Article

SHAP Informed Neural Network

Jarrod Graham * and Victor S. Sheng

Department of Computer Science, College of Engineering, Texas Tech University, Lubbock, TX 79409, USA; victor.sheng@ttu.edu

* Correspondence: jarrod.graham@ttu.edu

Abstract: In the context of neural network optimization, this study explores the performance and computational efficiency of learning rate adjustment strategies applied with Adam and SGD optimizers. Methods evaluated include exponential annealing, step decay, and SHAP-informed adjustments across three datasets: Breast Cancer, Diabetes, and California Housing. The SHAP-informed adjustments integrate feature importance metrics derived from cooperative game theory, either scaling the global learning rate or directly modifying gradients of first-layer parameters. A comprehensive grid search was conducted to optimize the hyperparameters, and performance was assessed using metrics such as test loss, RMSE, R^2 score, accuracy, and training time. Results revealed that while step decay consistently delivered strong performance across datasets, SHAP-informed methods often demonstrated even higher accuracy and generalization, such as SHAP achieving the lowest test loss and RMSE on the California Housing dataset. However, the computational overhead of SHAP-based approaches was significant, particularly in targeted gradient adjustments. This study highlights the potential of SHAP-informed methods to guide optimization processes through feature-level insights, offering advantages in data with complex feature interactions. Despite computational challenges, these methods provide a foundation for exploring how feature importance can inform neural network training, presenting promising directions for future research on scalable and efficient optimization techniques.

Keywords: SHAP; Adam optimizer; learning rate adjustments; neural networks; grid search; performance evaluation

MSC: 68T05

check for updates

Academic Editors: Zhen Wang and Jüri Majak

Received: 1 December 2024
Revised: 5 February 2025
Accepted: 28 February 2025
Published: 4 March 2025

Citation: Graham, J.; Sheng, V.S. SHAP Informed Neural Network. *Mathematics* **2025**, *13*, 849. https://doi.org/10.3390/math13050849

1. Introduction

1.1. Background

The optimization of neural networks remains a pivotal aspect of deep learning research and applications [1]. Among the various components influencing the training of neural networks, the choice of the optimizer and the learning rate strategy are paramount. These elements dictate the efficiency, speed, and performance of the learning process, impacting the model's ability to generalize from the training data to unseen test data.

The Adam (Adaptive Moment Estimation) optimizer, introduced by Kingma and Ba in 2014, has become one of the most widely used optimization algorithms in deep learning. Adam combines the advantages of two other popular methods: AdaGrad, which works well with sparse gradients, and RMSProp, which adjusts the learning rate based on recent gradient magnitudes [2]. By maintaining per-parameter learning rates that are adapted based on the first and second moments of the gradients, Adam optimizes the learning process by converging faster and more efficiently, particularly in complex, large-scale problems [3,4]. Alongside Adam, SGD (Stochastic Gradient Descent) remains a widely used optimizer due to its simplicity, scalability, and ability to avoid certain overfitting issues when combined with effective regularization methods. Both Adam and SGD are commonly used in practice, and their performance can often be improved with effective learning rate strategies.

Despite the efficacy of Adam and SGD, there is ongoing research to further enhance their performance through complementary techniques. This study explores the integration of global learning rate and gradient adjustment methods with the Adam and SGD optimizers to assess their collective impact on neural network performance.

Learning rate adjustment methods are designed to control the training process by modifying the magnitude of updates to the model's parameters during backpropagation. These adjustments help the optimization process converge more efficiently and avoid issues such as overshooting or becoming trapped in local minima. In this study, three specific learning rate adjustment methods are examined:

1. Learning Rate Annealing: Learning rate annealing gradually decreases the learning rate according to a predefined schedule, typically following an exponential decay. This method aims to fine-tune the model's weights as training progresses, thereby reducing the risk of overshooting and aiding in more precise convergence [5].
2. Step Decay: In step decay, the learning rate is reduced by a factor at specified intervals (epochs). This approach is straightforward yet effective, as it helps the model stabilize and converge more accurately during later stages of training [6].
3. SHAP-Informed Adjustments: SHAP (SHapley Additive exPlanations) values, based on cooperative game theory, provide a unified measure of feature importance [7]. These adjustments aim to incorporate feature importance into the optimization process to guide parameter updates. In this study, two distinct SHAP-informed adjustment methods are explored:
    a. SHAP-Informed Learning Rate Adjustment: This approach scales the global learning rate based on the aggregated importance of features. At specified intervals during training, SHAP values are computed and normalized to adjust the magnitude of parameter updates uniformly across all layers. This adjustment serves as a dataset-wide optimization influence, leveraging feature-derived insights to guide global learning rate adaptation.
    b. SHAP-Informed Gradient Adjustment: In contrast, this method applies SHAP values directly to the gradients of individual parameters in the first layer of the neural network. By prioritizing updates for more influential features, the SHAP-informed gradient adjustment introduces a fine-grained, feature-specific influence on optimization. This approach dynamically adjusts the gradients during backpropagation, enabling the model to focus more on the most important features while learning.

In recent years, interpretability and feature importance have become essential considerations in deep learning, especially as models grow in complexity and applicability. Feature importance provides insights into which input variables most significantly affect a model's predictions, offering a foundation for refining optimization strategies. While techniques like those proposed by Shrikumar et al. [8] and Ribeiro et al. [9] primarily focus on explaining model predictions post-training, their broader principles highlight the value of understanding feature influence. These insights, while not rendering the model itself intrinsically interpretable, can guide adjustments during training by shedding light on how features affect learning dynamics. This is particularly important in fields where decision-making transparency is valued, as it allows for better alignment of optimization strategies with domain-specific requirements.

Incorporating feature importance into learning rate or gradient adjustments, such as in SHAP-informed adjustments, leverages interpretability tools to guide the optimization process. By prioritizing more significant features during training, SHAP-informed learning rate adjustments aim to refine learning dynamics and potentially improve model accuracy. While this approach does not make the model intrinsically interpretable, it uses interpretable metrics like SHAP values to inform optimization strategies. This integration represents a novel intersection between interpretability tools and optimization techniques, demonstrating how signals from feature importance can support more informed learning rate adjustments.

The choice of learning rate adjustment strategy significantly impacts the convergence speed and final performance of neural network models, with different strategies offering unique advantages depending on the specific training scenario [10]. Past studies have shown that combining adaptive optimizers like Adam with advanced learning rate schedules can yield substantial improvements in training efficiency and model accuracy, underscoring the importance of continual innovation in optimization techniques [11]. Integrating interpretability tools such as SHAP into the optimization process not only has the potential to enhance model performance but also provides insights into the learning dynamics, enabling researchers to refine optimization strategies more effectively.

The primary aim of this research is to evaluate the efficacy of integrating global learning rate adjustments with the Adam and SGD optimizers. The specific objectives include:

1. Performance Assessment: Compare the performance of neural networks optimized with standalone Adam and SGD versus those using these optimizers with learning rate annealing, step decay, and SHAP-informed adjustments. Performance metrics include test loss, RMSE (Root Mean Square Error), R^2 (coefficient of determination), accuracy (for classification tasks), and training time.
2. Feature-importance Based Gradient Updates: Introduce a novel method where SHAP-informed feature importance is directly applied to scale the gradients of individual parameters in the first layer. This targeted adjustment replaces the global learning rate scaling used in earlier experiments, providing a more granular approach to learning rate adjustments.
3. Computational Efficiency: Analyze the trade-offs between performance improvements and computational efficiency, particularly focusing on the SHAP-informed method, which is computationally intensive due to the calculation of SHAP values.

Optimizing neural networks' learning rates is a critical aspect of enhancing their performance. While the Adam optimizer is well-regarded for its adaptive capabilities, the potential improvements through systematic global learning rate adjustments remain an area of active investigation. This study's findings contribute to the understanding of these combined methods, offering insights into best practices for neural network training and highlighting the practical benefits and limitations of each approach.

Combining global learning rate adjustments with the Adam optimizer is relatively conventional in contemporary deep learning practices. Researchers often employ various strategies to optimize neural networks, making the integration of these techniques a natural progression. However, the systematic comparison and evaluation presented in this study, especially involving SHAP values for targeted parameter adjustments, provide a novel and valuable addition to the existing body of knowledge.

1.2. Related Work

Research on optimization techniques for training neural networks has been extensive, with various methods developed to improve convergence rates and overall model performance. Among these, the Adam optimizer, proposed by Kingma and Ba [3], has become one of the most widely used due to its adaptive learning rate capabilities, which adjust the learning rate for each parameter individually based on estimates of the first and second moments of the gradients.

Several studies have explored enhancements to the Adam optimizer to further improve its efficiency and convergence properties. For instance, Kabiri et al. introduced AMAdam (2024) [12], an adaptive modifier of the Adam method that addresses challenges in gradient-based optimization. Their approach demonstrated promising improvements in both training speed and final model accuracy across various tasks. Similarly, Huang et al. presented the Nostalgic Adam optimizer in 2019 [13], which emphasizes the importance of past gradients more heavily when designing the adaptive learning rate. This method demonstrated superior performance in terms of convergence speed and robustness compared to the original Adam.

Beyond Adam, SGD (Stochastic Gradient Descent) remains a widely used foundational optimization algorithm in neural network training. Despite its simplicity, SGD often benefits from enhancements like momentum or learning rate scheduling. The impact of momentum and learning rate scheduling on SGD has been empirically evaluated, showing significant improvements in convergence speed and generalization performance. For instance, the use of momentum in SGD helps accelerate convergence by navigating along the relevant directions and dampening oscillations, leading to more efficient training processes [14].

In addition to optimizer-specific enhancements, various learning rate scheduling techniques have been developed to further improve training dynamics. These include exponential annealing, step decay, and adaptive schedules. For example, studies on step decay have shown its effectiveness in stabilizing the training process during later epochs, leading to more precise convergence [6]. Similarly, learning rate annealing has been widely adopted for its ability to gradually refine model parameters, especially in the final stages of training [5]. These strategies have been integrated with both Adam and SGD to evaluate their collective impact on neural network performance.

The use of SHAP-informed adjustments for learning rate scaling is a relatively novel approach that intersects the domains of interpretability and optimization. SHAP values, based on cooperative game theory, provide a consistent and unified measure of feature importance, which has been primarily used for explaining model predictions [7]. Earlier works by Shrikumar et al. [8] and Ribeiro et al. [9] emphasized the importance of interpretability in understanding neural network behavior. While these studies primarily focused on post-hoc explanations for model predictions, our work leverages SHAP values during the training process to inform optimization dynamics.

In particular, our recent adaptation of SHAP-informed adjustments represents a significant innovation by targeting gradients of individual parameters rather than just uniformly scaling a global learning rate. This approach aligns with broader trends in machine learning to incorporate feature importance into model optimization. By dynamically scaling parameter updates based on feature importance, our method facilitates targeted learning and potentially enhances model convergence and performance.

Research integrating feature importance with optimization techniques remains limited. Notably, Luo et al. [15] investigated adaptive gradient methods with dynamic learning rate bounds, which demonstrated improved stability and efficiency. Their approach parallels our use of SHAP-informed adjustments, as both methods emphasize dynamic scaling to improve training dynamics. Similarly, Droguett et al. [16] proposed a feature selection framework within deep neural networks that identifies and prioritizes critical input features during training, enhancing both interpretability and performance. By integrating feature importance directly into the learning process, their method exemplifies how optimization strategies can benefit from feature-driven adjustments, aligning with the principles behind SHAP-informed modifications.

Our study extends these ideas by leveraging SHAP values not just as a tool for interpretability but as a mechanism for real-time adjustments during training. By integrating SHAP-informed methods into both Adam and SGD optimizers, our experiments provide new insights into how feature importance metrics can guide and enhance optimization strategies.

This work is situated within the broader context of neural network optimization research. While adaptive optimizers like Adam have seen extensive study and modifications, our focus on integrating global and targeted learning rate adjustments with both Adam and SGD represents a unique contribution. In particular:

1. The evaluation of SHAP-informed targeted gradient updates and learning rate scaling provides a new avenue for incorporating feature importance directly into optimization, enhancing performance.
2. The systematic comparison of learning rate scheduling methods—step decay, annealing, and SHAP-informed adjustments—within both Adam and SGD expands the scope of optimization research. This work highlights their combined effects, addressing gaps in the literature regarding how feature importance-driven methods compare to conventional learning rate strategies.
3. By using datasets of varying sizes and types (Breast Cancer, Diabetes, and California Housing), this study ensures that findings are robust and applicable to a range of real-world scenarios.

2. Materials and Methods

2.1. Theory and Explanation

In this section, we use bold symbols (e.g., $\boldsymbol{g_t}, \boldsymbol{m_t}, \boldsymbol{\theta_t}$) to denote vector quantities, while non-bold symbols (e.g., $\boldsymbol{\phi_i}, \boldsymbol{\alpha}, \boldsymbol{\beta_1}, \epsilon$) represent scalars. All operations involving vectors (e.g., addition, multiplication) are applied element-wise unless otherwise specified.

The Adam optimizer is an adaptive learning rate optimization algorithm designed to handle sparse gradients on noisy problems. Adam combines the advantages of two other popular optimization methods: AdaGrad and RMSProp. The main equations for Adam are:

$$\boldsymbol{m}_t = \beta_1 \boldsymbol{m_{t-1}} + (1 - \beta_1)\boldsymbol{g_t}$$

$$\boldsymbol{v}_t = \beta_2 \boldsymbol{v_{t-1}} + (1 - \beta_2)\boldsymbol{g}_t^2$$

$$\hat{\boldsymbol{m}}_t = \frac{\boldsymbol{m}_t}{1 - \beta_1^t}$$

$$\hat{\boldsymbol{v}}_t = \frac{\boldsymbol{v}_t}{1 - \beta_2^t}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon}$$

where $\boldsymbol{\theta}$ represents the parameters (weights) of the neural network, $\boldsymbol{g}_t$ is the gradient at time step t, $\boldsymbol{m}_t$ and $\boldsymbol{v}_t$ are the first and second moment estimates, $\hat{\boldsymbol{m}}_t$ and $\hat{\boldsymbol{v}}_t$ are the bias-corrected moment estimates, α is the learning rate, β_1 and β_2 are the decay rates for the moment estimates, and ϵ is a small constant to prevent division by zero.

Stochastic Gradient Descent (SGD), on the other hand, is one of the simplest yet widely used optimization algorithms for training neural networks. Unlike Adam, which adaptively scales the learning rate for each parameter, SGD updates the parameters by moving in the direction of the negative gradient of the loss function. The equation for SGD is:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \boldsymbol{g}_t$$

where $\boldsymbol{\theta}$ is the parameter vector, $\boldsymbol{g}_t$ is the gradient of the loss with respect to $\boldsymbol{\theta}$ at time step t, and α is the global learning rate. SGD does not include moment estimates or adaptive scaling of the learning rate, making it computationally simpler but potentially slower to converge without further enhancements like momentum or learning rate schedules.

2.1.1. SHAP Values

SHAP (SHapley Additive exPlanations) values provide a unified measure of feature importance based on cooperative game theory. SHAP values are computed for each feature x_i in the input data. These values represent the contribution of each feature to the prediction.

For a given model f and an instance x, the SHAP value ϕ_i for feature i is defined as:

$$\phi_i = \sum_{S \subseteq N \setminus i} \frac{|S|!(|N| - |S| - 1)!}{|N|!} [f(S \cup i) - f(S)]$$

where N is the set of all features, S is a subset of features, $f(S)$ is the model prediction for the subset S, and $f(S \cup i)$ is the model prediction including feature i.

SHAP values are derived from cooperative game theory and ensure that the contribution of each feature to a model's prediction is consistently computed across instances. This consistency arises from SHAP's adherence to properties such as additivity and symmetry, which provide a theoretical basis for attributing importance in a model-agnostic manner [17]. While parameter weights inherently represent a model's learned relationships, they often reflect only the internal state of the network and lack a direct, interpretable mapping to input feature significance. SHAP values, in contrast, offer insights into the external importance of features as observed through their marginal contributions to the output.

In optimization, SHAP values complement parameter weights by guiding adjustments with an additional layer of interpretability tied to the dataset's feature space. Although not a substitute for weight magnitudes, SHAP values provide actionable information when normalized, enabling strategies like dynamic learning rate or gradient adjustments to focus more effectively on impactful features. This approach aligns with established practices in adaptive optimizers like RMSProp and AdaGrad, which prioritize parameter updates based on gradients but do not inherently incorporate feature-specific information from the data.

2.1.2. Integrating SHAP with Learning Rate Adjustment

In our first approach, we integrate SHAP values with the learning rate adjustment mechanism. SHAP values are used to periodically adjust the learning rate for the neural network parameters during training. This is achieved through the following steps:

1. Compute SHAP Values: At specified intervals during training, compute the SHAP values for the features in the training data.
2. Calculate Feature Importances: Determine the average absolute SHAP value for each feature, representing its importance.

$$\text{mean\_importance}_{\boldsymbol{i}} = \frac{1}{\boldsymbol{m}} \sum_{j=1}^{\boldsymbol{m}} \left| \boldsymbol{\phi}_{\boldsymbol{i}}^{(j)} \right|$$

where m is the number of samples in the training data, and $\phi_i^{(j)}$ is the SHAP value for feature i in sample j.

3. Normalize Feature Importances: Normalize the feature importances to obtain the normalized importance values.

$$\text{normalized\_importance}_i = \frac{\text{mean\_importance}_i}{\mathbf{max}(\text{mean\_importance})}$$

4. Adjust Learning Rates: Use the normalized importance values to adjust the learning rates for the neural network parameters. The learning rate for each parameter is scaled by the sum of the normalized importances of all features. This integration ensures that the learning rate is generally impacted by features by a single figure derived from SHAP values which encode general meaning from the features.

$$\alpha' = \alpha \cdot \sum_{i=1}^{n} \text{normalized\_importance}_i$$

where α is the base learning rate, α' is the adjusted learning rate, and n is the total number of features.

Integrating the SHAP-informed learning rate adjustment into the Adam optimizer involves modifying the learning rate in the Adam and SGD update rules. The adjusted learning rate $\boldsymbol{\alpha}'$ is applied to the Adam and SGD formulas as follows:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \boldsymbol{\alpha}\prime \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \boldsymbol{\epsilon}}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \boldsymbol{\alpha}\prime \boldsymbol{g}_t$$

Unlike adaptive optimizers like AdaGrad, where learning rate adjustments diminish continuously based on accumulated gradients, the periodic recalculation and application of the aggregated adjustment in this method prevent such diminishing effects. By introducing recalculated adjustments at fixed intervals during training, this approach retains its influence throughout later epochs, enabling the optimizer to maintain progress even as the loss landscape becomes more complex. The recalculated adjustment acts as a consistent stabilizing factor, smoothing extreme parameter updates and moderating the risk of overshooting during optimization. Additionally, the periodic recalibration serves as a form of external intervention, introducing renewed influences on the optimization process that may help navigate saddle points or shallow local minima in the optimization landscape.

This method complements Adam's parameter-specific adjustments by applying a global multiplier derived from external calculations. While Adam inherently adapts learning rates at the parameter level, this recalculated multiplier provides a fresh influence on the global optimization dynamics. By applying a uniform adjustment across all parameters, the method smooths fluctuations caused by individual parameter updates, thereby stabilizing the optimization process. This stability is particularly beneficial in high-dimensional optimization problems, where the interaction of numerous parameters and gradients can exacerbate noise and overfitting. The scheduled recalculated adjustment helps mitigate these effects, aligning the optimization trajectory with broader trends in the dataset without overemphasizing individual parameter influences.

The rationale for applying a recalculated multiplier lies in its ability to act as a bridge between theoretical feature relevance and practical optimization dynamics. While the aggregated adjustment itself is derived from a summation of feature contributions, its application to the learning rate does not require direct interpretability of the value. Instead, the value's incremental application reflects a form of external regularization that promotes smoother transitions in parameter updates, particularly in non-convex optimization spaces. Comparable mechanisms have been observed in other adaptive strategies, such as AdaGrad, which rely on accumulated gradients to scale learning rates. Similarly, this adjustment provides a dataset-wide influence that balances stability and adaptivity, explaining the observed improvements in generalization and convergence.

2.1.3. Integrating SHAP with Gradient Adjustment

In our second approach, the normalized SHAP values are integrated directly into the gradient adjustment process to prioritize parameter updates based on feature importance. This method ensures that gradients associated with more influential features are scaled appropriately, enhancing the learning dynamics of both Adam and SGD optimizers.

The integration of SHAP values into gradient scaling is rooted in the idea that the first layer of a neural network directly interacts with the input features, making its gradients the most interpretable and impactful for feature-specific adjustments. Gradients determine the magnitude of parameter updates and scaling them based on SHAP-derived feature importance ensures that these updates are guided by data-specific relevance. By focusing adjustments on the first layer, this approach leverages the fact that the initial transformation of input features has a cascading influence on downstream layers.

While the gradient updates are minor in magnitude, their cumulative effect across training epochs contributes to aligning the network's optimization dynamics with the dataset's most important features. This method introduces a fine-grained focus, amplifying updates for critical features and attenuating noise from less influential ones, thereby promoting generalization and reducing overfitting.

Limiting SHAP-based gradient adjustments to the first layer balances computational efficiency with optimization effectiveness. The first layer's gradients are most closely tied to the input feature dimensions, allowing the SHAP values to act as precise modifiers. This selective approach avoids unnecessary complexity in deeper layers, where gradients represent abstract transformations rather than direct feature relationships. Additionally, applying SHAP adjustments at the first layer introduces a stabilizing effect that propagates through the network, ensuring that the most important features influence optimization without disrupting the broader training dynamics.

The integration process involves the following steps:

1. Compute Gradients:
   a. At each training step, compute the gradient of the loss function $\boldsymbol{g_t}$ with respect to the model parameters $\boldsymbol{\theta_t}$.
   b. The gradient vector $\boldsymbol{g_t}$ is calculated as:

$$\boldsymbol{g_t} = \nabla \boldsymbol{\theta_t} \boldsymbol{\mathcal{L}}(\boldsymbol{\theta_t})$$

   where $\boldsymbol{\mathcal{L}}(\boldsymbol{\theta_t})$ is the loss function.

2. Scale Gradients Using SHAP Importances
   a. Normalize the SHAP importances for each feature i as before.

$$\text{normalized\_importance}_i = \frac{\text{mean\_importance}_i}{\mathbf{max}(\text{mean\_importance})}$$

   b. Update the parameter gradient $\boldsymbol{g_{ti}}$ to a new scaled gradient $\boldsymbol{g'_{ti}}$ by applying each normalized_importance$_i$, where i refers to the corresponding feature.

$$\boldsymbol{g'_{ti}} = \boldsymbol{g_{ti}} \cdot \text{normalized\_importance}_i$$

In this context, each feature i in the input data corresponds to a parameter in the first layer of the neural network. Gradients g'_{ti} are computed for these parameters during backpropagation.

3. Gradient Update with Adam Optimizer
    a. In the Adam optimizer, the scaled gradient g'_t is used to update the first and second moment estimates:

$$m'_t = \beta_1 m_{t-1} + (1-\beta_1) g'_t$$

$$\mathrm{v}'_\mathrm{t} = \beta_2 v_{t-1} + (1-\beta_2) {g'_t}^2$$

Here, m'_t is the updated first moment estimate, and v'_t is the updated second moment estimate.

    b. The bias-corrected estimates are then calculated as:

$$\hat{m}'_t = \frac{m'_t}{1-\beta_1^t}$$

$$\hat{v}'_t = \frac{\mathrm{v}'_\mathrm{t}}{1-\beta_2^t}$$

    c. Finally, the parameters are updated as:

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}'_t}{\sqrt{\hat{v}'_t} + \epsilon}$$

4. Gradient Update with SGD Optimizer
    a. In the SGD optimizer, the scaled gradient g'_t is used directly for parameter updates:

$$\theta_{t+1} = \theta_t - \alpha g'_t$$

This integration of SHAP values into the gradient adjustment process leverages feature importance to prioritize updates for more significant parameters. The approach ensures that both Adam and SGD optimizers dynamically adjust their updates, reflecting the importance of individual features.

Like the learning rate adjustment method, these gradient adjustments are calculated at fixed intervals, taking a different direction from AdaGrad. Rather than diminishing adjustments over time, this method recalculates SHAP values periodically to maintain adaptability to the dataset's evolving loss landscape. This interval-based recalibration equips the optimizer to better navigate complex optimization dynamics, handle high-dimensional spaces, and address challenges such as overfitting and local minima.

By scaling the gradient vector g_t using SHAP-derived feature importance values, this method introduces a novel mechanism that combines externally calculated feature importance with optimization. The SHAP-informed gradient adjustment offers potential improvements in model accuracy, especially in scenarios where externally derived feature importance provides additional guidance for training beyond what is captured by the model's parameters.

2.2. Methodology

Our study focuses on evaluating the effectiveness of various learning rate adjustment methods integrated with the Adam and SGD optimizers in neural network training. Specifically, we assess the performance of each base optimizer alone, each base optimizer with step decay, with annealing, and with SHAP-informed adjustments on three datasets: Breast

Cancer, Diabetes, and California Housing. The experimental setup is designed to ensure comprehensive evaluation through grid search optimization of key hyperparameters.

2.2.1. Data Preprocessing

We utilize three widely recognized datasets for this study:

1. Breast Cancer Dataset: This dataset contains 569 instances of breast cancer diagnoses, with 30 features representing various characteristics of cell nuclei present in the digitized image of a fine needle aspirate (FNA) of a breast mass. The target variable is binary, indicating whether the cancer is malignant or benign.
2. Diabetes Dataset: This dataset consists of 442 instances with 10 features representing physiological measurements and a target variable indicating a quantitative measure of disease progression one year after baseline.
3. California Housing Dataset: This dataset contains 20,640 instances collected from the 1990 California census, with 8 features representing socio-economic and geographical characteristics such as median income, average number of rooms, and population per household in each block group. The target variable is a continuous value indicating the median house value for each block group, expressed in hundreds of thousands of dollars.

For each of datasets, we perform standard preprocessing steps. First, we normalize the feature values to ensure that all features are on a comparable scale. After normalization, the data is randomly shuffled and split into three subsets: 70% for training, 15% for validation, and 15% for testing.

2.2.2. Model Architecture

We employ a simple feedforward neural network for each of the datasets. The network architecture consists of three fully connected layers with ReLU activation functions and dropout for regularization. The architecture is as follows:

1. Input Layer: The number of nodes equals to the number of features in the dataset.
2. Hidden Layer 1: 128 nodes with ReLU activation and dropout.
3. Hidden Layer 2: 64 nodes with ReLU activation and dropout.
4. Output Layer: A single node for regression tasks (Diabetes dataset and California Housing) or a single node with sigmoid activation for binary classification tasks (Breast Cancer dataset).

2.2.3. Optimization Methods

We investigate the following optimization methods:

1. **Adam**: The base optimizer used in all experiments. Adam combines the advantages of two other extensions of stochastic gradient descent: AdaGrad and RMSProp. It computes adaptive learning rates for each parameter [3].
2. **SGD (Stochastic Gradient Descent):** A simple and widely used optimization algorithm that updates model parameters by moving in the direction of the negative gradient of the loss function. Despite its simplicity, it serves as a robust baseline and can be enhanced with techniques like momentum and learning rate scheduling.
3. **Step Decay**: The learning rate is decreased by a factor after a set number of epochs. This method helps the model converge by making larger steps in the initial phase and smaller steps as it approaches the minima.
4. **Annealing**: A technique where the learning rate is exponentially decayed over epochs. This gradual reduction helps in fine-tuning the weights during the later stages of training [11].

5. **SHAP-informed Learning Rate Adjustments**: SHAP values, which provide a measure of feature importance, are used to guide global learning rate adjustments. In this method, the aggregated feature importance across each of the features is calculated at specified intervals during training, and the learning rate is scaled uniformly across all parameters based on this importance.
6. **SHAP-Informed Gradient Adjustments**: Here, SHAP values for each feature are directly integrated into the corresponding gradient calculations at the parameter level. This method adjusts the gradients during backpropagation by scaling each parameter's gradient proportionally to its associated feature importance. By prioritizing updates for more influential features, this method introduces feature-specific adjustments to the optimization process.

2.2.4. Experimental Setup

In this experiment, we utilize a well-defined procedure for training and evaluating each model across various hyperparameter configurations to ensure robust and accurate results. The experiment is performed in two stages: a grid search over hyperparameters and the selection of the best model based on validation set performance, followed by a final evaluation on the test set.

Upon constructing the neural network model, the weights (often referred to as kernels) of the layers are initialized using PyTorch's (version 2.1.1, Meta, Menlo Park, CA, USA) default initialization method. Specifically, for fully connected layers ('torch.nn.Linear'), the weights are initialized with a small random distribution following the Xavier initialization method, which is designed to maintain the variance of activations across layers. Biases are initialized to zero. This initialization is crucial as it sets the starting point for the optimization process, influencing the convergence behavior of the model during training.

The dataset is shuffled and split into three subsets: 70% for training, 15% for validation, and 15% for testing. The training set is used to train the model and adjust its weights, while the validation set is used to evaluate the model during training and select the best hyperparameters. The test set is used for final evaluation to assess the model's generalization performance. This process is repeated across multiple trials to ensure consistent results and robust model performance evaluation.

To address the computational demands of SHAP calculations on the larger California dataset, we implemented a subsampling strategy. At specified intervals during training, a random subset of 500 samples from the training data was selected for SHAP value computation. This subset size was determined empirically to balance computational efficiency and maintain the representation of feature importance distributions. The calculated SHAP values from this subset were then used to guide gradient and learning rate adjustments.

Each optimization method (e.g., Adam, SHAP-informed, etc.) is evaluated using a grid search across the following hyperparameters:

- **Learning Rate**: [0.01, 0.001, 0.0001]
    - SGD + certain SHAP methods required [0.00001] and gradient clipping.
- **Dropout Rate**: [0.1, 0.3, 0.7]
- **Epochs**: [25, 50, 100]
- **Additional Parameters**: These vary depending on the method, such as decay rates for annealing and step decay, and SHAP update intervals labeled SHAP10, SHAP20, SHAP40 (every 10, 20, 40 epochs for diabetes dataset, and every 20, 40, 60 epochs for breast cancer dataset).

During the training phase, for each combination of hyperparameters, models are trained on the training set, with the model's weights being updated over multiple epochs using the training data. After each epoch, the model is evaluated on the validation set. The validation phase is crucial for hyperparameter tuning, as it allows us to assess the model's performance on unseen data and helps in identifying the best-performing configuration. The model with the lowest validation loss across all hyperparameter combinations is selected as the best model.

Once the best model is selected based on its performance on the validation set, it is evaluated on the test set using the following evaluation metrics:

- **Test Loss**: The average of the model's prediction errors on the test data. It indicates how well the model generalizes to new, unseen data.
- **Root Mean Square Error (RMSE)**: A standard way to measure the error of a regression model (used in the diabetes dataset). RMSE is the square root of the average of squared differences between actual and predicted values.
- **Accuracy**: Used in classification tasks (for the breast cancer dataset), accuracy measures the percentage of correct predictions made by the model out of all predictions.
- **R^2 Score**: A statistical measure of how well the model explains the variance in the test data. It provides insight into the goodness of fit of the model.
- **Training Time**: The total time taken to train the model, including all epochs.

This evaluation process is repeated across 10 trials, where the dataset is shuffled and split anew for each trial. This ensures that the results are robust and not dependent on a specific data split. The results from the test set are averaged across these trials, and the final performance metrics are reported as the mean ± standard deviation.

3. Results

3.1. Breast Cancer Dataset

3.1.1. Breast Cancer Dataset: Adam Optimizer Base

The results from our extensive experiments on the Breast Cancer dataset reveal important insights into the efficacy of different learning rate adjustment methods when used in conjunction with the Adam optimizer. This section presents the performance metrics, including test loss, accuracy, and training time, for each method. The summary of results is presented in Table 1.

Table 1. Summary of average performance metrics across various optimization methods on the breast cancer dataset. Each method's results include average test loss, average accuracy, and average training time along with their respective standard deviations. The table provides a direct comparison of methods, with SHAP-informed learning rate adjustments (SHAP20, SHAP40, SHAP60) and SHAP-informed gradient adjustments (SHAP10 g, SHAP20 g) showing some improvements in accuracy but generally requiring longer training times.

Method	Test Loss	Accuracy	Training Time
Adam only	0.0063 ± 0.0018	0.9233 ± 0.0295	*1.8535 ± 0.2920*
Adam + Annealing	0.0063 ± 0.0022	0.9337 ± 0.0202	**1.6946 ± 0.4909**
Adam + SHAP20	*0.0054 ± 0.0021*	**0.9407 ± 0.0204**	40.8644 ± 9.4052
Adam + SHAP40	0.0068 ± 0.0028	0.9186 ± 0.0227	25.9194 ± 5.6558
Adam + SHAP60	0.0077 ± 0.0056	0.9209 ± 0.0441	18.0186 ± 2.8206
Adam + Step Decay	**0.0049 ± 0.0016**	*0.9407 ± 0.0267*	1.9208 ± 0.0355
Adam + SHAP10 g	0.0057 ± 0.0017	0.9221 ± 0.0351	112.6715 ± 5.0572
Adam + SHAP20 g	0.0069 ± 0.0024	0.9209 ± 0.0269	60.1762 ± 12.1188

Bold values indicate the best-performing method for each metric, while italic values indicate the second-best performing method for each metric.

The breast cancer dataset results provide a detailed comparison of the tested optimization methods, highlighting both the strengths and limitations of each approach.

Key Observations:

- **Test Loss**: The SHAP20 method achieved a test loss of 0.0054, performing better than SHAP40, SHAP60, and Annealing. However, Step Decay yielded the lowest test loss (0.0049), outperforming all other methods in this regard.
- **Accuracy**: SHAP20 and Step Decay both achieved the highest average accuracy (0.9407), marginally outperforming Adam (0.9233) and Annealing (0.9337). SHAP40 and SHAP60 had slightly lower accuracy values at 0.9186 and 0.9209, respectively.
- **Training Time**: The SHAP-informed methods incurred a significant increase in training time due to the computational overhead of calculating SHAP values. SHAP20 took an average of 40.8644 s, followed by SHAP40 at 25.9194 s and SHAP60 at 18.0186 s. In contrast, Adam, Annealing, and Step Decay demonstrated much faster training times, with Step Decay showing the fastest time (1.9208 s).
- **Standard Deviations**: The standard deviations for both accuracy and training time reveal the consistency of each method's performance. The SHAP methods exhibit relatively higher variability in training time, particularly SHAP20 with a standard deviation of 9.4052 s, compared to the lower variability seen in non-SHAP methods like Step Decay (0.0355).

The results highlight the competitive performance of SHAP-informed methods, particularly Adam + SHAP20, which achieved the lowest test loss and matched the highest accuracy observed. However, this improvement came at the cost of significantly higher training times compared to Step Decay, which maintained a strong balance of efficiency and performance. These findings underscore the trade-off between leveraging SHAP-based feature relevance and computational overhead, suggesting that SHAP methods may offer value in scenarios where accuracy takes precedence over training efficiency.

3.1.2. Breast Cancer Dataset: SGD Optimizer Base

The results from our experiments using the SGD optimizer as the base for training on the Breast Cancer dataset provide further insights into the impact of various learning rate adjustment strategies. These results are presented in Table 2 and include test loss, accuracy, and training time, alongside their respective standard deviations.

Key Observations:

- **Test Loss**: The SGD + SHAP20_gradient method achieved the lowest test loss (0.00824 ± 0.00114), marginally outperforming other SHAP-informed methods and SGD alone (0.00972 ± 0.00167). However, it was comparable to SGD + Step_Decay (0.00935 ± 0.00110), which also demonstrated strong performance.
- **Accuracy**: SGD + SHAP20_gradient also exhibited the highest accuracy (0.922 ± 0.027), demonstrating the effectiveness of SHAP-informed targeted parameter updates in improving classification performance. This method outperformed SGD only (0.913 ± 0.026) and SGD + Step_Decay (0.911 ± 0.023). Conversely, SGD + SHAP10_gradient showed reduced accuracy (0.886 ± 0.011), indicating that the SHAP update interval plays a significant role in determining the efficacy of the gradient-based adjustments.
- **Training Time**: Similar to the Adam experiments, SHAP-informed methods with SGD resulted in increased training time compared to non-SHAP methods. SGD + SHAP20_gradient required the highest training time (58.07 ± 7.28 s), followed by SGD + SHAP10_gradient (114.20 ± 17.95 s). In contrast, SGD + Step_Decay and SGD only required significantly less training time (1.25 ± 0.49 s and 0.86 ± 0.50 s, respectively).

Table 2. Summary of average performance metrics across various optimization methods on the breast cancer dataset. Each method's results include average test loss, average accuracy, and average training time along with their respective standard deviations. The table provides a direct comparison of methods, with SHAP-informed learning rate adjustments (SHAP20, SHAP40, SHAP60) and SHAP-informed gradient adjustments (SHAP10 g, SHAP20 g).

Method	Test Loss	Accuracy	Training Time
SGD only	0.0097 ± 0.0017	0.9128 ± 0.0266	**0.8608 ± 0.5045**
SGD + Annealing	0.0152 ± 0.0035	0.8558 ± 0.0251	*1.03728 ± 0.4938*
SGD + SHAP20	0.0104 ± 0.0021	0.9070 ± 0.0379	43.3277 ± 11.1142
SGD + SHAP40	0.0098 ± 0.0013	*0.91406 ± 0.0255*	21.5104 ± 8.5463
SGD + SHAP60	0.0105 ± 0.0023	0.9023 ± 0.0290	16.3279 ± 5.8155
SGD + Step_Decay	*0.0094 ± 0.0011*	0.9116 ± 0.0239	1.2524 ± 0.4924
SGD + SHAP10 g	0.0118 ± 0.0029	0.8860 ± 0.0114	114.2087 ± 17.9569
SGD + SHAP20 g	**0.0082 ± 0.0011**	**0.9221 ± 0.0275**	58.8706 ± 7.2843

Bold values indicate the best-performing method for each metric, while italic values indicate the second-best performing method for each metric.

3.1.3. Breast Cancer Dataset: Additional Low Epoch Results

To supplement the main findings, additional experiments were conducted to evaluate the performance of SHAP20, Annealing, Step Decay, and Adam-only methods at lower epoch counts (5, 10, 20, 40, and 50 epochs). These tests aimed to assess the impact of reduced training cycles on model performance and convergence behavior for the Breast Cancer dataset. The key results are summarized in Table 3.

Table 3. Performance metrics for optimization methods on the Breast Cancer dataset at lower epoch settings (5, 10, 20, 40, 50 epochs). The table reports average test loss, standard deviation of test loss, average RMSE, standard deviation of RMSE, average R^2, standard deviation of R^2, average training time, and standard deviation of training time.

Method	Test Loss	Accuracy	Training Time
Adam	**0.0063 ± 0.0017**	0.9267 ± 0.0280	0.9215 ± 0.0919
Adam + Annealing	*0.0065 ± 0.0014*	**0.9291 ± 0.0257**	*0.9175 ± 0.0745*
Adam + SHAP20	0.0067 ± 0.0022	0.9244 ± 0.0271	23.2167 ± 4.3548
Adam + Step Decay	0.0070 ± 0.0024	*0.9279 ± 0.0328*	**0.9140 ± 0.0881**

Bold values indicate the best-performing method for each metric, while italic values indicate the second-best performing method for each metric.

The low-epoch results for the Breast Cancer dataset reveal that while SHAP20 demonstrated competitive performance, it did not outperform simpler methods such as Annealing or Step Decay in terms of accuracy or test loss. Annealing showed strong consistency with an average accuracy of 92.91%, while Step Decay delivered slightly higher accuracy at 92.79%, despite marginally increased test loss.

3.2. Diabetes Dataset

3.2.1. Diabetes Dataset: Adam Base Optimizers

The results for the Diabetes dataset provide insights into the performance of SHAP-informed learning rate adjustments compared to more traditional methods. The metrics for each method, including test loss, RMSE, R^2, and training time, are summarized in Table 4.

Key Observations:

- **Test Loss:** The Step Decay method achieved the lowest test loss (134.445), marginally outperforming SHAP10, SHAP20, and SHAP40, as well as Adam and Annealing. SHAP20 g, which was expected to perform better, achieved a slightly higher test loss at 135.464.

- **RMSE:** Step Decay also recorded the lowest RMSE (11.421), which indicates a better fit to the data compared to the SHAP methods and other optimizers. SHAP20 g again had the second best RMSE of 11.503.
- **R^2 Score:** While Step Decay again performed well with the second highest R^2 score (0.498), SHAP20 g and SHAP10 g showed slightly better results, achieving R^2 scores of 0.507 and 0.505, respectively. Adam had the lowest R^2 value at 0.425, demonstrating its comparatively lower explanatory power in this context.
- **Training Time:** Similar to the Breast Cancer dataset, SHAP-informed methods had substantially longer training times due to the computational complexity involved in SHAP value calculations. SHAP10 g, in particular, had the longest average training time at 56.922 s, compared to the non-SHAP methods.

Table 4. This table presents the average performance metrics from the grid search for each method on the Diabetes dataset. It includes values for average test loss, RMSE, R^2, and training time along with their respective standard deviations. Each row corresponds to a specific learning rate adjustment with a base optimizer of Adam.

Method	Test Loss	RMSE	R^2	Training Time
Adam only	135.506 ± 26.987	11.583 ± 1.158	0.425 ± 0.136	1.169 ± 0.406
Adam + Annealing	142.553 ± 25.729	11.890 ± 1.091	0.443 ± 0.061	**0.635 ± 0.302**
Adam + SHAP10	137.926 ± 27.510	11.684 ± 1.187	0.433 ± 0.069	51.718 ± 20.022
Adam + SHAP20	137.023 ± 32.803	11.620 ± 1.412	0.426 ± 0.086	27.240 ± 8.190
Adam + SHAP40	142.746 ± 28.640	11.881 ± 1.259	0.450 ± 0.070	18.892 ± 3.375
Adam + Step Decay	**134.445 ± 51.251**	**11.421 ± 2.003**	0.498 ± 0.095	*0.908 ± 0.483*
Adam + SHAP10 g	139.392 ± 46.801	11.6683 ± 1.801	*0.505 ± 0.052*	56.922 ± 28.972
Adam + SHAP20 g	*135.464 ± 40.692*	*11.503 ± 1.774*	**0.507 ± 0.077**	45.522 ± 8.776

Bold values indicate the best-performing method for each metric, while italic values indicate the second-best performing method for each metric.

The SHAP-informed methods did not substantially outperform traditional optimization methods like Step Decay in terms of test loss or RMSE on the Diabetes dataset. However, SHAP gradient methods maintained competitive performance in terms of on test loss and RMSE and even offered slightly more explanatory ability in terms of R^2 scores, suggesting that SHAP gradient methods could be more beneficial when feature importance plays a critical role in model interpretation and generalization. The trade-off between higher training times and modest improvements in model performance remains a factor to consider when scaling this approach to larger datasets or more complex architectures.

3.2.2. Diabetes Dataset: SGD Base Optimizers

The results from the Diabetes dataset provide insights into the performance of SHAP-informed learning rate adjustments compared to traditional methods when paired with the SGD optimizer. The metrics for each method, including test loss, RMSE, R^2, and training time, are summarized in Table 5.

Key Observations:

- **Test Loss**: Among the tested methods, SGD only achieved the lowest test loss (137.441), demonstrating the strength of a straightforward approach. SGD + Annealing followed as the second-best, with a test loss of 165.152, showcasing its effectiveness in certain scenarios. Notably, the SHAP20 g method showed promise with a test loss of 274.564, though it exhibited greater variability. This suggests that while SHAP-informed gradient adjustments can drive improvements, further optimization is needed to reduce inconsistency.

- **RMSE**: Similar to test loss, SGD only again performed the best with the lowest RMSE (11.637), indicating a better fit to the data. SGD + Annealing came second with an RMSE of 12.723. The inclusion of SHAP20_gradient and SHAP10_gradient methods demonstrates potential for improved model fitting, although their RMSE metrics (15.634 and 16.4510, respectively) highlight room for refinement in reducing variability.
- **R^2 Score**: SGD + Annealing achieved the highest R^2 score (0.442), reflecting its ability to explain more variance in the data compared to other methods. SGD only followed closely with an R^2 score of 0.378. While the SHAP methods, such as SHAP10 g and SHAP20 g, yielded negative R^2 values (−0.0099 and −0.0088, respectively), indicating that they explained less variance than a simple baseline, this could be attributed to the exhibited variability in the SHAP-informed gradient adjustments.
- **Training Time**: SGD + Annealing exhibited the fastest training time (0.8485 ± 0.5180), narrowly outperforming SGD only (0.8987 ± 0.2036) while SHAP associated method exhibited higher training times proportional to how often SHAP values were calculated.

Table 5. This table presents the average performance metrics from the grid search for each method on the Diabetes dataset. It includes values for average test loss, RMSE, R^2, and training time along with their respective standard deviations. Each row corresponds to a specific learning rate adjustment with a base optimizer of SGD.

Method	Test Loss	RMSE	R^2	Training Time
SGD only	**137.441 ± 34.428**	**11.637 ± 1.423**	*0.378 ± 0.135*	*0.899 ± 0.204*
SGD + Annealing	*165.152 ± 46.205*	*12.723 ± 1.810*	**0.442 ± 0.117**	**0.848 ± 0.518**
SGD + SHAP10	1287.276 ± 252.710	35.712 ± 3.454	−3.852 ± 0.435	79.712 ± 9.581
SGD + SHAP20	1382.738 ± 298.472	36.967 ± 4.015	−3.760 ± 0.424	44.374 ± 4.561
SGD + SHAP40	1311.710 ± 229.291	36.073 ± 3.232	−4.078 ± 0.763	25.727 ± 3.363
SGD + Step Decay	1277.564 ± 186.158	35.650 ± 2.578	−4.011 ± 0.365	1.452 ± 0.053
SGD + SHAP10 g	247.279 ± 53.595	15.634 ± 1.686	−0.010 ± 0.052	87.398 ± 13.780
SGD + SHAP20 g	274.564 ± 65.803	16.451 ± 1.982	−0.009 ± 0.0681	41.094 ± 9.304

Bold values indicate the best-performing method for each metric, while italic values indicate the second-best performing method for each metric.

3.2.3. Diabetes Dataset: Additional Low Epoch Results

Additional experiments with a lower range of epochs (5, 10, 20, 40, 50) were also performed for the SHAP20, Annealing, Step Decay, and Adam-only methods to assess their performance under different training constraints. The complete findings of these trials are presented below in Table 6.

Table 6. Performance metrics for optimization methods on the Diabetes dataset at lower epoch settings (5, 10, 20, 40, 50 epochs). The table reports average test loss, standard deviation of test loss, average RMSE, standard deviation of RMSE, average R^2, standard deviation of R^2, average training time, and standard deviation of training time.

Method	Test Loss	RMSE	R^2	Training Time
Adam	142.962 ± 38.743	11.846 ± 1.622	*0.484 ± 0.027*	*0.543 ± 0.185*
Adam+Annealing	148.125 ± 40.829	12.054 ± 1.678	0.424 ± 0.089	**0.327 ± 0.235**
Adam+SHAP20	**123.883 ± 33.401**	**11.035 ± 1.452**	**0.493 ± 0.081**	17.159 ± 4.353
Adam+Step Decay	*139.864 ± 50.955*	*11.648 ± 2.047*	0.472 ± 0.074	0.559 ± 0.109

Bold values indicate the best-performing method for each metric, while italic values indicate the second-best performing method for each metric.

On the Diabetes dataset, SHAP20 exhibited superior performance with the lowest average test loss (123.88) and RMSE (11.04). This suggests that SHAP-informed adjustments could provide meaningful improvements, especially under shorter training schedules. It is important to note that while the SHAP update interval is set to 20, there is an initial calculation of SHAP values used to impact the learning rate before training has progressed even when epochs are set anywhere from 5–20. This suggests that a single SHAP update could be effective in specific scenarios. Nevertheless, the increased computational time associated with SHAP20 (17.16 s) remains a notable trade-off compared to the more computationally efficient Adam and Step Decay methods.

These results reinforce the trade-offs discussed in the main analysis—highlighting the balance between accuracy, efficiency, and computational cost. Future work could explore strategies for optimizing SHAP-based methods to make them more practical for shorter training durations and large-scale applications.

3.3. California Housing Dataset

3.3.1. California Housing Dataset: Adam Base Optimizers

The results for the California Housing dataset with the Adam optimizer provide valuable insights into the performance of a more advanced base optimizer adjustment methods used with the methods mentioned before. Table 7 summarizes the results, including average test loss, RMSE, R^2, and training time, along with their respective standard deviations.

Key Observations:

- **Test Loss**: The Adam + SHAP40 method achieved the lowest test loss (0.0153), showcasing its effectiveness in improving performance through SHAP-informed learning rate adjustments. Following closely, Adam + SHAP20 demonstrated the second-best test loss (0.0154), highlighting the consistency of SHAP-based methods in reducing test error. These results emphasize the potential of SHAP-informed strategies to outperform traditional learning rate methods such as step decay and annealing in this dataset.
- **RMSE**: For RMSE, Adam + SHAP40 again emerged as the top performer with an RMSE of 0.1237, indicating its ability to minimize prediction error. The Adam + Step Decay method followed with an RMSE of 0.1242, demonstrating its robustness as a non-SHAP learning rate adjustment method. These findings underline the competitive nature of SHAP-informed adjustments while also showing that step decay remains a strong alternative.
- **R^2 Score**: In terms of R^2, Adam + SHAP40 achieved the highest score (0.6332), further affirming its superior predictive capability. Adam + SHAP20 achieved the second-best R^2 value of 0.6262, showing its consistency across multiple metrics. These results highlight the value of SHAP-informed adjustments in enhancing the model's ability to explain variance in the dataset.
- **Training Time**: When considering training time, Adam + Annealing required the least time (23.3203), followed by Adam Only (33.1513).

3.3.2. California Housing Dataset: SGD Base Optimizers

The results for the California Housing dataset with the SGD optimizer provide valuable insights into the performance of traditional learning rate adjustment methods and SHAP-informed gradient adjustments. Table 8 summarizes the results, including average test loss, RMSE, R^2, and training time, along with their respective standard deviations.

Key Observations:

- **Test Loss**: SGD + SHAP20 achieved the lowest test loss (0.0377), demonstrating its ability to effectively minimize prediction errors on this dataset. SGD + SHAP10 followed closely with a test loss of 0.0380, further showcasing the potential of SHAP-informed gradient adjustments in optimizing performance.

- **RMSE**: The lowest RMSE was also achieved by SGD + SHAP20 (0.1939), underscoring its effectiveness in reducing prediction error. SGD + Step_Decay was the second-best with an RMSE of 0.1946, followed closely by SGD + SHAP10, which achieved an RMSE of 0.1948. This consistency across SHAP-informed methods highlights their robustness in improving error metrics.
- **R^2 Score**: SGD + SHAP20 delivered the best R^2 score (0.0973), indicating a potential ability to explain the variance in the data. SGD + Step_Decay achieved the second-best R^2 score of 0.0863, demonstrating its effectiveness.
- **Training Time**: SGD only exhibited the fastest training time (29.4050 s), followed by SGD + Annealing with a training time of 25.0927 s. While SHAP-informed methods, such as SGD + SHAP20 and SGD + SHAP10, demonstrated strong performance in test loss and RMSE, they required significantly more computational time, with training times of 80.9005 s and 103.6750 s, respectively.

Table 7. This table presents the average performance metrics from the grid search for each method on the California Housing Dataset. It includes values for average test loss, RMSE, R^2, and training time along with their respective standard deviations. Each row corresponds to a specific learning rate adjustment with a base optimizer of Adam.

Method	Test Loss	RMSE	R^2	Training Time
Adam only	0.0160 ± 0.0008	0.1267 ± 0.0031	0.6165 ± 0.0212	*33.1513 ± 11.1284*
Adam + Annealing	0.0372 ± 0.0079	0.1918 ± 0.0223	0.0947 ± 0.1870	**23.3203 ± 13.4268**
Adam + SHAP10	0.0155 ± 0.0003	0.1243 ± 0.0013	0.6290 ± 0.0070	163.1697 ± 24.9437
Adam + SHAP20	*0.0154 ± 0.0005*	0.1244 ± 0.0022	*0.6262 ± 0.0142*	89.6517 ± 22.2274
Adam + SHAP40	**0.0153 ± 0.0006**	**0.1237 ± 0.0022**	**0.6332 ± 0.0142**	105.1233 ± 22.5448
Adam + Step Decay	0.0154 ± 0.0009	*0.1242 ± 0.0038*	0.6296 ± 0.0188	57.2143 ± 7.3544
Adam + SHAP10 g	0.0155 ± 0.0005	0.1244 ± 0.0021	0.6284 ± 0.0123	159.9094 ± 43.4865
Adam + SHAP20 g	0.0158 ± 0.0008	0.1258 ± 0.0031	0.6237 ± 0.0141	104.0662 ± 20.7780

Bold values indicate the best-performing method for each metric, while italic values indicate the second-best performing method for each metric.

Table 8. This table presents the average performance metrics from the grid search for each method on the California Housing dataset. It includes values for average test loss, RMSE, R^2, and training time along with their respective standard deviations. Each row corresponds to a specific learning rate adjustment method with a base optimizer of SGD.

Method	Test Loss	RMSE	R^2	Training Time
SGD only	0.0497 ± 0.0044	0.2228 ± 0.0010	−0.1878 ± 0.1029	*29.4050 ± 29.4050*
SGD + Annealing	0.0419 ± 0.0012	0.2046 ± 0.0029	−0.0049 ± 0.0119	**25.0927 ± 7.2815**
SGD + SHAP10	*0.0380 ± 0.0032*	0.1948 ± 0.0085	0.0775 ± 0.0791	103.6750 ± 38.5406
SGD + SHAP20	**0.0377 ± 0.0046**	**0.1939 ± 0.0121**	**0.0973 ± 0.0950**	80.9005 ± 30.5020
SGD + Step Decay	0.0380 ± 0.0046	*0.1946 ± 0.0123*	*0.0863 ± 0.1034*	35.6515 ± 15.0702
SGD + SHAP10 g	0.0687 ± 0.0060	0.2618 ± 0.0115	−0.6554 ± 0.1275	72.3938 ± 37.3120
SGD + SHAP20 g	0.0736 ± 0.0059	0.2712 ± 0.0109	−0.7410 ± 0.1190	57.1104 ± 27.2620

Bold values indicate the best-performing method for each metric, while italic values indicate the second-best performing method for each metric.

4. Discussion

This study systematically evaluated the impact of global learning rate or gradient adjustments on neural network training, focusing on SHAP-informed adjustments, learning rate annealing, and step decay methods, all applied in conjunction with the Adam and SGD optimizers. The experiments were conducted on three distinct datasets—Breast Cancer, Diabetes, and California Housing—to explore the efficacy and computational cost of each method.

The SHAP-informed learning rate adjustment demonstrated clear performance advantages in terms of accuracy and generalization in most cases, though it exhibited mixed results across dataset and base optimizer combinations. For the Breast Cancer dataset, overall, Adam + SHAP20 achieved the tied highest accuracy (94.07% $\pm$ 0.0204) with a lower deviation compared to Adam + Step Decay (94.07% $\pm$ 0.0267). Although Adam + Step Decay demonstrated the lowest test loss (0.0049 $\pm$ 0.0016), with Adam + SHAP20 closely following at 0.0054 $\pm$ 0.0022. SGD + SHAP20 g (92.21%, test loss: 0.0082) and SGD + SHAP40 (91.41%) ranked first and second, respectively for SGD tests on this dataset. The strong performance of targeted gradient method (SGD + SHAP20 g) paired with SGD aligns with findings by Potharlanka and Bhat (2024), who demonstrated that feature importance feedback mechanisms can enhance optimization processes in ensemble-based models by prioritizing influential features [18]. These results suggest that step decay's gradual learning rate reduction allowed the model to adapt effectively to the dataset's characteristics, while the SHAP-informed methods provided an advantage by integrating either aggregated or gradient level feature importance into the learning process. However, the computational overhead of SHAP methods was evident, as these required significantly longer training times than both Adam-only and step decay methods.

On the Diabetes dataset, overall, Adam + SHAP20 (low epoch) again emerged as the best-permforming model with test loss 123.88 and RMSE of 11.035. Beyond that, Adam + Step Decay followed as the next closest-performing method with a test loss of 134.45 and an RMSE of 11.42. Adam + SHAP20 g ranked third with a test loss of 135.46 and an RMSE of 11.50. In the SGD context for this dataset SGD-only (137.44) and SGD + Annealing (165.15). Interestingly, while the targeted SHAP gradient adjustment (SHAP20 g) for SGD reduced RMSE compared to other SHAP implementations (e.g., SHAP10 and SHAP40), it still did not surpass the performance of SGD-only and SGD + Annealing. This finding indicates that while SHAP gradient adjustments enhance model learning by focusing on feature importance, their effectiveness likely varies depending on the optimizer and dataset characteristics. For this dataset, Adam + Step Decay likely excelled due to its smooth adjustment of learning rates, which helped the model better converge on this moderately sized tabular dataset.

The California Housing dataset presented a similar trend. Overall, Adam + SHAP40 achieved the best test loss (0.0153) and RMSE (0.1237), followed by Adam + Step Decay (0.0154, RMSE: 0.1242). Focusing on the SGD context, SGD + SHAP20 ranked first (test loss: 0.0377) and SGD + SHAP10 s (test loss: 0.0380). Notably, SGD + SHAP20g performed the worst among the evaluated methods, which could be attributed to the subsampling of 500 records used for SHAP calculations to reduce training time on this larger dataset. While this subsampling successfully mitigated computational burden, it may have limited the method's ability to fully capture the feature importance distribution required for SHAP gradient updates, potentially compromising performance.

Across datasets, the SHAP-informed methods demonstrated consistent performance gains, with Adam + SHAP20 achieving the best results on the Breast Cancer and Diabetes datasets and Adam + SHAP40 performing best on the California Housing dataset. These results suggest that the aggregated importance multiplier applied to the global learning rate may have facilitated enhanced generalization by aligning optimization dynamics with dataset-informed scaling. While the aggregated SHAP value is not directly interpretable as a standalone measure, its use as a global multiplier introduces a form of uniform adjustment that is expected to stabilize learning by moderating extreme parameter updates. This mechanism likely aids in navigating high-dimensional spaces where individual feature-specific adjustments may introduce instability or overfitting. Comparable effects have been observed in approaches such as adaptive gradient methods, where scaling factors derived from global statistics have been shown to improve convergence and generalization [19]. This aggregated strategy appears to provide a balance between integrating dataset-wide feature relevance and maintaining scalability, which could explain its superior performance across diverse datasets in this study.

The interval of SHAP updates also played a noticeable role in model performance. SHAP20, which updates every 20 epochs, generally outperformed SHAP40 and SHAP60 in terms of accuracy and RMSE, indicating that more frequent updates provide better adaptability to unseen data. However, this comes at the cost of longer training times and is not always the case. For example, SHAP40 showed competitive results on the California dataset, suggesting that less frequent updates may be more suitable for smaller datasets or tasks with fewer feature variations. Further research is needed to establish optimal SHAP intervals for various dataset characteristics.

The varying effectiveness of SHAP-informed methods across datasets highlights the importance of dataset-specific characteristics. For example, learning rate adjusting SHAP methods were particularly beneficial on the California dataset, likely because the sampled size of 500 records (from an original 20,000) for SHAP calculations reduced computational costs while still preserving enough variance in feature importance for effective SHAP global learning rate adjustments. In contrast, on the Diabetes dataset, where the relationships between features and targets are more linear, simpler methods like step decay and SGD-only outperformed both types of SHAP-informed adjustments. These findings suggest that SHAP-informed methods may be more advantageous or useful for datasets with complex feature interactions or non-linear relationships [7].

One clear drawback of the SHAP-informed approach was the substantial increase in training time due to the computational overhead of calculating SHAP values at regular intervals. This increased cost is a critical factor for large-scale deployments and real-time applications [20]. While SHAP values offer valuable externally calculated measures of feature importance that inform more targeted gradient or learning rate adjustments, this resource-intensive process emphasizes the trade-off between leveraging advanced feature importance metrics and computational efficiency. The overhead may limit the applicability of SHAP-informed methods in environments requiring real-time or large-scale processing, emphasizing the need for optimization to make this approach more viable.

Beyond performance metrics alone, the SHAP-based adjustments hold considerable promise for applications in fields requiring transparency and informed decision-making, such as healthcare, finance, or legal contexts. By integrating externally derived feature importance into the training process, SHAP-informed adjustments can provide additional insights into the relative influence of features on the optimization process. While the computational cost of these methods remains a challenge, their ability to incorporate feature importance into optimization strategies may justify their use in critical, high-stakes environments where accurate and reliable models are essential [19].

Potential optimization strategies for the SHAP-informed approach could include reducing the frequency of SHAP calculations by adjusting the SHAP update interval, striking a balance between the anticipated performance gains and computational cost. Incremental calculation methods for SHAP values would be beneficial, allowing for reduced computational load with each update. Additionally, leveraging parallel processing techniques tailored to SHAP calculations—such as distributing feature importance computations across multiple processors or GPUs—could further decrease training time by optimizing resource utilization [21].

Incorporating SHAP derived feature importance into learning rate or gradient adjustment process demonstrates promise for enhancing model performance. Future research could further refine SHAP value scaling or related methods within other optimization frameworks, potentially uncovering even more effective training methodologies. Exploring the development of structured rules for implementing and testing these types of learning rate modifications represents a non-trivial, intriguing research direction [22]. Notably, future studies should prioritize optimizing the SHAP-informed approach to reduce computational overhead, alongside assessing its applicability across various neural network architectures, datasets and other base optimizers like RMSprop and AdaGrad. Investigating the combination of SHAP-informed adjustments with other advanced optimizers may also yield deeper insights into improving neural network training processes.

In conclusion, while the SHAP-informed approach incurs a higher computational cost, its performance benefits make it a compelling option for applications where accuracy and model generalization are paramount. This study contributes to the growing body of knowledge on learning rate optimization techniques, offering a novel perspective by integrating SHAP values into the optimization process.

Author Contributions: Conceptualization, J.G.; Formal analysis, J.G.; Investigation, J.G.; Methodology, V.S.S.; Project administration, V.S.S.; Software, J.G.; Writing—original draft, J.G.; Writing—review & editing, J.G. and V.S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets used in this study are publicly accessible through the following sources: **Breast Cancer Dataset**: Available from the Scikit-learn library (load_breast_cancer function in python). Documentation and details can be found at https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html (accessed on 19 October 2024). **Diabetes Dataset**: Available from the Scikit-learn library (load_diabetes function in python). Documentation and details can be found at https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html (accessed on 20 October 2024). **California Housing Dataset:** Available from the Scikit-learn library (fetch_california_housing function in Python). Documentation and details can be found at https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_california_housing.html (accessed on 8 January 2025). All three datasets are publicly available and widely used for benchmarking in machine learning research.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Liu, W.; Wang, Z.; Liu, X.; Zeng, N.; Liu, Y.; Alsaadi, F.E. A survey of deep neural network architectures and their applications. *Neurocomputing* **2017**, *234*, 11–26. [CrossRef]
2. Soydaner, D. A Comparison of Optimization Algorithms for Deep Learning. *Int. J. Pattern Recognit. Artif. Intell.* **2020**, *34*, 2052013. [CrossRef]
3. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations (ICLR), Poster, San Diego, CA, USA, 7–9 May 2015.
4. Ruder, S. An overview of gradient descent optimization algorithms. *arXiv* **2016**, arXiv:1609.04747.
5. Nakamura, K.; Derbel, B.; Won, K.-J.; Hong, B.-W. Learning-Rate Annealing Methods for Deep Neural Networks. *Electronics* **2021**, *10*, 2029. [CrossRef]
6. Ge, R.; Kakade, S.; Kidambi, R.; Netrapalli, P. The Step Decay Schedule: A Near Optimal, Geometrically Decaying Learning Rate Procedure For Least Squares. In Proceedings of the NeurIPS, Vancouver, BC, Canada, 8–14 December 2019.
7. Lundberg, S.M.; Lee, S.-I. A Unified Approach to Interpreting Model Predictions. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 4768–4777.
8. Shrikumar, A.; Greenside, P.; Kundaje, A. Learning Important Features Through Propagating Activation Differences. In Proceedings of the 34th International Conference on Machine Learning (ICML), Sydney, Australia, 6–11 August 2017; pp. 3145–3153.
9. Ribeiro, M.T.; Singh, S.; Guestrin, C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016.
10. Merrouch, M.; Atif, K.; Skittou, M.; Benyoussef, Y.; Gadi, T. AutoLrOpt: An Efficient Optimizer Using Automatic Setting of Learning Rate for Deep Neural Networks. *IEEE Access* **2024**, *12*, 3413043. [CrossRef]

11. Loshchilov, I.; Hutter, F. Decoupled Weight Decay Regularization. In Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.
12. Kabiri, H.; Ghanou, Y.; Khalifi, H.; Casalino, G. AMAdam: Adaptive modifier of Adam method. *Knowl. Inf. Syst.* **2024**, *66*, 3427–3458. [CrossRef]
13. Huang, H.; Wang, C.; Dong, B. Nostalgic Adam: Weighting More of the Past Gradients When Designing the Adaptive Learning Rate. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI), Macao, China, 10–16 August 2019; pp. 2556–2562. [CrossRef]
14. Saparudin, A.S.; Ali, M.A.M.; Taib, M.N. A Statistical Test of The Effect of Learning Rate and Momentum Coefficient of SGD and Its Interaction on Neural Network Performance. In Proceedings of the 2020 IEEE 10th Symposium on Computer Applications & Industrial Electronics (ISCAIE), Penang, Malaysia, 18–19 April 2020; pp. 102–107.
15. Luo, L.; Xiong, Y.; Liu, Y.; Sun, X. Adaptive Gradient Methods with Dynamic Bound of Learning Rate. In Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.
16. Figueroa Barraza, J.; López Droguett, E.; Martins, M.R. Towards Interpretable Deep Learning: A Feature Selection Framework for Prognostics and Health Management Using Deep Neural Networks. *Sensors* **2021**, *21*, 5888. [CrossRef] [PubMed]
17. Li, M.; Sun, H.; Huang, Y.; Chen, H. Shapley value: From cooperative game to explainable artificial intelligence. *Auton. Intell. Syst.* **2024**, *4*, 2. [CrossRef]
18. Potharlanka, J.L.; Bhat, N.M. Feature importance feedback with Deep Q process in ensemble-based metaheuristic feature selection algorithms. *Sci. Rep.* **2024**, *14*, 2923. [CrossRef] [PubMed]
19. Duchi, J.; Hazan, E.; Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **2011**, *12*, 2121–2159.
20. Akbar, A.; Kousiouris, G.; Pervaiz, H.; Sancho, J.; Ta-Shma, P.; Carrez, F.; Moessner, K. Real-Time Probabilistic Data Fusion for Large-Scale IoT Applications. *IEEE Access* **2018**, *6*, 10015–10027. [CrossRef]
21. Kajdanowicz, T.; Kazienko, P.; Indyk, W. Parallel processing of large graphs. *Neurocomputing* **2016**, *192*, 107–119. [CrossRef]
22. André, J.; Strati, F.; Klimovic, A. Exploring learning rate scaling rules for distributed ML training on transient resources. In Proceedings of the 3rd International Workshop on Distributed Machine Learning (DistributedML 2022), Rome, Italy, 9 December 2022; pp. 1–8.