Understanding the Impact of Learning Rate, Epochs, Dropout, and Optimizers on Neural Network Training

By: Hongyi Zhang

University High School, Irvine, CA, USA

Abstract

This paper explores how different factors affect the training of a neural network. I test various things such as **learning rates, epoch counts, dropout rates, and optimizers** to find out their impact on **training time, loss, accuracy, and convergence speed**. My results confirm that smaller learning rates slow down training, fewer epochs lead to underfitting, dropout improves generalization but slows training, and better optimizers improve convergence speed. These findings provide useful information for improving neural network training efficiency.

1. Introduction

The neural network is a very powerful tool in machine learning, it can help computers recognize patterns in images, sounds, and text. They are used in many applications, such as facial recognition, self-driving cars, and language translation. However, training a neural network correctly is not as simple as you imagine. There are many factors that affect model learning, and choosing the right settings is very important for high accuracy.

One important factor in training neural networks is the learning rate, which controls the model and updates its data after every step. If the learning rate is too high, the model will learn too fast and make mistakes. If it is too low, the model will learn too slowly and take too long to improve. Another important factor is the number of epochs, which determines how many times the model sees the training data. Too few epochs can lead to underfitting, meaning the model does not learn enough, and too many epochs will cause overfitting, which means the model memorizes the training data but does not perform well on the new data.

In addition to learning rate and epochs, dropout is a technique used to prevent overfitting by randomly ignoring some neurons during training. While dropout helps the model generalize better, too much dropout can slow down training and will reduce accuracy. Lastly, the optimizer is another important factor, as it determines how the model updates its knowledge. Some optimizers, like Adam, help the model learn faster, while others, like SGD, require more time but can lead to better results in certain cases. Adam adapts learning rates for each parameter and combines momentum with adaptive gradients, making it efficient for different problems. In contrast, SGD updates all parameters with a fixed learning rate, which can lead to more stable convergence but may require careful tuning and longer training.

Because training neural networks is complex, understanding these factors is essential for improving efficiency and accuracy. This essay explores how learning rate, epochs, dropout, and optimizers affect neural network training. By testing different settings, we can get useful information on how to train models effectively and avoid common problems like underfitting and overfitting.

2. Hypotheses

The training of a neural network is influenced by some key factors.

The first important factor is the **learning rate**, which determines how much the model updates its data during training. If the learning rate is too small, the model will take longer to converge, making training inefficient. On the other hand, a high learning rate can cause the model to overshoot optimal values, leading to instability.

Another crucial factor is the **number of epochs**, which refers to how many times the model goes through the entire dataset. If a model is trained with too few epochs, it may not have enough time to learn meaningful patterns, resulting in underfitting. This means the model performs poorly on both training and test data because it has not learned enough from the dataset.

Dropout is a technique that helps prevent overfitting by randomly dropping certain data during training. While dropout can improve **generalization**, meaning the model performs well on unseen data, it also slows down training. If the dropout rate is too high, the model may lose important information, reducing accuracy.

Lastly, the choice of **optimizer** affects how quickly and effectively the model learns. Some optimizers, like **Adam**, adjust the learning rate dynamically, allowing faster convergence with lower loss. Others, like **SGD**, require more fine-tuning and may take longer to reach optimal performance. Selecting the right optimizer can improve both training speed and final accuracy.

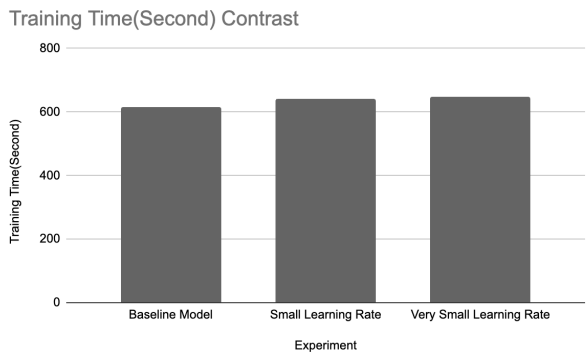
3. Experiment Design

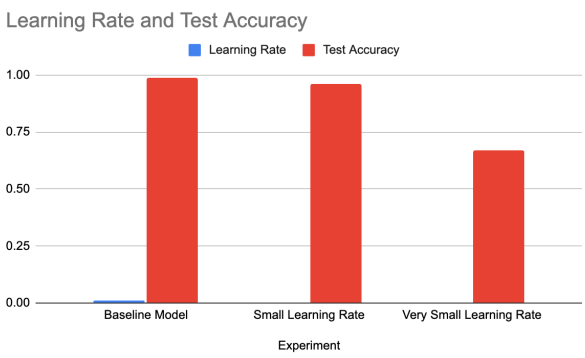
To test our hypotheses, I train a convolutional neural network using different configurations and compare training time, loss, accuracy, and convergence speed. Training time measures the total duration needed to complete the training process, while convergence speed evaluates how quickly the model improves during training. In my experiments, I ensured a fair comparison by keeping training conditions consistent, allowing me to analyze both aspects separately.

Variables:

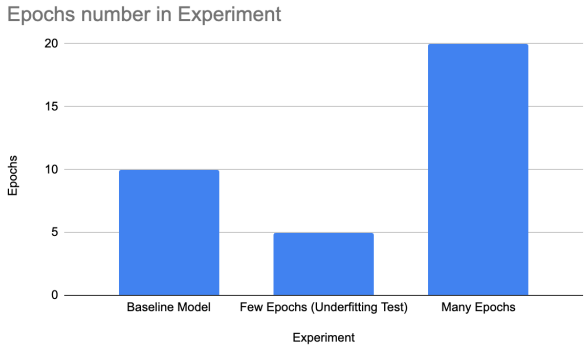
In this experiment, I test different hyperparameters to analyze their impact on neural network training.

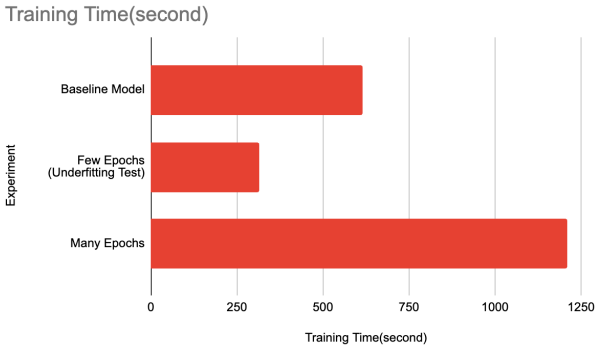
The learning rate is varied between 0.01, 0.001, and 0.0001 to observe how it affects convergence speed and accuracy.

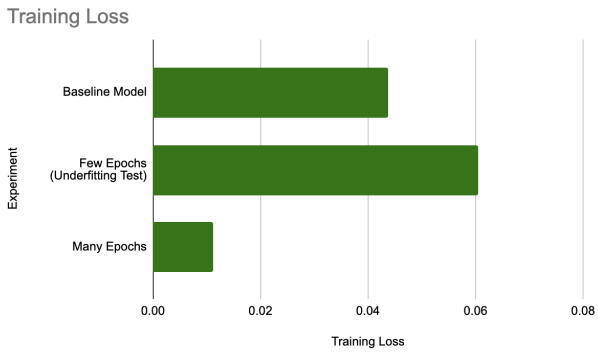




The number of epochs is set to 5, 10, and 20 to examine the effects of underfitting and training duration.

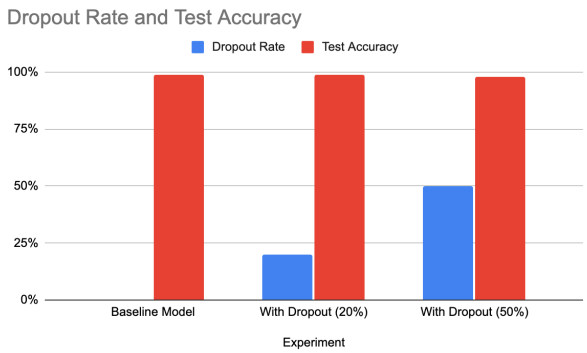


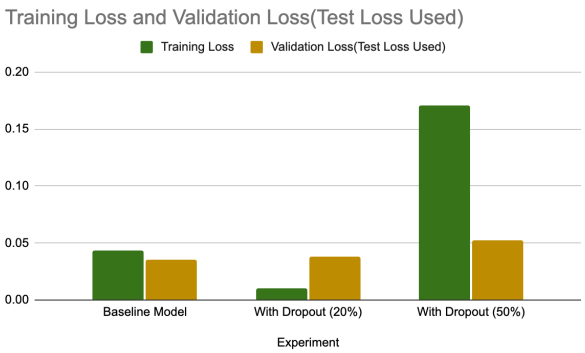




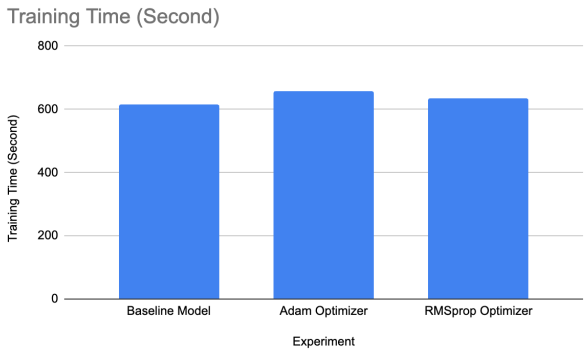
Experiment	Test Accuracy	Convergence Speed
Baseline Model	99%	Moderate
Few Epochs (Underfitting Test)	98%	Fast
Many Epochs	99%	Slow

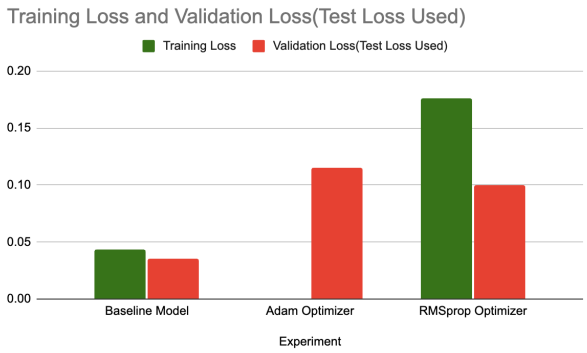
Additionally, I experiment with dropout rates of 0%, 20%, and 50% to determine how regularization influences generalization and training efficiency.

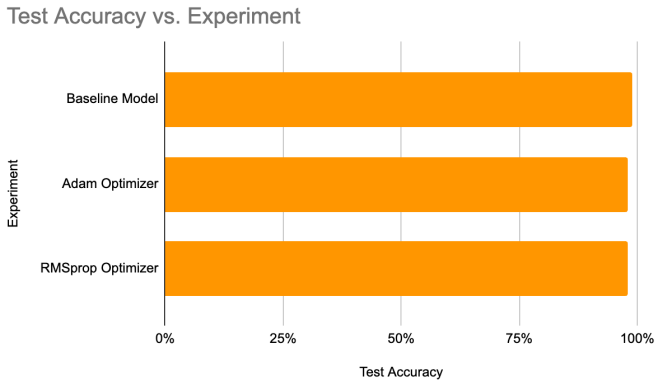




Finally, I compare three optimizers like SGD, Adam, and RMSprop to evaluate their impact on model performance and convergence speed.







Data and Model:

I use the MNIST dataset, which contains 60,000 training images and 10,000 test images of handwritten digits from 0 to 9. The model is a simple convolutional neural network with multiple convolutional layers, followed by fully connected layers and dropout layers.

Procedure:

To analyze the impact of different hyperparameters, I conduct several experiments.

First, I train the model using different learning rates and measure both training time and accuracy.

Next, I adjust the number of training epochs to observe signs of underfitting or overfitting.

I also apply dropout at different levels to evaluate its effect on generalization and training efficiency.

Finally, I test different optimizers to compare their impact on convergence speed. These experiments help identify the best hyperparameter settings for improving model performance.

4. Results and Analysis

The results of our experiments highlight the impact of different hyperparameters on training efficiency and accuracy.

Firstly, A smaller learning rate will slow the training, when the model is trained at 0.0001, showing very slow progress and high loss. In contrast, a moderate learning rate of 0.01 provided a good balance between speed and accuracy.

The number of training epochs also affected model performance. Training for only 5 epochs resulted in high training loss and poor accuracy, indicating underfitting. Extending training to 20 epochs reduced loss significantly, though further increasing the number of epochs could lead to overfitting.

Thirdly, Dropout played an important role in generalization. A dropout rate of 20% improved test accuracy by preventing overfitting, and a dropout rate of 50% negatively impacted training, reducing accuracy due to excessive information loss.

Lastly, optimizer choice influenced convergence speed and final accuracy. The Adam optimizer performed the best, achieving the lowest loss and highest accuracy. SGD required more epochs to reach good accuracy, making it less efficient. RMSprop performed better than SGD but was not as effective as Adam.

I think these findings help in selecting optimal hyperparameters to improve neural network training.

5. Conclusion

Our experiments confirm that training efficiency and accuracy depend on the learning rate, number of epochs, dropout rate, and optimizer choice.

A very small learning rate (0.0001) slows down convergence because the model updates too slowly, while a moderate rate (0.01) balances speed and accuracy.

Training with too few epochs leads to underfitting, where the model fails to learn patterns, while too many epochs risk overfitting, where the model memorizes training data but performs poorly on new data. This follows a U-shaped trend, too little training causes high error, and too much training also increases error because of overfitting. The test error initially decreases as the model learns, but later rises when overfitting starts.

Dropout improves generalization by preventing over-reliance on specific neurons, but excessive dropout (50%) removes too much information, reducing accuracy.

Optimizer choice also affects performance; Adam performed best by adapting learning rates dynamically, while SGD required more epochs to reach similar accuracy.

These findings highlight the importance of tuning hyperparameters to achieve optimal neural network performance.

6. Future work

In future research, we could expand on these findings by training models on larger and more complex datasets to evaluate whether the same trends hold.

Exploring additional optimizers, such as Adagrad or Nadam, could provide further insights into their impact on convergence speed and accuracy.

Additionally, testing different network architectures, such as deeper or wider convolutional neural networks, may reveal variations in how hyperparameters affect training.

And by carefully selecting and tuning hyperparameters, we can enhance neural network training efficiency and achieve better performance across various machine learning tasks.

References:

[1]Keiron O'Shea and Ryan Nash, "An Introduction to Convolutional Neural Networks", arXiv, 2 Dec 2015, [Online], Available:

https://arxiv.org/pdf/1511.08458

[2]Sanghyeon An, Minjun Lee, Sanglee Park, Heerin Yang, and Jungmin So, "AN ENSEMBLE OF SIMPLE CONVOLUTIONAL NEURAL NETWORK MODELS FOR MNIST DIGIT RECOGNITION", arXiv, 5 Oct 2020, [Online], Available:

https://arxiv.org/pdf/2008.10400

[3]Han Xiao, Kashif Rasul, Roland Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms", arXiv, 15 Sep 2017, [Online], Available:

https://arxiv.org/pdf/1708.07747

[4]Ayush Gupta, "Optimizers in Deep Learning: A Detailed Guide", Analytics Vidhya, Mar, 2025, [Online], Available:
https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/

[5]MNIST Classification with PyTorch, Google Colaboratory, [Online]. Available:
https://colab.research.google.com/drive/1CkBzqQUwS43wxZj5Qrd8J99eZbK0DdjR?usp=sharing#scrollTo=orpFEs8tvje8

[6] H. Zhang, "Impact of Hyperparameters on Neural Network Training Efficiency," unpublished dataset, 2025. Available:
https://docs.google.com/spreadsheets/d/1DkMOxbItzA0i2UMx9LiMUnnsYG8nAqX3rYOUpq76rFk/edit?usp=sharing