PERFORMATIVE ANALYSIS OF OPTIMIZERS ON FINE-TUNED LUNG DISEASE CLASSIFICATION *

Ray Hu	**Jiandong Guan**	**Cong Wei**	**Suhas Raghavendra**
rhhu1@asu.edu	jguan16@asu.edu	cwei35@asu.edu	sragha23@asu.edu
Group 15	Group 15	Group 15	Group 15

1 Abstract

This project evaluates the impact of optimizers on fine-tuning machine learning models for lung disease classification. We tested three optimizers—SGD (Stochastic Gradient Descent), AdamW, and Adadelta—on three image classification models: ZF-Net (a CNN), Sequencer2d (an LSTM-based model) and Resnet-50 (a Residual Neural Network). The dataset, augmented to 10,000 images, consisted of normal lung X-rays and cases of bacterial pneumonia, COVID-19, tuberculosis, and viral pneumonia. Performance metrics included training loss, validation loss, and the weighted F1 score.

Key findings indicate that AdamW performed well under default settings, SGD excelled in configurations with larger batch sizes, and Adadelta struggled to converge in most cases. This work emphasizes the importance of choosing the right optimizer and fine-tuning hyperparameters for medical diagnostic models, offering a foundation for building accurate and efficient tools for lung disease detection.

2 Introduction

2.1 Motivation

Machine learning has revolutionized computing by tackling problems once deemed unsolvable. At its core, the success of these models hinges on their ability to find optimal parameters through rigorous training. This process, however, demands significant computational resources. Optimizers play a critical role in accelerating this training by guiding models toward these optimal parameters efficiently.

In fields like medicine, particularly in lung disease diagnosis, precision and reliability are paramount. Doctors cannot rely on generic machine learning models due to the high stakes involved in identifying abnormalities. This study explores the role of optimizers in enhancing the performance of image classification models, specifically tailored for lung disease detection using chest X-rays.

2.2 Importance

Lung diseases such as bacterial pneumonia, COVID-19, tuberculosis, and viral pneumonia affect millions worldwide. Early and accurate diagnosis can significantly improve treatment outcomes, but traditional diagnostic methods can be time-consuming and prone to errors. Machine learning models have the potential to revolutionize this space by offering fast and accurate predictions. However, their effectiveness depends on the proper selection of optimizers and hyperparameter configurations.

By systematically evaluating popular optimizers like SGD, AdamW, and Adadelta across various hyperparameters, this project aims to identify the optimal configurations for medical imaging tasks. This not only improves diagnostic accuracy but also sets a benchmark for future research in applying machine learning to critical medical problems.

* *Citation*: **Hu et al. Performative Analysis of Optimizers on Fine-tuned Lung Disease Classification.**

3 Project Description

3.1 Objective

The goal of this project is to evaluate how different optimizers affect the performance of machine learning models designed for lung disease classification. We achieve this by using chest X-ray images to train machine learning models that can accurately classify lung conditions such as bacterial pneumonia, COVID-19, tuberculosis, and viral pneumonia. To achieve this, we designed a systematic approach that involves testing multiple optimizers on two types of models: a convolutional neural network (CNN) called ZF-Net, a Long Short-Term Memory (LSTM)-based model called Sequencer2d and a Residual neural network (RESNET) in which layers learn residual functions with reference to the layer inputs. These models are suitable for handling image data and excel in tasks requiring pattern recognition and classification.

3.2 Methodology

The dataset, sourced from public repositories, comprises 10,000 augmented chest X-ray images categorized into normal and diseased lungs. To prepare the data for training, preprocessing steps such as resizing images to a standard input size, normalizing pixel values, and converting them into a machine-learning-compatible format were employed. Additionally, data augmentation techniques, including flipping and rotation, were applied to enhance dataset variability and improve model generalization while mitigating overfitting.

The ZF-Net was selected for its efficiency and simplicity in feature extraction, while Sequencer2d offered robust temporal analysis suitable for sequence-based features. ResNet-50 was included for its proven hierarchical feature extraction and pre-trained capabilities, aligning with the computational and dataset constraints. Key performance metrics like training loss, validation loss, and the weighted F1 score were used to monitor model behavior and guide optimization.

3.3 Related Works

Several papers have been published regarding the performance of different optimizers. One of these papers [1] stood out to us as it made an interesting observation that while adaptive optimizers such as Adam were able to converge faster, it would generalize worse than other optimizers such as SGD. In this research, we also hope to replicate their findings.

4 Technical Background

4.1 Classification Models

We researched various machine learning models for image classification, focusing on models such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory Networks (LSTM). These models are well-suited for processing image data like x-ray diagrams, which makes them ideal for detecting lung abnormalities.

4.1.1 ZF-Net: A CNN Based Learning Model

We chose one such CNN model known as ZF-Net, developed by Mathhew D. Zeiler and Rob Fergus. This model became the best candidate for this project as this model is an improvement over the widely known Alex-Net but was still within the computational limits of this project. Furthermore, the research paper associated with this model [2] visualized the layer and component breakdown for CNNs and offered insights to how CNNs can be fine-tuned.

For this model, the paper [2] describes the input images are reshaped to a size of 224x224x3 and fed into five layers of convolution layers. Afterwards, the output of this is fed into two fully connected layers with a hidden size of 4096 units. We modified the model so that the output layer has our desired number of classes. We will be using 50 epochs to train this model.

4.1.2 Sequencer2d: A LSTM (RNN) Based Learning Model

We chose a LSTM based model known as Sequencer2d from the Hugging face library. In the paper [3] associated with the model, it was developed by Yuki Tatsunami and Masato Taki to challenge the performance of transformer based classifiers by replacing a tranformer's self-attension layer with a bi-direction LSTM layer. This model became a good candidate for us as it is a deep LSTM model which can make accurate image classifications, but was also within the

computational limit of this project. Moreover, this model can be found in the hugging face library so implementing this complex model proved to be relatively simple.

For this model, the images are also resized to 224x224x3 where they are fed into a transformer like pathway. As mentioned above and in the paper [3], the main difference is that the self-attention blocks are replaced by the authors' bi-directional LSTM "Sequencer" blocks. We will be using 50 epochs to train this model.

4.1.3 RESNET-50: Residual Neural Network

We chose a convolutional neural network (CNN) model known as ResNet-50 for this project, leveraging its pre-trained version from the PyTorch torchvision library. ResNet-50 was introduced by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in the paper Deep Residual Learning for Image Recognition [4], was designed to address the vanishing gradient problem in deep networks using residual connections. These connections allow the model to learn identity mappings, facilitating the training of very deep networks. ResNet-50 has 50 layers, including convolutional, batch normalization, and activation layers grouped into residual blocks.

For this project, we modified the final fully connected layer of ResNet-50 to classify images into five categories relevant to our lung disease dataset. This model became an ideal candidate due to its strong ability to capture hierarchical image features and its proven performance on image classification tasks. Additionally, by fine-tuning the pre-trained model, we achieved computational efficiency while maintaining high accuracy, making it well-suited for our dataset and the constraints of this project.

4.2 Optimizers

In machine learning, there exist several optimizers we can use and compare. Because each has their own general pros and cons, we can use this to help determine which hyperparameters values to look out for in the learning models.

4.2.1 SGD

Stochastic Gradient Descent (SGD) is a variant of the traditional gradient descent optimization technique. Instead of using the entire dataset to calculate the gradient at each step, SGD uses a small, randomly selected portion of the data, called a mini-batch. This makes the process faster and less memory-intensive, which is particularly helpful when working with large datasets. However, this method can result in noisier updates, making the convergence process less smooth. While SGD is mostly used on models such as Support Vector Machines and Logistic Regression as described in [5], it can also be applied to deep learning models for classification.

With this in mind, we chose SGD as it had a static learning rate which we believe could perform better in scenarios where the batch sizes are bigger and the hidden layers are smaller as it seems to excel in quick computation. In Pytorch, we will be using SGD with Nesterov's momentum and momentum value of 0.9 to help it converge faster.

4.2.2 AdamW

AdamW is one of the most widely used optimization algorithms for training deep learning models. It builds on the ideas of earlier methods like AdaGrad and RMSProp by adjusting the learning rate for each parameter during training. This adjustment helps the model converge more quickly and efficiently. AdamW also includes a weight decay feature, which helps prevent overfitting by discouraging excessively large parameter values. Furthermore, since Adam uses first order gradients and dynamic learning rates, Adam converges faster than most deep learning optimizers [6].

With this in mind, we chose Adam as it represented the current state-of-the-art optimizer, and we believe that Adam could perform best in the "default" or baseline hyperparameter cases. We also believe it may also perform better in more complex models (more hidden layers) and smaller training cycles. In PyTorch, we will be using AdamW with the default beta values of (0.9, 0.999).

4.2.3 Adadelta

Adadelta improves gradient descent by using adaptive learning rates. It adjusts its step size dynamically based on recent updates, allowing the optimizer to balance cautious and aggressive steps depending on the situation. This flexibility can help the model learn more effectively without requiring extensive manual tuning. To do so, it keeps historical data of past gradients to determine the best learning rates as mentioned in [7]. This requires significantly more memory usage than SGD, but it also helps the weight to converge faster [7].

With this in mind, we chose Adadelta as it represented the middle ground between static learning rate optimizers and advanced dynamic optimizers. We believe that Adadelta could work best in cases where the hyperparameters differ

moderately from the baseline hyperparameters. In PyTorch, we will be using Adadelta with a weight decay value of 1e-2 to discourage overfitting.

4.3 Performance Metrics

To assess the models, we focused on three main metrics:

4.3.1 1. Training Loss:

Training loss measures how well the model fits the training data. It calculates the difference between the model's predictions and the actual labels. A lower training loss means the model is doing a better job at fitting the training data.

4.3.2 2. Validation Loss:

Validation loss is calculated on a separate dataset that the model hasn't seen before. It helps evaluate how well the model generalizes to new data. If validation loss is much higher than training loss, it may indicate overfitting, meaning the model is too specific to the training data and doesn't perform well on new data.

4.3.3 3. Weighted F1 Score:

The F1 score balances precision (how many predicted positives are correct) and recall (how many actual positives are correctly identified). The weighted F1 score gives more importance to classes that are less common, ensuring the model performs well across all classes, especially in imbalanced datasets.

5 Project Execution

5.1 Approaches and Methods

To evaluate the performance of different optimizers, we trained three distinct machine learning models: ZF-Net, a Convolutional Neural Network (CNN); Sequencer2d, an LSTM-based architecture; and ResNet-50, a Residual Neural Network with 50 layers. Each model was designed to process chest X-ray images resized to a standard input size of 224x224x3, with normalization ensuring consistency across the dataset.

ZF-Net utilized convolutional layers for feature extraction followed by fully connected layers for classification, while Sequencer2d replaced traditional self-attention blocks with bi-directional LSTM layers to capture temporal dependencies. ResNet-50 employed residual connections to address challenges in training deep networks. The models were trained using three optimizers—SGD, AdamW, and Adadelta—under various hyperparameter configurations, including adjustments to batch size, learning rate, and dropout rate. This experimental setup enabled a comprehensive assessment of optimizer performance across diverse scenarios.

5.2 Evaluation Criteria

The optimizers were compared based on their ability to minimize loss and maximize the weighted F1 score. Stability and consistency were also considered, as optimizers like SGD and AdamW performed reliably across different configurations, while Adadelta showed higher variance and struggled to converge in some cases.

Additionally, we analyzed the impact of hyperparameter variations on each optimizer's performance. For instance, we observed that SGD excelled with larger batch sizes due to its computational efficiency, while AdamW performed well under default settings, demonstrating its adaptability. The performance trends were visualized using loss curves and F1 score plots, offering a clear picture of each optimizer's behavior during training and validation.

This systematic approach allowed us to identify strengths and weaknesses of each optimizer, contributing to a deeper understanding of their applicability in fine-tuning models for medical image classification.

6 Results

6.1 ZF-Net

Here shown are the results of our testing using ZF-Net with their respective hyper-parameter and scheduler variations, trained with 50 epochs. We list each respective hyper-parameters' training and validation loss in figures 1-4 and the

Table 1: ZF-Net Hyperparameters

Hyperparameters	Batch Size	Learning Rate	Dropout rate
Baseline	32	1e-4	0.5
1st Variant	64	1e-2	0.5
2nd Variant	16	1e-4	0.25
Scheduler	32	1e-2	0.5

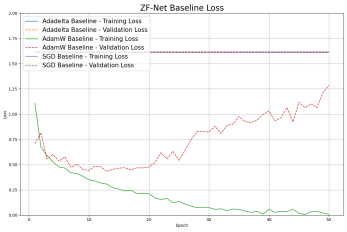

Figure 1: Baseline Loss

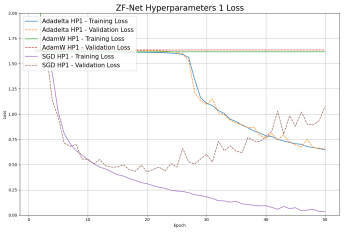

Figure 2: 1st Hyper-parameter Loss

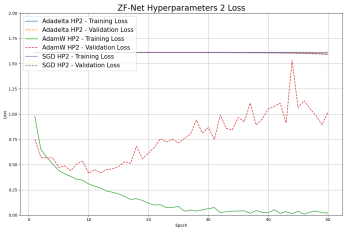

Figure 3: 2nd Hyper-parameter Loss

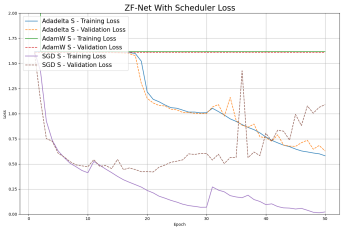

Figure 4: Scheduler Loss

Figure 5: F1-Score

overall F1-score in figure 5. We observe some interesting findings. AdamW performed the best in the baseline and second hyper-parameter variations in which the learning rate and batch size was relatively lower than other scenarios. From the graphs of the baseline and hyper-parameter 2, we see that only AdamW was able to reach a convergence, suggesting the dynamic nature of AdamW excels in finding a minima with smaller and frequent updates. More interestingly, in cases where AdamW does not perform well, the other two optimizers do, as pictured in the first hyper-parameter and scheduler scenarios. These hyper-parameters include a larger learning rate, which gives fascinating insight into the nature of SGD and Adadelta. Because these optimizers are not as dynamic as AdamW, we believe that these optimizers require a larger area to work with in order to make better decisions in the descent direction , especially in the case of Adadelta. In both graphs of the aforementioned scenarios, Adadelta is stuck at some local minima before being able to escape using the larger learning rate.

6.2 Sequencer-2d

Table 2: Sequencer-2d Hyperparameters

Hyperparameters	Batch Size	Learning Rate
Baseline	32	1e-4
1st Variant	64	1e-2
2nd Variant	16	1e-4
Scheduler	32	1e-2

Here shown are the results of our testing of Sequencer-2d, trained in 100 epochs. We list each respective hyper-parameters' training and validation loss in figures 6-9 and the overall F1-score in figure 10. Compared to ZF-net, Sequencer-2d takes longer to train, but is much more consistent. With this, the graphs also reveal some interesting findings. For all cases and optimizers, it appears that the model has not finished training yet as there is no sign of overfitting. This leads us to believe, we may be able to achieve an even lower loss if we let the model continue training

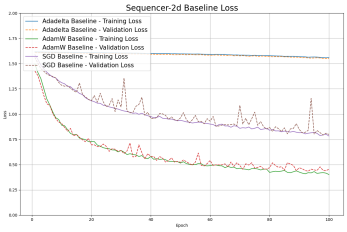

Figure 6: Baseline Loss

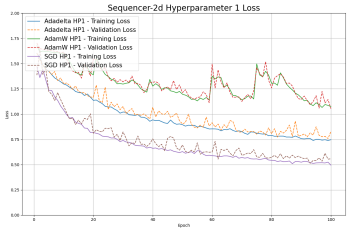

Figure 7: 1st Hyper-parameter Loss

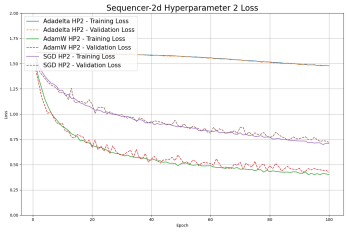

Figure 8: 2nd Hyper-parameter Loss

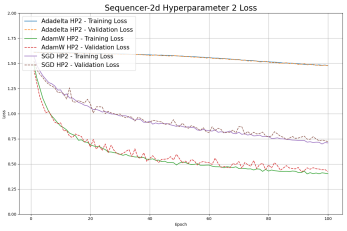

Figure 9: Scheduler Loss

Figure 10: F1-Score

longer. For the optimzers, AdamW follows the same pattern as ZF-net in which the optimizer does the best in the baseline and second hyper-parameter variation of the model. The pattern continues with Adadelta and SGD, where they performed best in the scenarios different than AdamW, contributing to our beliefs in the behavior patterns of these optimizers. Most surprisingly, SGD performed best overall in this model as it was able to approach some noticeable minima in all cases, supporting the aforementioned paper in related works that SGD is better in generalizing than most dynamic optimizers. This is interesting as Sequencer-2d, which follows a transformer like pathway, seems to support SGD better than a traditional vision based model like ZF-net. We predict that transformers are structured to support generalization rather than fast convergence, something that SGD excels over AdamW.

6.3 Resnet-50

Table 3: Sequencer-2d Hyperparameters

Hyperparameters	Batch Size	Learning Rate
Baseline	32	1e-4
1st Variant	64	1e-2
2nd Variant	16	1e-4
Scheduler	32	1e-2

Shown below is the results of Resnet-50. We list each respective hyper-parameters' training and validation loss in figures 11-14 and the overall F1-score in figure 15.Based on the graphs, there are not much information we can extract between the various optimizers due to its pre-trained nature, leading to overfitting in validation data despite reduced training loss. Our theory suggests that since Resnet-50 is already pre-trained, this is actively affecting the training of our data from the very beginning, which may be the reason why the training has been unpredictable. One exception to this is Adadelta in the baseline and hyper-parameter 2 variation scenarios. This breaks away from the trend established so far in this project as Adadelta usually does better under scheduler and hyper-parameter 1 conditions. We predict that the pretrained nature of our Resnet-50 model allows an optimizer like Adadelta, who usually struggled in the past training, to make more accurate steps.

7 Individual Contribution

In this subsection, we detail the individual contributions of each team member. This includes the specific tasks they were responsible for, their methodological decisions, and the insights they brought to the project. By delineating these

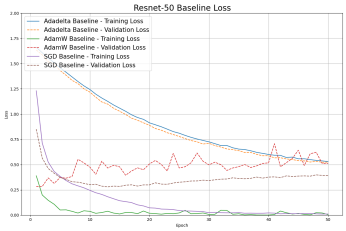

Figure 11: Baseline Loss

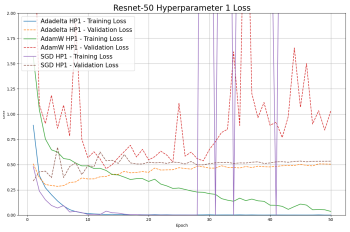

Figure 12: 1st Hyper-parameter Loss

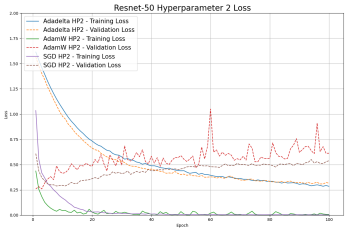

Figure 13: 2nd Hyper-parameter Loss

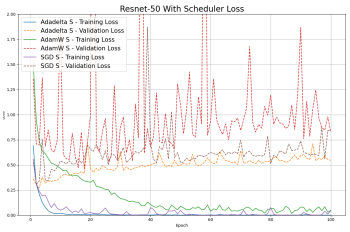

Figure 14: Scheduler Loss

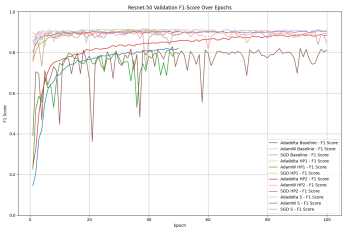

Figure 15: F1-Score

contributions, we aim to provide clarity on how each member's efforts supported the overall goals and outcomes of this work.

7.1 Ray Hu

- **Exploring ZF-Net and Sequencer-2d Models:**
  I explored various CNN and LSTM based models by reading several research papers and decided upon using ZF-Net and Sequencer-2d for this project. From there, I did various research and decided on key hyper-parameters to use for these models. From there, I implemented ZF-Net model and imported the Sequencer-2d model from the Hugging face library.
- **Template Training File Creation:**
  I created our group's main training file and its scheduler variant file which was used for training and recording our data. This template file was used across the ZF-Net, Sequencer-2d, and ResNet models and was also SOL compatible. Hyper-parameters and optimizers in the file were easily accessible for the purpose of this project.
- **Model Training/Experiments:**
  I led most of the training on the models on the SOL supercomputer as I had the most experience using the platform. I uploaded the necessary training files and dataset onto the supercomputer and trained many of the ZF-Net and Sequencer-2d models. After getting results, I assisted in giving analysis of the data of each model.

7.2 Jiandong Guan

- **Exploring the RESNET Model:**
  I explored the RESNET model to analyze how different optimizers and hyperparameters affect its performance. The objective was to identify configurations that could enhance the model's accuracy and reduce loss. To achieve this, I implemented the RESNET model and conducted a series of experiments using various optimizers, including SGD, Adam, and adadelta. I also adjusted key hyperparameters such as the learning rate, batch size, and the number of layers.
- **Assisting Experiments on CNN and RNN Models**
  I assisted in running experiments on CNN and RNN models using the SOL supercomputer to test the effects of different hyperparameters and optimizers. This involved configuring the environment on the SOL system, ensuring all required software and libraries were properly installed.By leveraging the SOL supercomputer's computational power, I facilitated large-scale testing and analysis of the models. This approach ensured faster execution and allowed us to draw meaningful comparisons between different setups, significantly contributing to the project's experimentation and optimization efforts.
- **Project Documentation and Presentation Preparation**
  For the project documentation and presentation, I focused on ensuring that the key contributions were clearly summarized and presented in a structured format. The primary goal was to document my work in a

comprehensive and organized way, while also preparing a professional presentation to effectively convey the essential points.

7.3 Cong Wei

- **Data visualization and evaluation:**
  I took the lead on comprehensive data visualization efforts, translating raw data and model outputs into accessible graphical representations. By developing clear, informative plots and charts, you enabled the team to quickly interpret complex patterns, detect trends, and identify areas needing further refinement. In addition, you conducted thorough model evaluations, leveraging both quantitative and qualitative metrics to assess model performance. This work ensured that the team could effectively measure improvements, select optimal hyperparameters, and confidently recommend best practices for future model training and deployment.
- **Result writing:**
  I took the responsibility for interpreting the experimental findings and translating them into a coherent results section. This involved examining key metrics such as training loss, validation loss, and F1 scores across various optimizers and hyperparameter configurations, then identifying clear patterns and trends. By synthesizing this information, I provided actionable insights and detailed recommendations, ultimately framing the narrative of how each optimizer performed and guiding future optimization strategies.

7.4 Suhas Raghavendra

- **Final Report Crafting:**
  Played a key role in drafting the final report, including compiling and structuring project details, organizing content, summarizing findings concisely, and ensuring adherence to project guidelines. Additionally, integrated team feedback and refined the report for final submission
- **Assisted in Training of Machine Learning Models on the SOL Supercomputer:**
  Supported the training of machine learning models on the SOL supercomputer by configuring the computational environment, managing data transfers, debugging training processes, and optimizing resource utilization for large datasets.
- **Presentation Creation:**
  Contributed to the refinement of the project presentation by ensuring consistent formatting and style, selecting impactful visuals, and preparing concise and accessible slides. Collaborated with the team to emphasize clarity and key takeaways.
- **Conducted Literature Review:**
  Conducted a comprehensive review of academic and industry research on optimizers, image classification models, and related topics. This analysis provided critical insights into optimizer performance and informed the direction of the project.

8 Conclusion

This study highlights the pivotal role of optimizer selection and hyperparameter tuning in fine-tuning machine learning models for lung disease classification. By experimenting with three optimizers—SGD, AdamW, and Adadelta—on a dataset of augmented chest X-ray images, we observed distinct performance patterns across various configurations. AdamW consistently demonstrated strong baseline performance, achieving the best results with moderate batch sizes and lower learning rates while SGD and Adadelta performed best in larger learning rates.

The evaluation of training loss, validation loss, and F1 scores provided key insights into the behavior of each optimizer. AdamW's ability to minimize both training and validation losses translated into high F1 scores, making it an effective choice for baseline setups. SGD, while effective with larger batch sizes, also showed strong generalization but required more careful attention to training parameters. Adadelta's inability to converge effectively under different conditions suggests it may not be suitable for image classification tasks involving medical datasets.

Our findings highlight the importance of early stopping to prevent overfitting, especially with AdamW and SGD. Monitoring validation loss and F1 scores helped identify optimal epochs, ensuring efficient training without extra computational costs.

These findings have significant implications for the development of automated diagnostic tools in healthcare. Accurate and efficient models trained with the right configurations can enable faster and more reliable disease detection, reducing diagnostic errors and supporting overburdened medical professionals. This is particularly vital in resource-limited settings where access to expert radiologists is scarce. By optimizing these models, hospitals and clinics can potentially implement scalable solutions to address global health challenges.

References

[1] Ashia Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *arXiv:1705.08292*, 2017.

[2] Matthew Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *arXiv:1311.2901*, 2013.

[3] Yuki Tatsunami and Masato Taki. Sequencer: Deep lstm for image classification. *arXic:2205.01972*, 2022.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv:1512.03385*, 2015.

[5] Stochastic gradient descent. `https://scikit-learn.org/stable/modules/sgd.html#stochastic-gradient-descent`. Accessed: 2024-10-25.

[6] Diederik Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[7] Matthew Zeiler. Adadelta: An adaptive learning rate method. *arXiv:1212.5701*, 2012.