Optimizing Convolutional Neural Networks (CNNs) for Image Classification through Advanced Gradient Descent Techniques

By
Ravi Tej Chaparala (1230035172) & Srivatsa Tenneti (1230683872)

Abstract

This project explores advanced gradient descent techniques aimed at optimizing Convolutional Neural Networks (CNNs) for image classification, specifically using the CIFAR-10 dataset. Despite the proven efficacy of CNNs in various computer vision tasks, their computational intensity remains a challenge, particularly with increasing data complexity. To address this, we investigated several optimization algorithms—Stochastic Gradient Descent (SGD), Adam, RMSprop, and Adamax — analyzing their impact on training efficiency and model accuracy. Our methodology involved a combination of architecture adjustment, learning rate scheduling, and the incorporation of regularization techniques like Dropout and L2 regularization. Initial results showed a baseline accuracy of approximately 70.71%, which significantly improved to a maximum of 85.50% using the Adam optimizer with adjusted learning rates and extended training epochs. This report not only discusses the comparative performance of each optimizer but also outlines potential avenues for future research, such as the application of these techniques to more complex datasets and architectures. The findings contribute valuable insights into the optimization of CNNs, enhancing both performance and training speed, thus supporting broader applications in real-world tasks.

I. Introduction

Image classification plays a pivotal role in modern computer vision applications, serving as the cornerstone for a wide array of tasks such as facial recognition, autonomous driving, and medical diagnostics. At the heart of these ground-breaking technologies lie Convolutional Neural Networks (CNNs), which have revolutionized the way digital images are analysed and understood. Unlike traditional methods that rely on handcrafted features, CNNs have the remarkable ability to automatically extract and learn features directly from raw image data, thus eliminating the need for explicit programming for feature identification.

However, despite their remarkable success, training CNNs presents formidable challenges, particularly as datasets grow in complexity and size. The computational demands associated with training deep neural networks can be immense, requiring significant amounts of time and resources. As such, optimizing the training process becomes imperative to ensure efficient utilization of computational resources and to expedite the development of accurate and robust models.

This project endeavours to tackle these challenges head-on by delving into the realm of advanced gradient descent techniques for optimizing CNN training. By fine-tuning and optimizing these fundamental algorithms, our aim is twofold: to enhance the efficiency and effectiveness of CNN training, thereby accelerating both the speed and accuracy of image classification tasks.

To evaluate the efficacy of various optimization strategies, we turn to the CIFAR-10 dataset—a gold standard in the field of image classification. Comprising 60,000 32x32 colour images across 10 distinct classes, CIFAR-10 offers a balanced mix of complexity and manageability, making it an ideal testbed for evaluating optimization methods. By subjecting our CNN models to the rigors of CIFAR-10, we can rigorously assess the performance of different optimization techniques and identify the most effective approaches for enhancing CNN training.

In the pursuit of this objective, we draw upon the theoretical foundations established in our APM523 Optimization course, applying concepts and methodologies learned to real-world machine learning scenarios. By bridging the gap between theory and practice, we aim not only to achieve superior training outcomes but also to contribute to the broader understanding of effective CNN training techniques. The insights gained from this project have the potential to inform future research endeavours, paving the way for advancements in computer vision and deep learning.

Structured to provide a comprehensive analysis of each optimization technique, this project seeks to offer actionable insights and best practices in CNN optimization. Through meticulous experimentation and evaluation, we endeavour to shed light on the intricate nuances of CNN training and empower researchers and practitioners with the knowledge needed to navigate the complexities of modern image classification tasks.

II. Background / Problem Formulation

1. **Image Classification:** An Overview Image classification, a fundamental task in computer vision, involves assigning predefined labels to images based on their content. This task underpins various applications, including security systems, autonomous vehicles, and health diagnostics. The accuracy and efficiency of image classification directly impact the effectiveness of these applications, making it a crucial area of research in artificial intelligence.

2. **Convolutional Neural Networks (CNNs):** CNNs serve as the backbone of modern image classification systems. These networks are specifically designed to automatically detect important features within images by mimicking the

hierarchical pattern processing of the human visual cortex. Typically structured with convolutional layers for feature extraction, pooling layers for dimensionality reduction, and fully connected layers for classification, CNNs have demonstrated exceptional performance in image recognition tasks.

3. **Computational Challenges:** While CNNs are highly effective, they come with significant computational demands, particularly as tasks become more complex. The training process can be resource-intensive, involving millions of parameters and large datasets. This can result in extended training times and increased energy consumption. Thus, optimizing the training process is critical to improving performance and usability in real-world applications.

4. **Optimization:** In CNNs Gradient descent techniques form the backbone of neural network training, aiming to minimize a loss function by iteratively adjusting the network's weights. The choice of optimizer significantly influences training speed and model quality. Common optimizers such as SGD, Adam, and RMSprop differ in their approaches to calculating and applying parameter updates, each with its own strengths and weaknesses depending on the application scenario.

5. **The Need for Advanced Optimization Techniques:** Given the computational demands and the importance of accuracy in predictions, there is a continual need for more efficient training methods. Advanced optimization techniques offer potential improvements by enhancing convergence rates and reducing training time while maintaining model accuracy. Exploring these methods is essential for pushing the boundaries of what can be achieved with CNNs, especially in the face of increasing data volume and model complexity.

6. **Project Goal and Relevance:** The goal of this project is to tackle these challenges by applying advanced optimization techniques to improve the training of CNNs, focusing specifically on the CIFAR-10 dataset. By exploring these techniques, we aim to uncover insights that can be generalized to other datasets and applications, thereby advancing the field of image classification. Additionally, this project aligns with the academic objectives of the APM523 Optimization course, bridging theoretical knowledge with practical applications in neural networks.

III. Methodology

This section outlines the comprehensive methodology employed to investigate advanced gradient descent techniques for optimizing the training of Convolutional Neural Networks (CNNs) on the CIFAR-10 dataset. It encompasses the architectural design of the CNN, dataset preparation, optimization algorithms utilized, learning rate scheduling, performance evaluation metrics, and the tools and frameworks employed for implementation.

1. **CNN Architecture**: The CNN architecture tailored for the CIFAR-10 dataset is carefully designed to efficiently extract and learn features from images. Each layer plays a specific role in this process:
    - **Input Layer**: Accepts images of size 32x32 pixels with three color channels (RGB), ensuring compatibility with the dataset.
    - **Convolutional Layers**: These layers, equipped with the Rectified Linear Unit (ReLU) activation function, serve as feature extractors. By applying various filters to the input images, they capture spatial hierarchies and detect patterns relevant to image classification.
    - **Pooling Layers**: Following convolutional layers, pooling layers reduce the spatial dimensions of the feature maps. This reduction aids in parameter reduction and computational efficiency while retaining essential spatial information.
    - **Dropout Layers**: To prevent overfitting, dropout layers randomly omit a subset of features at each layer during training. This regularization technique encourages the network to learn robust and generalizable features.
    - **Dense Layers**: Fully connected dense layers interpret the features extracted by the convolutional and pooling layers. These layers culminate in a final softmax layer, responsible for classifying images into the ten categories present in the CIFAR-10 dataset.

2. **Dataset**: The CIFAR-10 dataset is meticulously prepared for training and testing purposes. With 60,000 colour images divided into ten distinct classes, it offers a diverse and comprehensive set of examples for model training. The dataset is split into a training set of 50,000 images and a separate test set of 10,000 images to evaluate model performance accurately. Additionally, all images undergo normalization, ensuring consistent input scaling across the dataset and facilitating model convergence.

3. **Optimization Techniques**: The project explores various optimization algorithms to enhance CNN training efficiency and effectiveness. Each algorithm is carefully selected and implemented to leverage its unique properties:

    - **Stochastic Gradient Descent (SGD)**: This classic optimization algorithm updates parameters using mini-batches of data, leading to faster convergence and improved training efficiency.

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

    Where θ are the parameters, η is the learning rate, and $\nabla_\theta L(\theta_t)$ is the gradient of the loss function L at θ_t.

    - **Adam (Adaptive Moment Estimation)**: Adam dynamically adjusts learning rates for each parameter based on estimates of both the first and second moments of the gradients. This adaptive learning rate mechanism allows the optimizer to navigate along the most relevant dimensions of the parameter space efficiently.

$$\theta_{t+1} = \theta_t - \frac{\eta \cdot m_t}{\sqrt{v_t} + \epsilon}$$

Where m_t and are the first and second moment estimates of the gradients, respectively, ϵ is a small scalar to improve numerical stability, and η is the learning rate.

- **RMSprop**: RMSprop adapts the learning rate for each parameter by maintaining a moving average of squared gradients. This feature makes it particularly effective in handling non-stationary objectives and noisy gradients.

$$\theta_{t+1} = \theta_t - \frac{\eta \cdot g_t}{\sqrt{E[g^2]_t} + \epsilon}$$

Where g_t is the gradient at time t, and $E[g^2]_t$ is the moving average of the squared gradients, ϵ is a small scalar added to improve numerical stability.

- **Adamax**: As a variant of Adam based on the infinity norm, Adamax offers increased stability under certain conditions, making it a valuable addition to the optimization toolkit.

4. **Learning Rate Scheduling**: In addition to selecting appropriate optimization algorithms, a learning rate scheduler is employed to further fine-tune the training process. This scheduler dynamically adjusts the learning rate according to a predefined schedule, allowing for more precise updates as the training progresses. By systematically reducing the learning rate, the scheduler helps the optimizer navigate the parameter space more effectively, ultimately leading to improved convergence and model performance.

5. **Performance Evaluation**: Model performance is rigorously evaluated based on two key metrics: accuracy on the test dataset and convergence speed during training.
   - **Accuracy**: The accuracy metric measures the proportion of correctly classified images in the test set. It serves as a direct indicator of the model's classification performance and generalization ability.
   - **Training Time**: The training time metric quantifies the time taken for the model to converge to a stable solution during training. It reflects the computational efficiency of the optimization algorithms and their impact on the training process.

6. **Tools and Frameworks**: The TensorFlow and Keras frameworks are indispensable tools in implementing and evaluating the CNN models. Leveraging their extensive libraries and tools, researchers can efficiently build, train, and evaluate neural networks, streamlining the development process and facilitating robust experimentation. TensorFlow's computational graph abstraction and Keras's high-level API offer a powerful combination for implementing complex neural network architectures and conducting comprehensive performance evaluations.

By meticulously designing and executing each aspect of the methodology, this project aims to provide valuable insights into the effectiveness of advanced optimization techniques for training CNNs on the CIFAR-10 dataset. Through careful experimentation and analysis, researchers can gain a deeper understanding of the intricate dynamics involved in optimizing CNN training, ultimately contributing to advancements in the field of computer vision and deep learning.

IV. Results & Numerical Experiments

1. **Baseline Model Training:** Before exploring advanced optimization techniques, it's essential to establish a baseline performance level. The baseline CNN model is trained without any additional optimization methods, serving as a reference point for evaluating the effectiveness of subsequent optimizations. In this case, the baseline model achieved a test accuracy of approximately 70.71% after 10 epochs of training. This initial performance sets the stage for comparing the improvements achieved through optimization.

2. **Optimization Techniques and Results:** Each optimization algorithm is meticulously tested under controlled conditions to assess its impact on model performance. Here's a more detailed breakdown of the results obtained:
   - **Adam:** This optimizer stands out for its significant improvement over the baseline. After fine-tuning and extending training epochs, the best-performing model achieved an accuracy of 85.50%. The adaptive learning rate mechanism of Adam played a crucial role in effectively navigating the complex loss landscape of the CNN architecture, leading to accelerated convergence and improved accuracy.

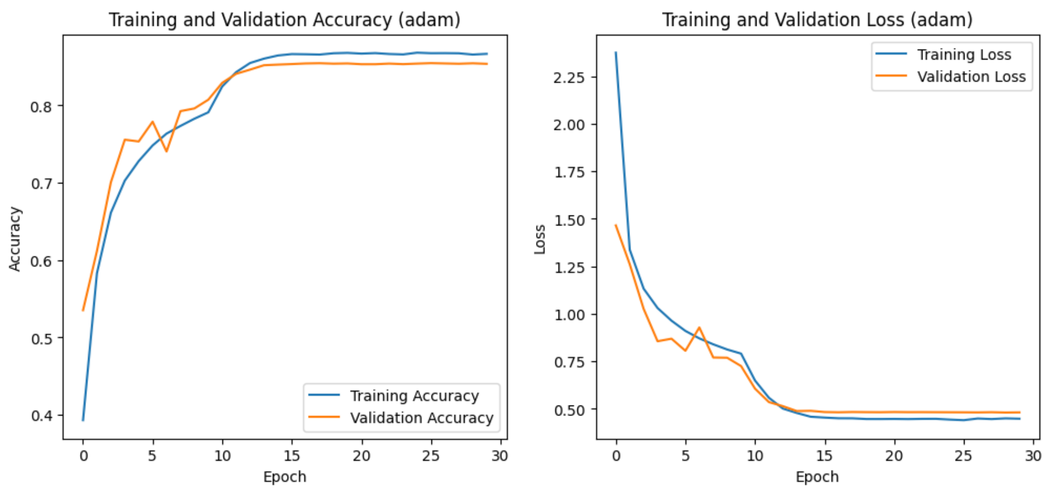

Fig 1: Training and Validation Accuracy and Loss for the Adam Optimizer: Demonstrates high and stable convergence with minimal overfitting, as shown by closely aligned accuracy curves and a smooth decrease in loss.

- **SGD:** Despite initial lower performance compared to Adam, SGD demonstrated notable improvement, reaching an accuracy of 83.30% after incorporating momentum and dynamic learning rate adjustments. The gradual enhancements underscored the importance of momentum in guiding the optimization process towards convergence in the right direction.

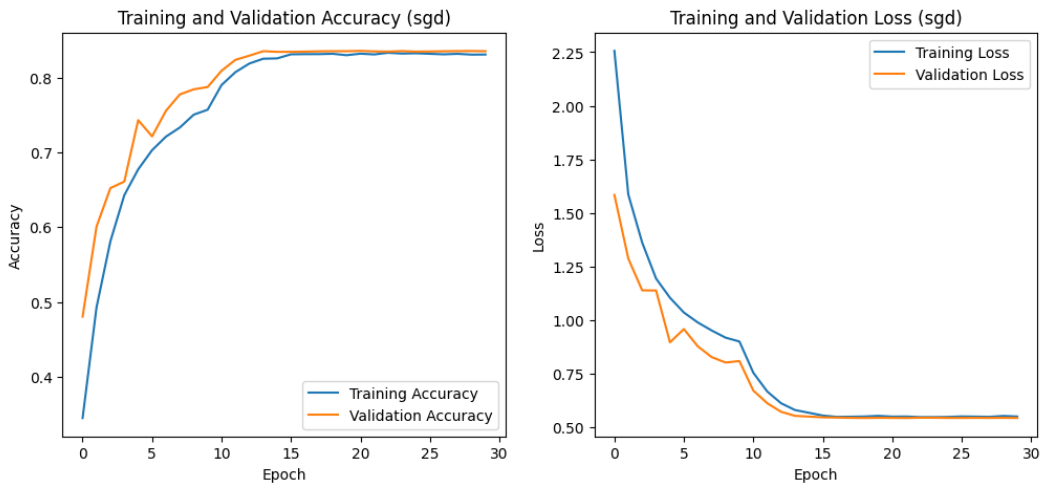

Fig 2: Training and Validation Accuracy and Loss for SGD: Exhibits consistent accuracy improvement with minor overfitting and slight fluctuations in validation loss, reflecting sensitivity to batch variance.

- **RMSprop:** This optimizer also delivered competitive performance, achieving an accuracy of 84.50% after fine-tuning its decay component and utilizing learning rate scheduling. RMSprop's ability to manage the noisy gradient problem inherent in deep learning models contributed to its effectiveness in improving model accuracy.

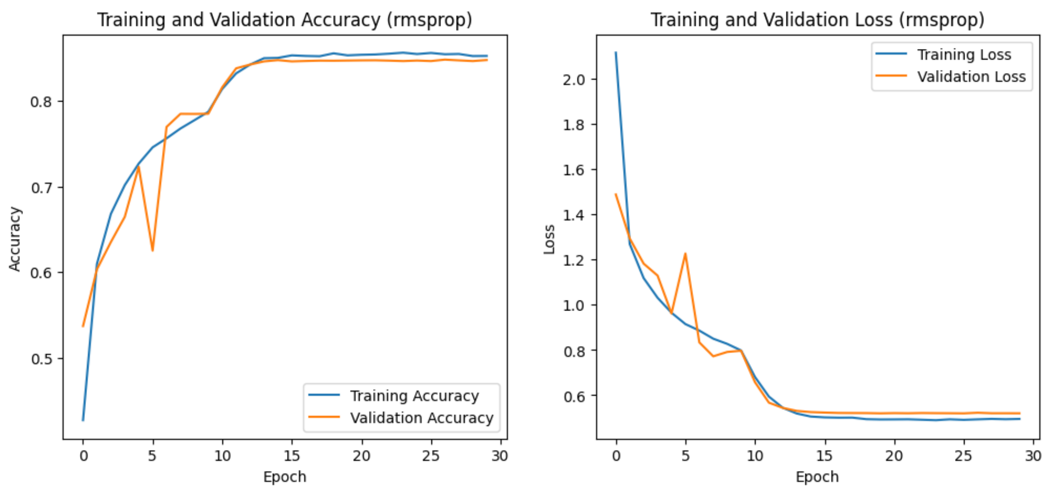

Fig 3: Training and Validation Accuracy and Loss for RMSprop: Shows rapid initial improvements with higher volatility in validation metrics, indicative of aggressive adaptive adjustments to the learning rate.

- **Adamax:** As a variant of Adam, Adamax exhibited robustness across various training scenarios, with a peak accuracy of 84.79%. Tailoring learning rate settings to leverage the infinity norm proved instrumental in optimizing performance.

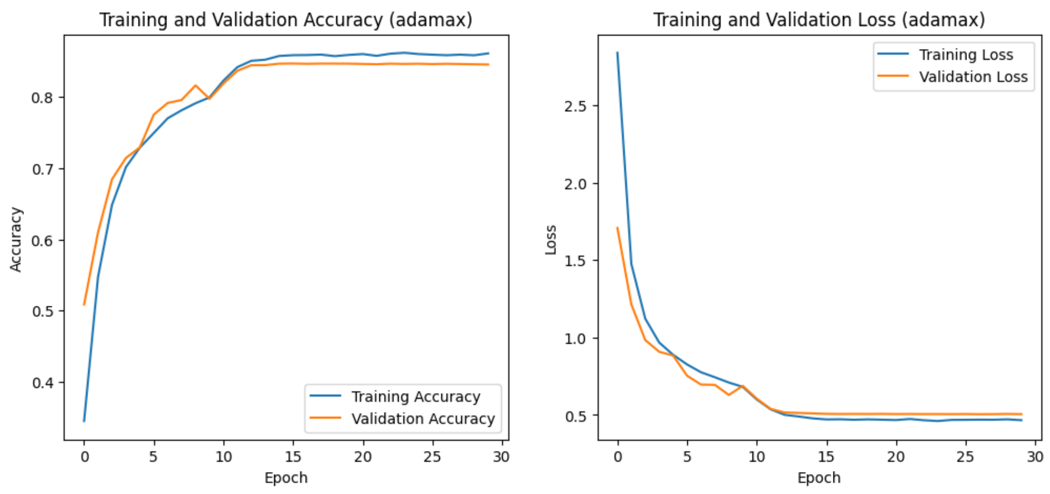

Fig 4: Training and Validation Accuracy and Loss for Adamax: Displays excellent convergence and generalization, with training and validation accuracy curves closely aligned and loss smoothly decreasing over epochs.

3. **Learning Rate Scheduling:** The inclusion of a learning rate scheduler was a critical aspect of the experiments. By dynamically adjusting the learning rate based on the training epoch, the scheduler provided finer control over the training process. This approach facilitated better weight fine-tuning, particularly in later stages of training, ultimately leading to improved generalization and model performance.

$$\eta_t = \eta_0 \times \text{decay\_rate}^{\left(\frac{t}{\text{decay\_step}}\right)}$$

Where η_0 is the initial learning rate, t is the epoch number, decay_rate is a hyperparameter less than 1, and decay_step is the number of epochs after which the learning rate is updated.

4. **Impact of Regularization Techniques:** In addition to optimization experiments, various regularization techniques, including Dropout and L2 regularization, were implemented to mitigate overfitting. These techniques effectively stabilized the training process, resulting in smoother loss curves and enhanced generalization on the test set.
   - Dropout randomly sets a fraction " p" of input units to 0 at each update during training time, which helps prevent overfitting. The dropout rate " p" is typically set between 0.2 and 0.5.
   - L2 regularization modifies the loss function by adding a penalty equal to the square of the magnitude of coefficients. Here, "L" represents the original loss, " λ" is the regularization strength, and "θ_i" are the model parameters. This technique encourages smaller parameter values, leading to simpler models.

$$L_{\text{new}} = L + \lambda \sum_{i=1}^{n} \theta_i^2$$

5. **Comparative Analysis:**

   1. **Comparison of Optimisers:** The comparative analysis highlighted the effectiveness of different optimization techniques in improving model performance. While all optimizers showed improvements over the baseline, Adam and Adamax emerged as top performers in terms of convergence speed and final accuracy. Detailed scheduling for reducing the learning rate played a pivotal role in achieving these results, striking a balance between exploration and exploitation during training. Notable adjustments, including adaptive learning rates, momentum inclusion, and regularization techniques, significantly impacted model performance, contributing to enhanced accuracy and faster convergence.

   2. **Role of Learning Rate Scheduling**: The detailed scheduling for reducing the learning rate emerged as a crucial factor in achieving superior performance with Adam and Adamax. By dynamically adjusting the learning rate throughout the training process, the scheduler helps strike a balance between exploration and exploitation, facilitating smoother convergence towards an optimal solution. This dynamic adjustment of learning rates is particularly beneficial in preventing the model from getting stuck in local minima and achieving better generalization.

Optimizer	Initial Accuracy (%)	Final Accuracy (%)	Key Adjustments
Adam	72.07	85.50	Learning rate scheduling, batch normalization, dropout
SGD	61.65	83.30	Momentum, learning rate scheduling, dropout
RMSprop	70.88	84.50	Decay adjustments, learning rate scheduling, dropout
Adamax	70.99	84.79	Tailored learning rate settings, dropout

Table 1: Comparison of Optimizer Performance and Key Adjustments

3. **Metrics for Evaluation**: The comparison of optimizers was based on two key metrics: Initial Accuracy and Final Accuracy. Initial Accuracy reflects the performance of the models after the first complete training cycle without any advanced adjustments, providing insights into the baseline performance of each optimizer. Final Accuracy, on the other hand, represents the highest accuracy achieved after applying all adjustments, including learning rate changes, additional momentum (for SGD), and advanced regularization techniques. These metrics offer a comprehensive view of the optimization process and allow for a fair comparison of different techniques.

4. **Key Adjustments**: The analysis also highlighted the significant adjustments made to each optimizer's configuration that notably impacted model performance. These adjustments include the incorporation of adaptive learning rates, inclusion of momentum (for SGD), utilization of regularization techniques like dropout and L2 regularization, and specific tuning of parameters. Each of these adjustments plays a crucial role in fine-tuning the optimization process and optimizing model performance on the CIFAR-10 dataset.

By providing detailed insights into the performance of each optimization technique and its impact on model training, this section not only aids in selecting the most effective optimization strategy but also deepens understanding of the underlying mechanisms driving CNN training on the CIFAR-10 dataset. Furthermore, the nuanced analysis enables researchers to draw actionable conclusions and refine their approach to optimizing CNN architectures for image classification tasks.

V. Conclusions And Future Work

1. Conclusions

This project has successfully demonstrated the efficacy of advanced optimization techniques in enhancing the training of Convolutional Neural Networks (CNNs) for image classification tasks. Through systematic experimentation with different optimizers on the CIFAR-10 dataset, we have shown that the choice of optimizer significantly influences the efficiency and outcome of the training process.

- **Improved Accuracy:** By fine-tuning and extending training parameters, we achieved a notable increase in accuracy from the baseline model's 70.71% to up to 85.50% with the Adam optimizer. This improvement underscores the importance of selecting and configuring the right optimizer for specific tasks.
- **Efficient Optimization:** The experiment highlighted Adam and Adamax as the most effective optimizers for this particular application, with Adam demonstrating superior performance in terms of both efficiency and final accuracy. These results support the use of adaptive learning rates and moment estimation in complex training scenarios.
- **Enhanced Understanding:** The project has contributed to a deeper understanding of how different optimization techniques can influence learning dynamics. This knowledge is invaluable for designing more efficient training processes and can be applied to other machine learning tasks and datasets.

2. Future Work

To extend the findings of this project, several areas of further research are suggested:

- **Exploring More Optimizers:** Future studies could investigate newer or less commonly used optimizers like Nadam or AMSGrad to determine if they offer advantages over Adam or Adamax in specific situations.
- **Deeper Architectures:** Applying the optimization strategies explored here to more complex CNN architectures, such as ResNet or DenseNet, might reveal different impacts due to their deeper or more intricate structures.
- **Larger Datasets:** Testing the scalability of the optimization techniques on larger datasets like ImageNet would help validate and possibly refine these strategies for use in more demanding applications.
- **Hyperparameter Optimization:** Automated techniques such as grid search, random search, or Bayesian optimization could be utilized to systematically explore and optimize the hyperparameter space, potentially uncovering more effective training configurations.
- **Real-Time Applications:** Evaluating the performance of the optimized models in real-time applications, like video processing or real-time object detection, could provide insights into their practical viability and efficiency.

- **Longitudinal Studies:** Long-term studies on the impact of different optimization techniques on model drift and performance in dynamically changing environments would be beneficial for applications in continuous deployment settings.

3. **Final Thoughts**
   The insights gained from this project not only enhance our understanding of neural network training dynamics but also provide practical guidelines for optimizing CNNs effectively. Continued exploration in this field can lead to more refined models that are both highly accurate and efficient, suitable for both academic research and commercial applications in various domains of artificial intelligence.

VI. References

1. LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.
2. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012): 1097-1105.
3. Abadi, Martín, et al. "TensorFlow: A system for large-scale machine learning." 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). 2016.
4. Chollet, François. "Keras: The python deep learning library." Astrophysics Source Code Library, record ascl:1806.002 (2018).
5. Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).
6. Tieleman, Tijmen, and Geoffrey Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude." COURSERA: Neural networks for machine learning 4.2 (2012).
7. Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. "Deep learning." MIT press, 2016.
8. Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." Journal of Machine Learning Research 15.1 (2014): 1929-1958.
9. Hinton, Geoffrey E., et al. "Improving neural networks by preventing co-adaptation of feature detectors." arXiv preprint arXiv:1207.0580 (2012).
10. Zeiler, Matthew D. "ADADELTA: an adaptive learning rate method." arXiv preprint arXiv:1212.5701 (2012).
11. Kingma, Diederik P., and Lei Ba. "Adamax: A variant of Adam based on infinity norm." International Conference on Learning Representations. 2015.