Evaluation of Optimizers on KMNIST Dataset

1. Introduction

Deep learning models have demonstrated significant success across various applications, including image recognition, speech processing, and natural language understanding. The performance of these models heavily depends on the choice of optimization algorithms used for training. In this report, we evaluate and compare the performance of three widely used optimizers—Adam, RMSprop, and AdamW—on a machine learning task involving the KMNIST dataset.

KMNIST Dataset

The Kuzushiji-MNIST (KMNIST) dataset is a collection of 28x28 grayscale images representing handwritten characters from the Japanese Hiragana alphabet. It consists of 60,000 training images and 10,000 test images, each belonging to one of 10 classes. Similar to the KMNIST dataset, it serves as a useful benchmark for image classification tasks, though it introduces additional complexity due to variations in handwriting styles.

Objective

This study investigates how different optimizers affect the training performance of a simple feedforward neural network (FNN) on the KMNIST dataset. Specifically, we compare the final test accuracy, training time, and convergence behavior of the following optimizers:

1. Adam
2. RMSprop
3. AdamW

By analyzing the results, we aim to determine which optimizer performs best for this particular task.

2. Optimizers Description

2.1 Adam (Adaptive Moment Estimation)

Adam is one of the most widely used optimization algorithms in deep learning. It combines the benefits of two other extensions of stochastic gradient descent (SGD): **Momentum** and **RMSprop**. Specifically, Adam computes adaptive learning rates for each parameter from estimates of first and second moments of the gradients (mean and variance, respectively).

The key features of **Adam** include:

1. **Momentum:** Like classical momentum, Adam accumulates a moving average of past gradients, which helps to smooth the updates and navigate through ravines (areas with steep gradients in one dimension and flat gradients in another).
2. **Adaptive Learning Rate:** Adam adjusts the learning rate for each parameter individually, based on the estimated variance of the gradients. This allows for efficient updates, especially for parameters that have sparse gradients or different scales.
3. **Bias Correction:** Adam also includes a bias-correction mechanism to compensate for the initial values of the first and second moment estimates, which may be biased toward zero.

Advantages of Adam:

1. **Efficient computation**: It is computationally efficient, requiring only a few extra parameters (moment estimates and squared gradients) compared to standard SGD.
2. **Adaptivity**: By adjusting the learning rate dynamically, Adam is able to adapt to the characteristics of the loss surface, which often results in faster convergence.

3. **Good for sparse gradients**: Adam is particularly effective when gradients are sparse or noisy, making it suitable for complex, high-dimensional problems like image recognition.

2.2 RMSprop (Root Mean Square Propagation)

RMSprop is an adaptive learning rate optimization algorithm. It addresses some of the limitations of standard stochastic gradient descent (SGD) by dividing the learning rate for each parameter by a running average of the magnitude of recent gradients for that parameter. This helps to stabilize the updates and prevent overly large steps in areas where the gradients are large.

Key features of **RMSprop**:

1. **Moving Average of Gradients:** The algorithm computes a moving average of squared gradients over a specified window of past iterations. This helps to control the magnitude of updates and improves convergence on non-stationary objectives.
2. **No Momentum:** Unlike Adam, RMSprop does not use momentum, which can sometimes make it less effective when dealing with highly oscillatory loss surfaces.

Advantages of RMSprop:

1. **Improved convergence for non-stationary problems**: RMSprop performs well when training on online or non-stationary objectives, making it suitable for tasks like reinforcement learning or non-convex optimization problems.
2. **Less tuning required**: RMSprop typically requires fewer hyperparameter adjustments than SGD.

2.3 AdamW (Adam with Weight Decay)

AdamW is a variant of the Adam optimizer that incorporates **weight decay regularization**. Regularization techniques, such as L2 regularization, are commonly used to prevent overfitting by penalizing large weight values. In AdamW, weight decay is decoupled from the optimization step, which helps maintain the adaptive learning rates used by Adam while still applying regularization.

Key features of **AdamW**:

1. **Weight Decay:** It applies the L2 regularization term (weight decay) separately from the gradient update. This ensures that the regularization is applied after computing the gradients, preventing the learning rate from being directly influenced by the weight decay.
2. **Improved Generalization:** Weight decay regularization helps prevent overfitting, especially when dealing with small datasets or complex models.

Advantages of AdamW:

1. **Better generalization**: By decoupling weight decay, AdamW leads to better generalization and performance on out-of-sample data compared to Adam.
2. **Handles overfitting**: The regularization helps mitigate overfitting issues, which can be a common problem in deep learning models with many parameters.
3. **Efficient and robust**: AdamW retains the efficiency and robustness of Adam while adding weight decay, making it a good choice for more complex tasks.

3. Experimental Setup

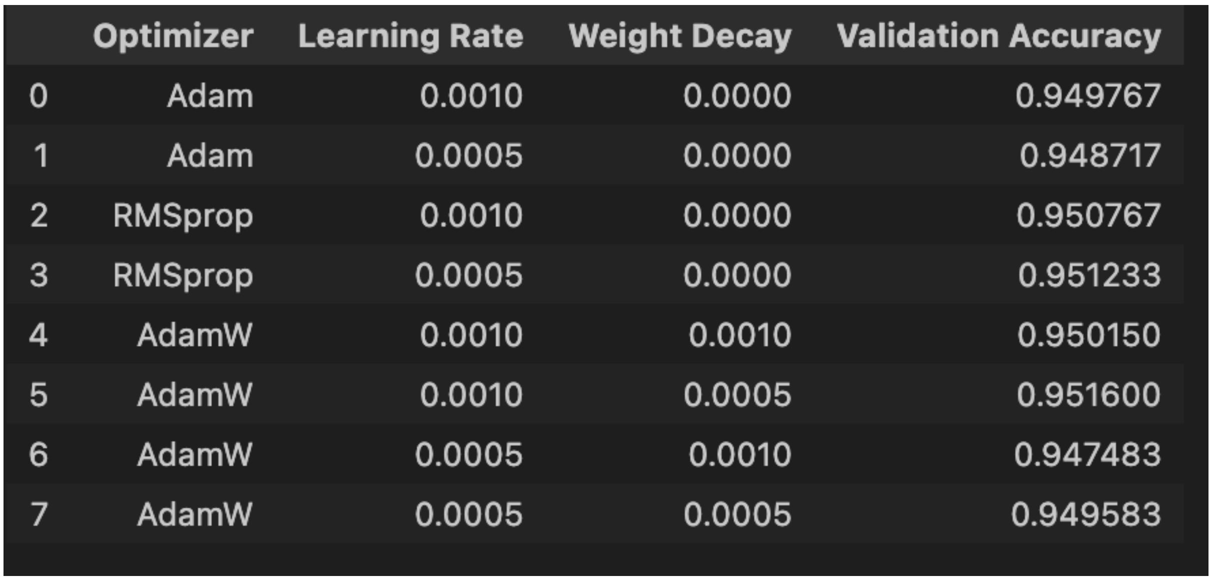
Model and Training Configuration

1. **Neural Network Architecture:** Simple Feedforward Neural Network (FNN)
2. **Dataset:** KMNIST (60,000 training images, 10,000 test images)
3. **Batch Size:** 32
4. **Epochs:** 10
5. **Hyperparameters Tuned:** Learning Rate and Weight Decay

4. Results and Finding

Table: Validation Accuracy for Different Optimizers

The table below shows the validation accuracy across different learning rate ,weight decay and optimizers



	Optimizer	Learning Rate	Weight Decay	Validation Accuracy
0	Adam	0.0010	0.0000	0.949767
1	Adam	0.0005	0.0000	0.948717
2	RMSprop	0.0010	0.0000	0.950767
3	RMSprop	0.0005	0.0000	0.951233
4	AdamW	0.0010	0.0010	0.950150
5	AdamW	0.0010	0.0005	0.951600
6	AdamW	0.0005	0.0010	0.947483
7	AdamW	0.0005	0.0005	0.949583

Fig 4.1. Validation Accuracy for different optimizers

2. Heat map below explains the validation accuracy against all the optimizers where AdamW performs the best against Adam and RMSprop

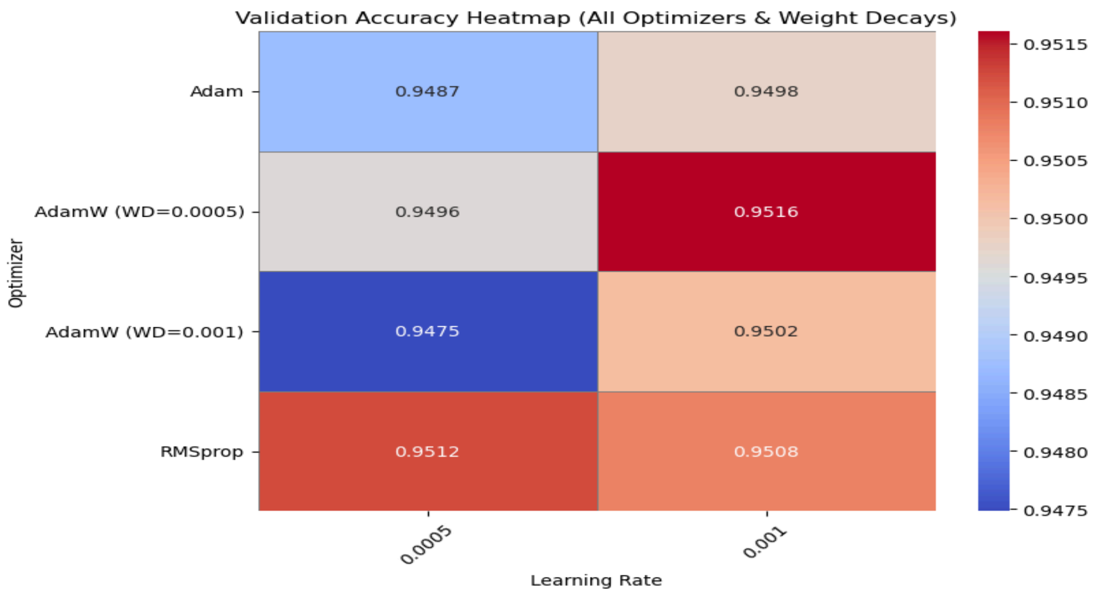


Fig 4.2. Heatmap

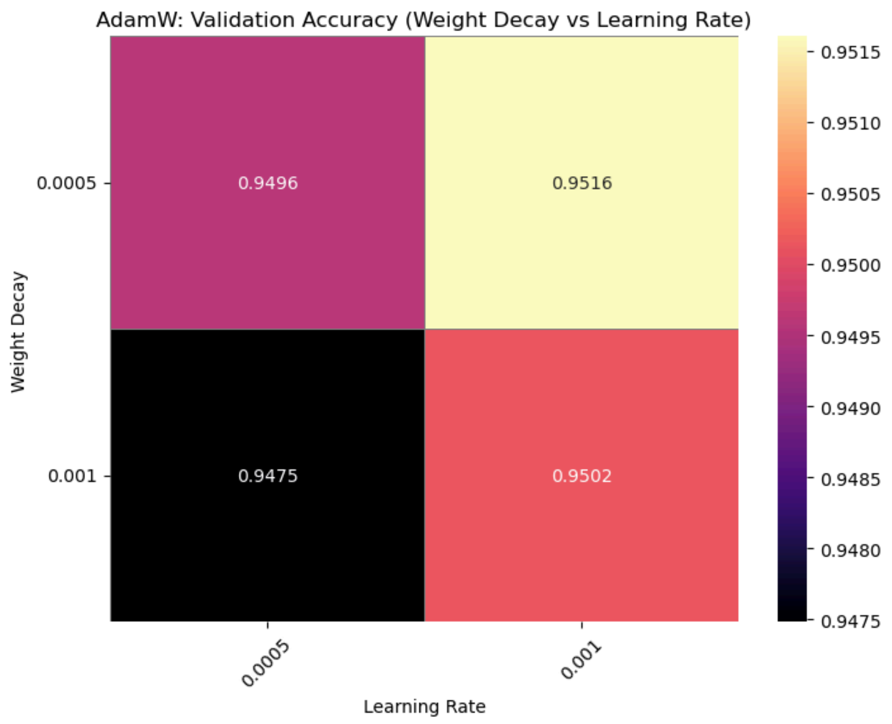


Fig 4.3. Weight Decay vs Learning Rate

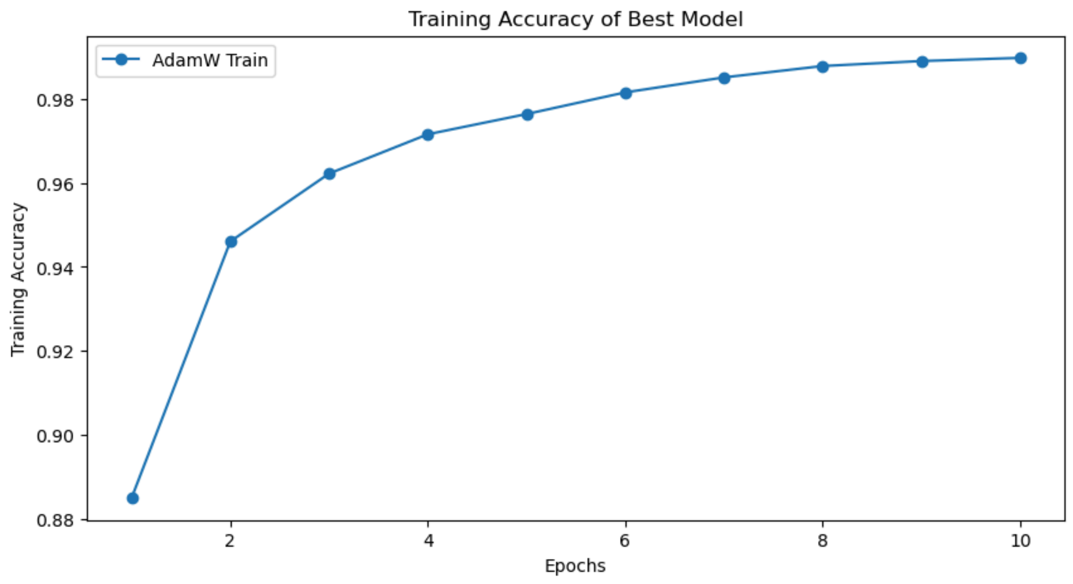


Fig 4.4. AdamW training accuracy(best model)

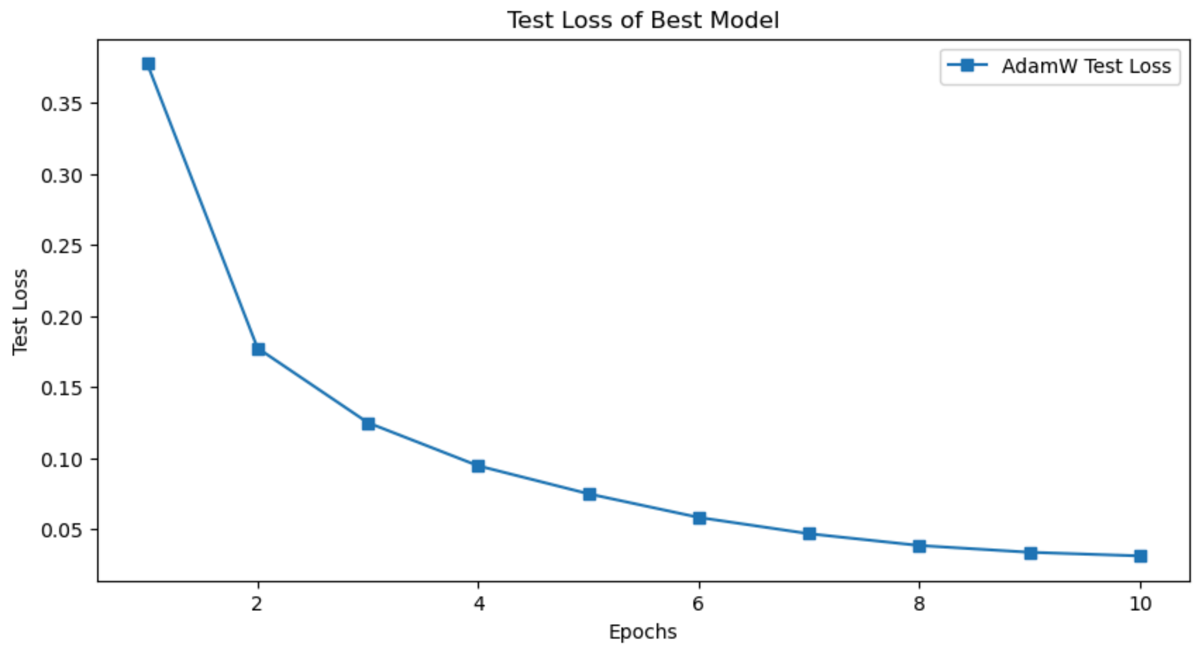


Fig 4.5. AdamW test loss (best model)

Key Observations:

1. **AdamW Outperforms RMSprop**: AdamW with a learning rate of 0.001 and weight decay of 0.0005 achieved the highest validation accuracy (0.9516), showing its effectiveness in this task.
2. **RMSprop is Competitive**: RMSprop performed nearly as well as AdamW, with a validation accuracy of 0.9512, despite not requiring weight decay. This suggests RMSprop is a strong alternative, especially when weight decay is not used.
3. **Effect of Weight Decay**: Weight decay plays an important role in preventing overfitting, but excessive decay negatively impacts performance, as seen with AdamW when adjusted.

4. **Learning Rate Sensitivity**: Both optimizers show sensitivity to the learning rate, with AdamW performing best at 0.001 and RMSprop at 0.0005, highlighting the need for fine-tuning hyperparameters for optimal performance.

5. Interpretation and Discussion

Regularization and Generalization

AdamW’s decoupled weight decay leads to better generalization than Adam, which incorporates weight decay directly into gradient updates. AdamW achieved a higher validation accuracy highlighting its superior regularization.

Overfitting and Stability

Adam, while fast, tends to overfit smaller datasets like KMNIST. RMSprop lacks momentum and regularization, leading to suboptimal minima. AdamW’s weight decay mitigates overfitting and ensures stable convergence, making it ideal for KMNIST.

Convergence Behavior

- Adam: Converges quickly but may sacrifice stability.
- RMSprop: Struggles with oscillations in complex tasks.
- AdamW: Balances fast convergence with stability, leading to smoother learning and better final accuracy

.

KMNIST Complexity

Due to high intra-class variation, KMNIST requires strong generalization. AdamW's regularization prevents overfitting, helping the model learn general patterns rather than memorizing specific instances.

6. Conclusion

1. AdamW with a learning rate of 0.001 and weight decay of 0.0005 is the best choice for this task, achieving the highest validation accuracy (0.9516).
2. RMSprop is also a strong alternative, performing nearly as well as AdamW without requiring weight decay.
3. Weight decay plays a crucial role in preventing overfitting, but excessive decay negatively impacts performance.

7. References

- D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–15.
- G. Hinton, N. Srivastava, and K. Swersky, “Lecture 6a: RMSprop,” *Neural Networks for Machine Learning*, 2012.
- I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," *ICLR 2019*.

Names	Contribution
1. Antra Nayudu	1. Implemented and analyzed the **Adam** optimizer. 2. Wrote detailed documentation on methodology and experimental setup.
2. Kartik Kalra	1. Implemented and analyzed the **RMSprop** optimizer and visualization for the comparison of all optimizers. 2. Created the README file for the GitHub repository and set up the GitHub repository.
Pushkar Kafley	1. Implemented and analyzed the **AdamW** optimizer. 2. Designed and prepared the PowerPoint presentation.

Each team member contributed equally to research, discussions, documentation, and presentation to ensure a well-rounded project.