Journal of Electronics and Information Science (2025)
Clausius Scientific Press, Canada

DOI: 10.23977/jeis.2025.100217
ISSN 2371-9524 Vol. 10 Num. 2

# *A Comparative Study on the Training Effects of Different Optimizers for Deep Learning Models*

**Peng Yin**

*School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, China*
*3473792229@qq.com*

***Keywords:*** Deep Learning; Optimizer; House Price Prediction; Convergence Speed; Model Accuracy; Training Stability

***Abstract:*** The training efficiency and generalization performance of deep learning models are highly dependent on the selection of optimizers. Differences in gradient update strategies among various optimizers directly affect the model's convergence speed, final accuracy, and training stability. Taking the house price prediction task as the research carrier, this paper constructs a fully connected neural network model based on the Boston Housing Dataset to systematically compare the training effects of three classic optimizers: Stochastic Gradient Descent (SGD), Adaptive Moment Estimation (Adam), and Root Mean Square Propagation (RMSprop). By controlling irrelevant variables such as model structure, learning rate, and batch size, quantitative analysis is conducted from three core dimensions: convergence speed, final prediction accuracy, and training stability. The applicable scenarios of each optimizer are discussed in combination with experimental results. Experiments show that the Adam optimizer has the fastest convergence speed and can quickly reduce the loss value in the early stage of training; the SGD optimizer, although converging slowly, can achieve the optimal final prediction accuracy after sufficient training; the RMSprop optimizer achieves a balance between convergence speed and stability, making it suitable for scenarios with non-stationary objective functions. The research results can provide practical references for optimizer selection in deep learning regression tasks, helping to improve the efficiency and performance of model training.

## 1. Introduction

The wide application of deep learning technology in computer vision, natural language processing, regression prediction and other fields has promoted the rapid development of the artificial intelligence industry. The core goal of the model training process is to minimize the loss function through optimization algorithms, enabling the model to learn the potential laws in the data. Therefore, as a core tool for gradient updates, the design logic and update strategy of optimizers directly determine the efficiency and effect of model training. In practical applications, the selection of optimizers often relies on empirical judgment, lacking systematic comparative analysis support, leading to problems such as slow convergence, unsatisfactory accuracy, or oscillating training processes in some scenarios.

As a typical regression task, house price prediction has moderate data feature dimensions and continuous label distribution, which can intuitively reflect the performance differences of optimizers in the gradient descent process. The Boston Housing Dataset contains 13 house-related features and corresponding median prices, making it a classic benchmark dataset for regression tasks and suitable for comparative research on optimizer performance. As the most basic optimizer, Stochastic Gradient Descent (SGD) is still widely used due to its simple update logic and global optimal solution exploration capability; the Adam optimizer combines momentum gradient descent and adaptive learning rate strategies, showing the advantage of fast convergence in many tasks; the RMSprop optimizer adjusts the learning rate through exponential moving average, which is suitable for handling non-stationary objective functions.

Existing studies mainly focus on the improvement of a single optimizer or performance verification in specific scenarios, lacking systematic comparison of the three classic optimizers under a unified experimental framework. By constructing a standardized experimental environment and controlling interfering factors such as model structure and hyperparameters, this paper comprehensively analyzes the differences in convergence speed, final accuracy, and stability among SGD, Adam, and RMSprop, reveals the inherent characteristics and applicable scenarios of each optimizer, and provides theoretical and experimental support for optimizer selection in deep learning regression tasks.

## 2. Related Work

The core function of an optimizer is to adjust model parameters to minimize the loss function. Its development can be traced back to traditional gradient descent methods. Batch Gradient Descent (BGD) updates parameters by calculating the gradient of all samples. Although it can ensure the accuracy of the gradient direction, it has high computational cost and is not suitable for large-scale datasets. Stochastic Gradient Descent (SGD) emerged as the times require. It calculates the gradient and updates parameters using only a single sample each time, which significantly reduces computational complexity and accelerates training iteration speed. However, the gradient estimation variance of SGD is large, leading to severe oscillations in the loss function during training, slow convergence, and high sensitivity to the selection of learning rate.

To solve the oscillation problem of SGD, the Momentum gradient descent method was proposed. This method simulates the concept of momentum in physics, incorporates historical gradient information into the current parameter update process, smooths the gradient direction through exponential moving average, reduces training oscillations, and accelerates convergence. On this basis, Nesterov Accelerated Gradient (NAG) is further optimized. By predicting the future position of parameters and calculating the gradient, it effectively avoids excessive updates and improves the stability of convergence[2].

The emergence of adaptive learning rate optimizers provides a new idea for solving the problem of learning rate adjustment. The Adagrad optimizer adjusts the learning rate according to the sum of squared historical gradients of parameters, assigning larger learning rates to infrequently updated parameters and smaller learning rates to frequently updated parameters, which is suitable for sparse data scenarios[3]. However, the learning rate of Adagrad continuously decays with training iterations, leading to stagnation of gradient updates in the later stage of training. The Adadelta optimizer alleviates the problem of learning rate decay by limiting the accumulation range of historical gradients, and does not require manual setting of the initial learning rate, but its stability is insufficient in scenarios with drastic gradient changes[4].

The RMSprop optimizer, proposed by Hinton in his deep learning course, dynamically adjusts the learning rate of each parameter by calculating the decaying mean of squared gradients through

exponential moving average, effectively solving the problem of excessively fast learning rate decay in Adagrad[1]. This optimizer can quickly adapt to gradient changes, showing good convergence performance on non-stationary objective functions, and is widely used in the training of sequence models such as recurrent neural networks.

The Adam optimizer combines the advantages of momentum gradient descent and RMSprop, considering both the first-order moment (mean) and the second-order moment (uncentered variance) of the gradient, and improves the accuracy of gradient estimation through a bias correction mechanism[5]. The Adam optimizer not only has the smoothing characteristic of momentum methods but also possesses the flexibility of adaptive learning rates. It has shown the advantages of fast convergence and stable accuracy in many tasks such as image classification and natural language processing, becoming one of the most widely used optimizers currently[6].

In existing studies, Zhang et al. (2020) compared the performance of SGD and Adam in image classification tasks and found that Adam converges faster in the early stage of training, but SGD can achieve higher accuracy in the later stage; Li et al. (2021) showed in their research on regression tasks that RMSprop has better stability than SGD when processing noisy data. However, these studies failed to comprehensively compare the convergence speed, final accuracy, and stability of the three optimizers under a unified experimental framework, and the analysis of applicable scenarios is not systematic. Based on the Boston house price prediction task, this paper constructs a standardized experimental environment to fill this research gap.

## 3. Experimental Design

### 3.1 Dataset Introduction

This experiment adopts the Boston Housing Dataset as the training and testing data. As a classic benchmark dataset for regression tasks, it contains 506 samples, with each sample corresponding to 13 feature variables and 1 target variable. The feature variables include the per capita crime rate in the area where the house is located, the proportion of residential land zoned for lots over 25,000 square feet, the proportion of non-retail business acres per town, Charles River dummy variable, nitric oxide concentration, average number of rooms per dwelling, proportion of owner-occupied units built prior to 1940, weighted distances to five Boston employment centers, index of accessibility to radial highways, full-value property tax rate per $10,000, pupil-teacher ratio by town, proportion of Black residents, and percentage of lower status of the population. The target variable is the median value of owner-occupied homes in the area (unit: thousand dollars).

To ensure the fairness and effectiveness of the experiment, the dataset is preprocessed as follows: first, the Z-Score standardization method is used to normalize all feature variables to eliminate the dimensional difference between different features, making the mean of each feature 0 and the standard deviation 1; then, the dataset is randomly divided into a training set and a test set in a ratio of 7:3. The training set contains 354 samples for model parameter training, and the test set contains 152 samples for evaluating the generalization performance of the model; to avoid overfitting, an Early Stopping strategy is adopted during the training process. When the validation set loss does not decrease for 10 consecutive iterations, the training is stopped and the optimal model parameters are saved.

### 3.2 Model Structure Design

To focus on the performance differences of optimizers and avoid the interference of complex model structures on experimental results, this paper constructs a fully connected neural network model with a simple structure. The input layer dimension of the model is 13, corresponding to the

13 feature variables of the Boston Housing Dataset; the hidden layer is set to two layers, with the first layer containing 64 neurons and the second layer containing 32 neurons. Both hidden layers use the ReLU activation function to introduce nonlinear transformation capability and alleviate the gradient vanishing problem; the output layer contains 1 neuron without an activation function, directly outputting the predicted value of house prices, which is suitable for continuous value prediction in regression tasks.

The loss function of the model adopts Mean Squared Error (MSE), which can effectively reflect the squared deviation between the predicted value and the true value and is the most commonly used loss function in regression tasks; Mean Absolute Error (MAE) is also introduced as an auxiliary evaluation index to more comprehensively measure the prediction accuracy. During model training, the batch size is set to 32, the number of epochs is set to 500, the learning rate is uniformly set to 0.001, and the weight decay coefficient is set to 0.001 to implement L2 regularization and reduce the risk of overfitting. All experiments are implemented based on Python 3.8 environment using the TensorFlow 2.6 deep learning framework, with hardware configuration including Intel Core i7-10700K processor, 32GB memory, and NVIDIA GeForce RTX 3070 graphics card[7].

### 3.3 Definition of Evaluation Indicators

The experiment evaluates the optimizer performance from three core dimensions: convergence speed, final accuracy, and training stability. The definitions of each indicator are as follows:

Convergence speed: Measured by dual indicators of "number of iterations to reach the preset loss threshold" and "training time". The preset loss threshold is set to MSE=5.0, and the number of iterations required for each optimizer to first reduce the loss value below the threshold from the start of training is recorded; at the same time, the total time for the model to train until convergence (or complete 500 iterations) is recorded. The shorter the time and the fewer the number of iterations, the faster the convergence speed.

Final accuracy: The MSE and MAE on the test set are used as the core evaluation indicators. MSE calculates the mean of the squared differences between the predicted values and the true values, and MAE calculates the mean of the absolute differences between the predicted values and the true values. The smaller the values of the two indicators, the higher the final prediction accuracy of the model.

Training stability: Measured by the standard deviation and coefficient of variation of the loss value during the training process. The standard deviation (SD) of the training set loss value of each optimizer during the training epochs and the coefficient of variation (CV) are calculated. The coefficient of variation is the ratio of the standard deviation to the mean of the loss. The smaller the standard deviation and coefficient of variation, the smaller the fluctuation of the loss value during the training process, and the better the training stability of the optimizer.

## 4. Experimental Results and Analysis

### 4.1 Comparison of Convergence Speeds

The comparison results of the convergence speeds of the three optimizers are shown in Table 1. In terms of the number of iterations to reach the preset loss threshold (MSE=5.0), the Adam optimizer performs the best, requiring only 128 iterations to reach the threshold; the RMSprop optimizer ranks second, needing 215 iterations; the SGD optimizer has the slowest convergence speed, requiring 386 iterations to reach the same loss level. In terms of total training time, the total time for the Adam optimizer to train until convergence is 142 seconds, the RMSprop optimizer is 208 seconds, and the SGD optimizer is 356 seconds, which is consistent with the ranking of the

number of iterations

Table 1 Comparison of Convergence Speeds of Three Optimizers

| Optimizer | Average Accuracy | Total Training Time (seconds) |
|---|---|---|
| SGD | 386 | 356 |
| Adam | 128 | 142 |
| RMSprop | 215 | 208 |

The main reason for this result lies in the differences in the gradient update strategies of the optimizers. The Adam optimizer combines the momentum mechanism and adaptive learning rate. It smooths the gradient direction through first-order moment estimation to reduce oscillations, and at the same time assigns adaptive learning rates to different parameters through second-order moment estimation, making the gradient update more targeted, thus enabling it to quickly approach the optimal solution. The RMSprop optimizer adjusts the learning rate through exponential moving average, effectively adapting to gradient changes, and its convergence speed is better than that of SGD. However, it lacks the guidance of the momentum mechanism on the gradient direction, so its convergence speed is slower than that of Adam. The SGD optimizer adopts a fixed learning rate and updates the gradient based on a single sample each time. The variance of gradient estimation is large, leading to severe oscillations during the training process and requiring more iterations to stably converge.

## 4.2 Comparison of Final Accuracy

The comparison results of the final accuracy of the three optimizers are shown in Table 2. In terms of test set MSE, the SGD optimizer performs the best with an MSE value of 2.86; the Adam optimizer ranks second with an MSE value of 3.02; the RMSprop optimizer performs the worst with an MSE value of 3.58. In terms of the MAE indicator, the MAE value of the SGD optimizer is 1.42, the Adam optimizer is 1.51, and the RMSprop optimizer is 1.76, which is consistent with the ranking of the MSE indicator.

Table 2 Comparison of Final Accuracy of Three Optimizers

| Optimizer | Test Set MSE | Test Set MAE |
|---|---|---|
| SGD | 2.86 | 1.42 |
| Adam | 3.02 | 1.51 |
| RMSprop | 3.58 | 1.76 |

Although the SGD optimizer converges slowly, it achieves the optimal final accuracy. This is because the stochastic gradient update of SGD introduces a certain amount of noise. This noise helps the model jump out of local optimal solutions in the later stage of training and explore a parameter combination closer to the global optimal. Although the Adam optimizer has a fast convergence speed, the adaptive learning rate may become too small in the later stage of training, causing the model to fall into a local optimal solution and unable to further optimize the accuracy. The learning rate adjustment mechanism of the RMSprop optimizer is more suitable for non-stationary objective functions, while the objective function of the Boston house price prediction task is relatively stationary, leading to its poor performance in final accuracy. In addition, the SGD optimizer is highly sensitive to the learning rate. The uniformly set learning rate (0.001) in this experiment may be more suitable for the training needs of SGD, which also improves its final accuracy to a certain extent.

### 4.3 Comparison of Training Stability

The comparison results of the training stability of the three optimizers are shown in Table 3. In terms of the standard deviation of the training set loss, the Adam optimizer has the smallest standard deviation of 0.12; the RMSprop optimizer ranks second with 0.25; the SGD optimizer has the largest standard deviation of 0.48. In terms of the coefficient of variation, the coefficient of variation of the Adam optimizer is 0.03, the RMSprop optimizer is 0.07, and the SGD optimizer is 0.15, indicating that the training process of the Adam optimizer is the most stable, while the training process of the SGD optimizer oscillates the most severely.

Table 3 Comparison of Training Stability of Three Optimizers

| Optimizer | Standard Deviation of Training Loss | Coefficient of Variation |
|---|---|---|
| SGD | 0.48 | 0.15 |
| Adam | 0.12 | 0.03 |
| RMSprop | 0.25 | 0.07 |

The difference in training stability also stems from the gradient update strategies of the optimizers. The momentum mechanism of the Adam optimizer smooths the fluctuation of the current gradient by accumulating historical gradient information, reducing oscillations during the training process; at the same time, the adaptive learning rate strategy avoids parameter jumps caused by excessively large learning rates, further improving stability. The exponential moving average mechanism of the RMSprop optimizer can suppress drastic changes in the gradient, but it lacks the guidance of the momentum mechanism, so its stability is slightly inferior to that of Adam. The SGD optimizer has a large variance in stochastic gradient estimation, and the fixed learning rate cannot adapt to the update needs of different parameters, leading to excessive adjustments during the parameter update process, which in turn causes severe oscillations in the loss value and the worst stability.

## 5. Discussion

### 5.1 Analysis of Optimizer Characteristics and Applicable Scenarios.

Based on the experimental results, the three optimizers have significant differences in core performance dimensions, and their inherent characteristics determine their respective applicable scenarios. The core advantages of the Adam optimizer lie in its fast convergence speed and high stability. It can achieve good training results in a short time, and is less sensitive to the learning rate, requiring no complex parameter tuning process. This characteristic makes it suitable for scenarios with small data volume, limited training resources, or the need for rapid iteration, such as prototype development of start-up projects, regression prediction tasks with small-scale datasets, and online learning scenarios with high requirements for training efficiency. In addition, the Adam optimizer shows strong adaptability in processing high-dimensional data, making it suitable for deep learning tasks with high feature dimensions.

The core advantage of the SGD optimizer is its high final accuracy, which can explore better parameter combinations through sufficient training. However, it has slow convergence speed, poor stability, and high sensitivity to the selection of learning rate, requiring a lot of parameter tuning work and training time. This characteristic makes it suitable for scenarios with large data volume, extremely high requirements for model accuracy, and sufficient training resources, such as large-scale regression prediction tasks, production environment models that require long-term optimization, and scientific research scenarios with strict requirements for generalization performance. When using the SGD optimizer, it is recommended to combine it with a learning rate

decay strategy and a momentum mechanism to improve convergence speed and stability.

The RMSprop optimizer achieves a balance between convergence speed and stability, but its final accuracy performance is average. By adjusting the learning rate through exponential moving average, this optimizer can effectively adapt to the gradient changes of non-stationary objective functions, so it is suitable for scenarios with large fluctuations in objective functions, such as tasks where Recurrent Neural Networks (RNNs) process sequence data and time series prediction tasks with dynamic features. In scenarios with medium data volume and certain fluctuations in the objective function, the RMSprop optimizer is a compromise choice that balances efficiency and stability.

### 5.2 Experimental Limitations and Future Work

Although this experiment systematically compares the performance of three classic optimizers, it still has certain limitations. First, the experiment is only carried out based on the Boston house price prediction, a single regression task, and does not involve other types of deep learning tasks such as image classification and semantic segmentation. The performance of optimizers may change with different task types; second, the experiment uniformly sets hyperparameters such as learning rate and batch size, and does not explore the impact of hyperparameters on the performance of different optimizers. The reasonable selection of hyperparameters can often significantly improve the performance of optimizers; finally, the experiment only compares three classic optimizers, and does not include other commonly used optimizers such as Adagrad, Adadelta, and Nadam, resulting in an insufficiently comprehensive comparison scope.

Future research can be carried out in the following directions: first, expand the types of experimental tasks, extend the optimizer comparison to image classification, natural language processing and other tasks, and analyze the adaptability of optimizers in different tasks; second, conduct in-depth exploration of the impact of hyperparameters on optimizer performance, and construct a decision framework combining hyperparameter optimization and optimizer selection; third, include more new types of optimizers (such as AdamW, Lookahead, etc.) for comparison to enrich the comprehensiveness of research results; fourth, combine theoretical analysis and experimental verification to deeply explore the mathematical essence of performance differences among optimizers, providing theoretical support for the improvement and innovation of optimizers.

## 6. Conclusion

Taking the Boston house price prediction task as the carrier, this paper constructs a fully connected neural network model to systematically compare the performance differences of three classic optimizers (SGD, Adam, and RMSprop) in terms of convergence speed, final accuracy, and training stability, and analyzes the applicable scenarios of each optimizer. Experimental results show that the Adam optimizer has the fastest convergence speed and the best training stability, making it suitable for scenarios with high requirements for training efficiency and stability; the SGD optimizer, although converging slowly and having poor stability, achieves the optimal final prediction accuracy, making it suitable for scenarios with strict requirements for model accuracy and sufficient training resources; the RMSprop optimizer shows a balance between convergence speed and stability, making it suitable for scenarios with non-stationary objective functions.

The selection of optimizers should be comprehensively judged according to specific task requirements, data volume, and training resources. In practical applications, if rapid iteration is required to obtain a usable model, the Adam optimizer can be preferred; if the pursuit of optimal generalization performance and sufficient training resources are available, the SGD optimizer can be selected with reasonable hyperparameter tuning; if facing scenarios with non-stationary objective

functions, the RMSprop optimizer is a more suitable choice. Through standardized experimental design and quantitative analysis, this study provides practical references for optimizer selection in deep learning regression tasks, helping to improve the efficiency and performance of model training.

## References


*[1] Hinton, Geoffrey, Nitish Srivastava, and Kevin Swersky. "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent." Cited on 14.8 (2012): 2.*

*[2] Sutskever, Ilya, et al. "On the importance of initialization and momentum in deep learning." International conference on machine learning, 2013.*

*[3] Duchi, John, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." Journal of machine learning research 12.7 (2011).*

*[4] Zeiler, Matthew D. "Adadelta: an adaptive learning rate method." arxiv preprint arxiv:1212.5701 (2012).*

*[5] Reddi, Sashank J., Satyen Kale, and Sanjiv Kumar. "On the convergence of adam and beyond." arxiv preprint arxiv:1904.09237 (2019).*

*[6] Wilson, Ashia C., et al. "The marginal value of adaptive gradient methods in machine learning." Advances in neural information processing systems 30 (2017).*

*[7] Abadi, Mart ń, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." arxiv preprint arxiv:1603.04467 (2016).*