**GROUP: BitBrains**

**TITLE: Analysis of State-of-the-Art Optimizer Algorithms on Fashion MNIST Dataset**

**GROUP MEMBERS:**

**Noah Nsimbe**

**AmakaGenevieve Jane Awa**

**Oluwadamilare Matthew Kolawole**

**Ryhan Sunny**

[Introduction:](#_cwrr36d4uosz)

[Dataset Description:](#_826v1c682qhy)

[Deep Learning Model:](#_9ip8p4kovdth)

[Optimizer Algorithms:](#_9o2dw3ebfoo6)

[Adam optimizer](#_v8oqfcf62cvz)

[Details of the Adam Optimizer](#_q5qhfo31whf)

[Pros of Adam Optimizer](#_hbq928vss8uh)

[Cons of Adam Optimizer](#_lnwdhuei5vgk)

[Application to Fashion MNIST Dataset:](#_defj1qlc9ur1)

[Analysis of the results:](#_m1fr69rgei)

[RMSprop optimizer](#_1ibuqr2xocw)

[Details of RMSprop Algorithm](#_hgik1krp5rao)

[Summary of how RMSprop works:](#_94qhqtv582t2)

[Pros of RMSprop optimizer](#_w7nnlcryesyx)

[Cons of RMSprop optimizer](#_z77zgt6b89m4)

[Adagrad optimizer](#_7n8vvfh4m1xa)

[Details of Adagrad Algorithm](#_pzaaw49p5lmn)

[Pros of Adagrad optimizer](#_1fr4icf7jnrm)

[Cons of Adagrad optimizer](#_2jj6m9lwg1fv)

[Results:](#_7l0dfq64nl1b)

[Interpretations and Discussions](#_qdh21rzenbqo)

[RMSprop Optimizer](#_us6e56c1k1w)

[Adam Optimizer](#_wx81uywska7w)

[Adagrad Optimizer](#_3ues361yczqt)

[Conclusions](#_7zsn1kx9z6ks)

# **Introduction:**

In this report, we investigated three state-of-the-art optimizer algorithms: Adam, RMSprop, and Adagrad, applied to an artificial neural network (ANN) trained on the Fashion MNIST dataset. Our objective is to compare the performance, advantages, and disadvantages of each optimizer in terms of training efficiency, accuracy, and loss minimization.

# **Dataset Description:**

The Fashion MNIST dataset consists of 60,000 training images and 10,000 test images, each being a 28x28 grayscale image, associated with a label from 10 classes representing different fashion items. It is widely used for benchmarking machine learning models in the field of computer vision.

# **Deep Learning Model:**

We employ a simple ANN architecture with multiple layers, including input, hidden, and output layers, to classify the fashion items in the Fashion MNIST dataset. The model's performance is evaluated based on its accuracy and loss metrics.

# **Optimizer Algorithms:**

## **Adam optimizer**

The Adam optimizer is a stochastic gradient descent algorithm used in training artificial neural networks (ANNs). It combines elements of two other popular optimization techniques, AdaGrad and RMSProp:

It is based on adaptive estimation of first-order and second-order moments. First, it looks at how steep the slope is when we are trying to find the best way to decrease the error (that's the first-order moment). Then, it also considers how quickly the slope changes in different directions (that's the second-order moment).

Adam adjusts their learning process by keeping track of these moments as it goes along. This helps it figure out how much to change the learning rates for different parts of the neural network. By doing this, Adam can learn more effectively and get better results, especially when the data is noisy or the learning process is complicated.

### **Details of the Adam Optimizer**

* **Adaptive Learning Rate:** Adam adjusts the learning rate for each parameter individually, based on the first and second moments of the gradients.
* **Momentum:** It utilizes momentum to accelerate the optimization process by accumulating exponentially decaying average of past gradients.
* **Bias Correction:** Adam applies bias correction to the estimates of the first and second moments to alleviate their initialization bias towards zero, especially in the initial training stages.
* **Parameter Updates:** The parameters are updated based on the calculated moving averages of the gradients.

### **Pros of Adam Optimizer**

* **Efficient:** Adam is known for its efficiency in training deep neural networks, often converging faster than other optimization algorithms.
* **Adaptive Learning Rates:** Its adaptive learning rate mechanism allows for dynamic adjustments to the learning rates for each parameter, which can be beneficial in different stages of training.
* **Suitability for Sparse Data:** Adam performs well with sparse gradients and noisy data, making it suitable for a wide range of applications.

### **Cons of Adam Optimizer**

* **Memory Usage:** Adam maintains additional state variables for each parameter, which can increase memory consumption, especially for large models.
* **Hyperparameter Sensitivity:** Adam requires careful tuning of hyperparameters such as learning rate, beta1, beta2, and epsilon, to achieve optimal performance. Improper tuning may lead to suboptimal results.
* **Robustness to Noise:** In some cases, Adam may exhibit sensitivity to noise in the objective function, causing fluctuations in convergence behavior.

### **References**

<https://keras.io/api/optimizers/adam/>

<https://www.geeksforgeeks.org/adam-optimizer-in-tensorflow/>

<https://www.geeksforgeeks.org/intuition-of-adam-optimizer/>

<https://spotintelligence.com/2023/03/01/adam-optimizer/>

## 

## **RMSprop optimizer**

RMSprop (Root Mean Square Propagation) is an optimization algorithm commonly used in training artificial neural networks (ANNs). It is particularly effective in scenarios where other optimization algorithms like vanilla stochastic gradient descent (SGD) may struggle due to problems such as vanishing or exploding gradients.

### **Details of RMSprop Algorithm**

RMSprop is an adaptive learning rate optimization algorithm proposed by Geoffrey Hinton in his course on Neural Networks for Machine Learning. The algorithm is designed to adaptively adjust the learning rates for different parameters during training.

### **Summary of how RMSprop works:**

* **Compute Squared Gradients:** RMSprop maintains a moving average of the squared gradients for each parameter. This is similar to AdaGrad but with a decaying average.
* **Update Parameters:** The update rule adjusts the learning rate for each parameter based on the average of the squared gradients.
* **Adaptive Learning Rates:** RMSprop divides the learning rate by the square root of the exponentially decaying average of squared gradients for each parameter. This helps to normalize the learning rates and overcome the problems of vanishing or exploding gradients.

### **Pros of RMSprop optimizer**

* **Adaptive Learning Rates:** RMSprop adapts the learning rates for each parameter individually based on the magnitude of their gradients. This helps converge faster and more efficiently, especially in deep neural networks.
* **Stability:** It helps to stabilize the learning process by mitigating the issues of vanishing and exploding gradients.
* **Simple Implementation:** RMSprop is relatively easy to implement and widely used in practice.

### **Cons of RMSprop optimizer**

* **Hyperparameter Sensitivity:** RMSprop, like other adaptive methods, has hyperparameters that need to be tuned, such as the learning rate and the decay rate. Improper tuning can lead to suboptimal performance.
* **Memory Usage:** RMSprop maintains a moving average of squared gradients for each parameter, which can require additional memory, especially for large models with many parameters.

### **References**

<https://keras.io/api/optimizers/rmsprop/>

## **Adagrad optimizer**

Adagrad (Adaptive Gradient Algorithm) is an optimization algorithm commonly used in training artificial neural networks (ANNs). It dynamically adjusts the learning rates of each parameter based on the historical gradients.

### **Details of Adagrad Algorithm**

Adagrad is designed to adaptively adjust the learning rates for each parameter during training. It achieves this by scaling the learning rates based on the historical gradients of each parameter. Here's how Adagrad works:

* **Compute Squared Gradients:** Adagrad maintains a sum of the squared gradients for each parameter.
* **Adapt Learning Rates:** It divides the learning rate by the square root of the sum of squared gradients for each parameter. This effectively reduces the learning rate for parameters that have large gradients and increases it for parameters that have small gradients.
* **Accumulation of Gradients:** Adagrad accumulates the squared gradients over time, so the learning rates decrease monotonically during training.

### **Pros of Adagrad optimizer**

* **Adaptive Learning Rates:** Adagrad adapts the learning rates for each parameter individually based on the historical gradients. This can help converge faster and more efficiently, especially for sparse data or when dealing with features that occur infrequently.
* **No Manual Tuning of Learning Rate:** Adagrad automatically adjusts the learning rates based on the gradients, reducing the need for manual tuning of learning rate hyperparameters.

### **Cons of Adagrad optimizer**

* **Decreasing Learning Rates:** Adagrad's accumulation of squared gradients can lead to learning rates that decrease too aggressively over time. This can cause the learning process to slow down prematurely, especially for deep neural networks.
* **Memory Usage:** Adagrad needs to store and update the sum of squared gradients for each parameter, which can lead to increased memory usage, especially for models with many parameters.
* **RMSprop and AdaDelta:** While Adagrad was one of the early adaptive learning rate algorithms, more recent algorithms like RMSprop and AdaDelta have been developed to address some of its shortcomings, such as the aggressive decrease in learning rates.

# **Application to Fashion MNIST Dataset:**

1. **Loss:** The "loss" measures how well the neural network is performing. It represents the difference between the predicted output and the actual output. Lower loss values indicate better performance.
2. **Accuracy:** The "accuracy" measures how often the neural network makes correct predictions. Higher accuracy values indicate better performance.

## **Analysis of the results:**

* **Epochs:** The training process was repeated 30 times (epochs) to improve the neural network's performance.
* **Training and Validation:** The neural network was trained on a training set and evaluated on a validation set after each epoch. This helped to monitor how well the model generalizes to new data.
* **Improvement:** Over the epochs, both training and validation loss decrease, and accuracy increases. This indicates that the model is learning from the data and improving its performance.
* **Performance:** The neural network achieved an accuracy of around 87% on the validation set by the end of training. This means that it correctly classified the images in the Fashion MNIST dataset about 87% of the time.
* **Stability:** The loss and accuracy values on the validation set appear to stabilize towards the end of training, suggesting that the model's performance isn't changing significantly after a certain number of epochs.

# **Results:**

The performance of the three optimizers is summarized in the following table:

| **Optimizer** | **Training Accuracy** | **Validation Accuracy** | **Loss** |
| --- | --- | --- | --- |
| **RMSprop** | 97.04% | 88.93% | 0.673 |
| **Adam** | 88.75% | 84.78% | 0.554 |
| **Adagrad** | 85.60% | 84.08% | 0.457 |

# **Interpretations and Discussions**

## **RMSprop Optimizer**

This optimizer achieved the highest training accuracy among the three, indicating that it might have converged faster or found better local minima during training.

The validation accuracy is slightly lower than the training accuracy, suggesting some degree of overfitting, though the gap between training and validation accuracies is not too large.

The loss value is relatively high compared to the other optimizers, indicating that the model might still have room for improvement in terms of minimizing the loss function.

## **Adam Optimizer**

The training accuracy is lower than RMSprop but still reasonably high.

The validation accuracy is also lower than RMSprop, indicating that Adam might not have converged as well as RMSprop on this specific dataset.

The loss value is lower than RMSprop, suggesting that Adam might have found a better optimization path compared to RMSprop.

## **Adagrad Optimizer**

Both the training and validation accuracies are lower compared to RMSprop and Adam.

The loss value is the lowest among the three optimizers, indicating that Adagrad might have found the optimal parameters more efficiently in terms of the loss function.

However, the accuracy metrics are relatively lower, indicating that it might not generalize as well as the other optimizers.

# **Conclusions**

RMSprop seems to strike a balance between training accuracy and generalization (validation accuracy). It has a high training accuracy but slightly lower validation accuracy, indicating some level of overfitting.

Adam performs reasonably well but might not generalize as effectively as RMSprop. It converges faster than RMSprop in terms of loss minimization but lags slightly in validation accuracy.

Adagrad achieves the lowest loss value but does not perform as well in terms of accuracy metrics, indicating potential issues with generalization.