0.6835	0.6869	49.0000	0.0024	0.5859	0.8667	0.7358
Adadelta	0.6945	0.6944	49.0000	0.0001	0.3947	0.4000	0.4301
Adamax	0.2394	0.5230	17.0000	0.0288	0.7901	0.8533	0.8777
Nadam	0.0878	0.8899	16.0000	0.0960	0.7500	0.8800	0.8539
AMSGrad	0.1192	0.9574	3.0000	0.1407	0.7614	0.8933	0.8556
AdamW	0.1061	1.0600	8.0000	0.1846	0.7386	0.8667	0.8412
SGD Nesterov	0.3980	0.5564	47.0000	0.0470	0.7125	0.7600	0.8297

TABLE 1 Showing the results of the performance of each optimizer

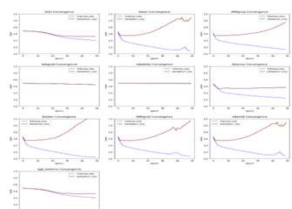


Fig. 5 An overview of Convergence of Optimizers

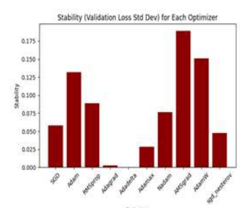


Fig. 6 Stability Performance of Optimizers

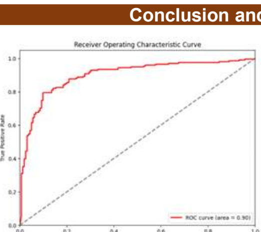
The experiment highlighted several key insights into the performance of different optimizers for training a deep neural network (DNN) using a heart disease dataset. The primary aspects analyzed include convergence speed, stability, and final performance metrics.

1. Convergence Speed: Adam, RMSProp AMSGrad, and AdamW achieved the fastest convergence, requiring only 3-10 epochs to reach optimal performance.
2. Stability: Adagrad ,Adadelta and Adamax exhibited the most stable training, indicating minimal fluctuation during training.
3. Final Performance Metrics: In terms of precision, Adamax performed the best, with 0.79 precision. Final recall was highest for Adam and AMSGrad, indicating strong performance in identifying positive instances. Final AUC scores showed a similar trend, with Adamax leading.

- Trade-off between Convergence Speed and Stability

The results demonstrate a clear trade-off: faster optimizers like Adam, AMSGrad and AdamW offer quick convergence but at the cost of stability, while Adagrad and Adadelta prioritize stability, resulting in slower convergence. Adamax is seen to be the most effective optimizer for this task as it balances performance across metrics.

Conclusion and Area For Further Studies



In summary, our experiments demonstrate that Adamax is the most effective optimizer for this heart disease prediction task. It is stable and achieves a high AUC of 0.90, precision of 0.863, and recall of 0.885 after hyper parameter tunning. These findings underscore the importance of careful selection of optimizers for healthcare-related and machine learning tasks.

Future Work

Future research should explore the role of neural network architectures in influencing optimizer performance, such as their interactions with deeper layers or attention mechanisms. Additionally, future work can focus on developing novel optimizers that balance convergence speed and stability without trade-offs. Emphasis should also be placed on creating efficient techniques for selecting the most suitable optimizers for specific deep learning models, ensuring optimal performance across diverse tasks.

References

Parmar, Mahesh. (2020). Heart Diseases Prediction using Deep Learning Neural Network Model. International Journal of Innovative Technology and Exploring Engineering. 9. 2244-2248. 10.35940/ijitee.C9009.019320.

García-Ordás, M.T., Bayón-Gutiérrez, M., Benavides, C. et al. Heart disease risk prediction using deep learning techniques with feature augmentation. Multimed Tools Appl 82, 31759–31773 (2023). https://doi.org/10.1007/s11042-023-14817-z

Mustapha, A., Lachgar, M., and Ali, K. An Overview ofGradient Descent Algorithm Optimization in MachineLearning: Application in the Ophthalmology Field, pp.349–359. 06 2020. ISBN 978-3-030-45182-0. doi: 10.1007/978-3-030-45183-7 27.

Shaziya, H. A study of the optimization algorithms in deeplearning. 03 2020a. doi: 10.1109/ICISC44355.2019.9036442.

World Health Organization. Cardiovascular dis-eases, 2023. URL https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds). Accessed:2023-09-16