International Journal of Computing and Digital Systems
2025, VOL. 17, NO. 1, 1–13

http://dx.doi.org/10.12785/ijcds/1571023677

Variance Adaptive Optimization for the Deep Learning Applications

Nagesh Jadhav[1], Rekha Sugandhi[2], Rajendra Pawar[3], Swati Shirke[4] and Jagannath Nalavade[5]

[1,3,5]*Department of Computer Science and Engineering, MIT Art, Design and Technology University, Pune, India*
[2]*Department of Information Technology,MIT Art, Design and Technology University, Pune, India*
[4]*SOET,Pimpri Chinchwad University, City, Country*

Received 23 April 2024, Revised 3 October 2024, Accepted 25 October 2024

Abstract: Artificial intelligence jargon encompasses deep learning that learns by training a deep neural network. Optimization is an iterative process of improving the overall performance of a deep neural network by lowering the loss or error in the network. However, optimizing deep neural networks is a non-trivial and time-consuming task. Deep learning has been utilized in many applications ranging from object detection, computer vision, and image classification to natural language processing. Hence, carefully optimizing deep neural networks becomes an essential part of application development. In the literature, many optimization algorithms like stochastic gradient descent, adaptive moment estimation, adaptive gradients, root mean square propagation etc. have been employed to optimize deep neural networks. However, optimal convergence and generalization on unseen data is an issue for most of the conventional approaches. In this paper, we have proposed a variance adaptive optimization (VAdam) technique based on Adaptive moment estimation (ADAM) optimizer to enhance convergence and generalization during deep learning. We have utilized gradient variance as useful insight to adaptively change the learning rate resulting in improved convergence time and generalization accuracy. The experimentation performed on various datasets demonstrates the effectiveness of the proposed optimizer in terms of convergence and generalization compared to existing optimizers.

Keywords: Deep Neural Networks, Optimization, Variance, Deep Learning, Convergence

1. INTRODUCTION

Deep learning has developed as an effective method for handling complicated problems in a variety of disciplines, including computer vision and natural language processing. However, optimizing deep learning models strongly relies on effective optimization strategies. Deep learning model convergence and generalization are essential elements that have a direct influence on their performance and applicability in real-world circumstances. Convergence relates to a model's capacity to find an optimal solution during the training phase, whereas generalization refers to the model's ability to function effectively on previously unknown data. In deep learning, optimization techniques including Stochastic Gradient Descent (SGD), Adagrad, Adadelta, RMSProp and its derivatives have been frequently utilized [1]. DNN architecture consist of the input layer, hidden layers, and output layers. Deep neural networks are sequential feed-forward network, which processes an input and provide it to hidden layers for further processing. The hidden layer stores the necessary information and passes it to the output layer for processing. The output layer supports different types of distributions in the output unit. Primarily it uses Gaussian distribution for linear output units, binomial distribution for binary classification problems and multinouli distribution for multiclass classification [2]. The hidden unit utilizes activation functions like sigmoid, tanh, softplus and ReLu based on the input and desired output. With the ability to learn complex patterns and representations from data, type of machine learning model is called a deep neural network (DNN), made up of numerous layers of interconnected nodes. To handle specific challenges presented by language data, DNN designs in NLP have undergone substantial evolution, adding specialized layers and methods. For example, recurrent neural networks (RNNs) are well-suited for tasks like sentiment analysis, machine translation, and language modelling because of their recurrent connections, which enable them to analyze sequences of inputs. To capture long-range dependencies in sequences, classical RNNs are limited by vanishing and exploding gradient problems. Deep architectures like Gated Recurrent Units (GRUs) and Long Short-Term Memory Networks (LSTMs) have been created to overcome this

E-mail address: nagesh10@gmail.com, rekha.sugandhi@gmail.com, rgpawar13@gmail.com

constraint of vanishing and exploding gradients. Furthermore, transformer-based models, first described by Vaswani et al. in 2017 [3], have become the prevailing paradigm in NLP. Transformers use self-attention methods to capture global dependencies in input sequences, allowing them to mimic long-term interactions more effectively than typical recurrent architectures. Models such as BERT (Bidirectional Encoder Representations from Transformers) [4] and GPT (Generative Pre-trained Transformer) [5] have demonstrated cutting-edge performance across a wide range of NLP tasks, including question answering, language comprehension, and text generation. The primary objective of deep neural networks is to diminish the difference between likely results and expected results. This is the process of optimization where $\theta(w, b)$ parameters are tuned to reduce the loss or error that occurs. The optimization algorithm plays a significant role in forward and backward propagation during the neural network training process. An optimization algorithm helps in driving the solution to the global optimum. The most popular optimization algorithm is gradient descent. It comes in three flavours: vanilla or batch gradient descent, stochastic gradient descent, and mini-batch gradient descent. While these approaches have achieved amazing success, they frequently encounter difficulties in terms of convergence speed and generalization performance. Deep learning models are complicated and high-dimensional, making it difficult to strike the correct balance between convergence and generalization. When it comes to non-convex optimization, gradient-based approaches struggle to converge. The best approach to overcome the non-convex optimization problem is to improvise gradient-based learning using a momentum-based approach and learning rate adaption [6].

In the deep learning process learning is an important parameter to set. Choosing the ideal learning rate is the most critical aspect of training deep neural networks as it affects overall network performance. Setting small learning leads to slower or delayed convergence and configuring a large learning rate may result in overshooting the global optimum. Therefore, finding the best learning rate is a process of finding a tradeoff between small and large values. Algorithms like adaptive gradient (Adagrad), Adadelt, RMSProp, Adaptive moment estimation (ADAM), and Nesterov ADAM (NADAM) provide a platform for adaptive change learning rate based on the model's performance [7]. However, most of above mention optimization convergence poses some issues and sometimes results in delayed convergence. Convergence is a common problem for deep neural networks during training. The reason behind this is optimization process involves going through complex, non-convex loss surfaces with several saddle points and local minima. Standard optimization techniques, such as Stochastic Gradient Descent (SGD), tend to converge slowly, particularly when dealing with deep models and high-dimensional data. Another significant problem for neural networks is ensuring that they generalize well to new input. When deep models overfit the training set, noise is captured instead of the underlying patterns. On the other hand, models may underfit, failing to account for the complexities of the data.

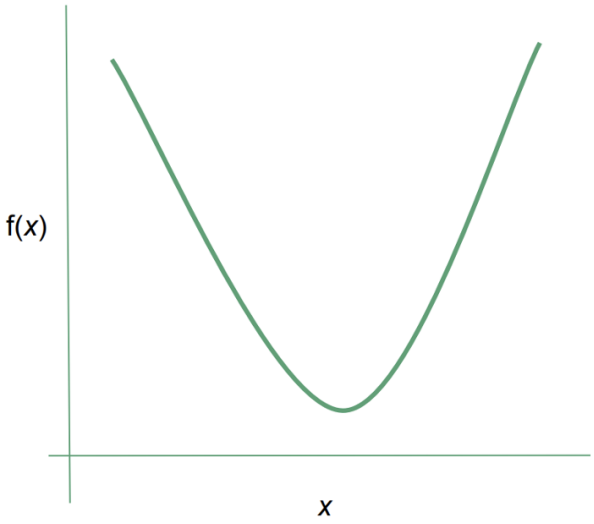


Figure 1. Convex Optimization

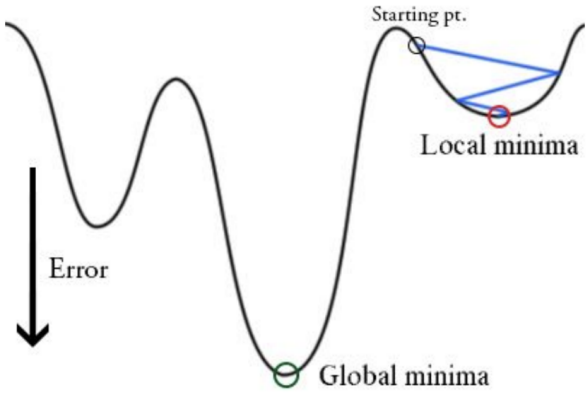


Figure 2. Non-Convex Optimization

To address these issues, this work presents a unique variance adaptive optimization technique for improving deep learning model convergence and generalization. The suggested approach tries to change the learning rate dynamically depending on gradient variance, allowing for adaptive and fine-grained optimization during training. The suggested approach tries to enhance convergence time while avoiding overfitting and boosting generalization capabilities by introducing variance information into the optimization process. In this work, we undertake a thorough investigation to assess the efficacy of the suggested variance adaptive optimization approach. We compare its performance to that of known optimization methods, considering a variety of benchmark datasets and assessment measures. We study the convergence speed, generalization performance, and other relevant parameters through thorough experiments to determine the efficacy of the suggested approach. This research paper's contributions are as follows: First, we

provide a unique variance adaptive optimization strategy that dynamically adapts the learning rate depending on gradient variances. Second, we present a comprehensive study and comparison of the proposed strategy with existing optimization methods, highlighting its strengths and limits. Third, we offer experimental data demonstrating the usefulness of the suggested strategy in terms of improving convergence and generalization in deep learning models.

2. RELATED WORK

Gradient-based learning is a famous and widely used optimization technique in machine learning and deep learning [8]. Gradient-based learning works effectively in convex problem space. Figure 1 shows the convex optimization problem. Gradient-based learning struggles to reach the global optimum in non-convex optimization. Figure 2 shows non-convex optimization. Gradient-based approach get stuck in local minima making it difficult to get out of it. The mini-batch gradient descent algorithm manages to escape shallow local minima; however, it struggles to get out of deep local minima. Deep learning is an important development in the domain of AI it leverages machines to comprehend and understand patterns in a human way [9]. Deep learning provides state-of-the-art architectures like convolutional neural networks, recurrent neural networks and autoencoders to handle a wide variety of problems. Deep learning models are complex and tend to overfit many times. Optimizing the deep neural network is an important aspect of deep learning. Various optimizers have been offered to train deep neural networks. A very common approach used by most of the deep neural networks is gradient descent algorithms. To achieve better convergence gradient-based learning is categorized into two major parts: 1) momentum-based optimization and 2) adaptive learning rate-based optimization. Momentum-based approaches accumulate historical gradients to update the weights. However, it results in oscillating over the global optimum before it converges. Misra [10] proposed a new activation function to advance the general functioning of stochastic gradient descent and adaptive moment-based estimation. Wang et al. [11] their investigation presented an optimization strategy that combines the best properties of ADAM and stochastic gradient descent algorithms. Experimental confirmed the usefulness of the suggested optimizer for non-convex problem spaces. The vanishing gradient problem is very prominent in deep learning because squashing activation functions like sigmoid and tanh tend to get the update values close to zero, resulting in no progress during the learning process. Authors [12] have proposed an evolved gradient direction optimizer to handle the aforementioned issues. The weights here are updated utilizing first-order gradients and hyperplane values. Kim and Choi [13] have proposed an Adam-based hybrid optimization algorithm specifically for convolutional neural networks. The proposed optimization algorithm provided robust and stable performance in a convolutional neural network. In [14] Using dynamic coefficients and composite gradients based on stochastic block coordinate descent, Liu et al. have attempted to overcome Adam's slow convergence. The gradient deviation value is adjusted using adaptive coefficients to adjust the direction of momentum. The random block coordinate determines the gradient update mode. Reyad et al. modified Adam optimizer for deep neural network optimization [15]. The proposed approach adjusts step size automatically over the epochs. The updates are calculated based on the norm values of gradients and utilized dynamically in step updates. Authors of [16] created a new optimization algorithm utilizing the training dataset batch size to increase learning rate adaptively. Lie et al. [17] presented the RAdam algorithm to rectify variance in learning rate. To tackle the problem of local minima, authors of [18] have proposed boosting-based gradient Adam for optimization. Yan and Cai [19] have addressed the issue of poor model generalization by proposing an AdaDB optimizer which works by constraining the learning rate on the upper bound and lower bound of the data. In [20] authors have improved the performance of Adam by adjusting the value of the division coefficient epsilon. In [21] authors have proposed an optimized fuzzy deep learning model utilizing non-dominated sorting genetic algorithm-II for optimization. It addresses the issues of imprecise and uncertain data and noise-sensitive data. It combines deep learning with fuzzy learning and non-dominated sorting genetic algorithm II. The author of [22] has demonstrated the use of deep learning networks using adaptive optimization like RMSprop for the detection of industrial cyber-physical attacks. Reyad et al. in proposed modified version of Adam called as HN-Adam optimizer which automatically adjusts step-size during optimization. HyperAdam is another optimizer which utilizes Adam as baseline optimizer. It uses concept of learning to optimize [23]. Following table 1 provides the recent optimizers and their shortcomings. Our research is trying to mitigate issues related to latest Adam based optimizers.

3. PROPOSED METHODOLOGY

The proposed variance adaptive optimization technique is a unique technique for addressing the issues of deep learning convergence and generalization. This approach tries to dynamically alter the learning rate depending on gradient variance, allowing adaptive optimization during the training process. The suggested approach attempts to create a compromise between convergence speed and generalization performance by using variance information. The proposed variance adaptive optimization approach is based on the idea that gradient variance might give useful insights into the optimization landscape. It uses this data to change the learning rate for every parameter in the deep learning model adaptively. The main concept is to control the learning rate depending on gradient variability, allowing for fine-grained optimization. The suggested approach computes the variance of gradients for each parameter throughout the training process using a continuous window or an exponential moving average. The variance is a measure of the reliability or fluctuation of gradients, reflecting how the optimization process behaves. If the variance is significant, it indicates that the gradients are highly fluctuating, indicat-

TABLE I. Adam Based Optimizer

Optimizer	Key features	Performance	Drawbacks
Radam (Rectified Adam)	- Variance Rectification	Improved stability and convergence, especially in early stages	May still struggle with highly non-stationary environments
HyperAdam	- Dynamically adjusts learning rate, - Task-specific learning	- State-of-the-art performance for MLPs, CNNs, and LSTMs	- Increased computational complexity, - Requires careful tuning
HN Adam (Hypernetwork Adam)	Uses hypernetworks to generate optimizer parameters	- Improved optimization performance	- More complex implementation, - Requires additional resources for hypernetworks
AdaDB (Adaptive Dynamic Bound)	- Dynamically adjusting bounds on learning rates	- More stable and often faster convergence	- Complexity in setting and tuning dynamic bounds, - May not outperform other optimizers in all scenarios
EVGO (Evolving Gradient Optimizer)	- Evolving gradient estimates, - Adapts to training dynamics over time	- Improved convergence rates, - Robustness against noisy gradients	- May require additional tuning, - Can be computationally intensive

ing a complicated optimization surface. A low variance, on the other hand, indicates that the gradients are reasonably steady, indicating a smoother optimization surface. The learning rate for every parameter is adaptively modified utilising the variance values. When the variance is high, suggesting that the optimization landscape is unstable, the learning rate is lowered to guarantee stability and avoid overshooting the optimal solution. When the variance is low, which indicates a more stable optimization landscape, the learning rate is raised to speed up convergence [24]. The proposed approach can efficiently navigate the optimization landscape and optimize the deep learning model due to the dynamic modification of the learning rate exploiting gradient variances. The proposed variance adaptive optimization approach is based on two essential principles: adaptive learning rate regulation and utilizing gradient variance.

A. Adaptive Learning Rate Adjustment

The suggested approach modifies the learning rate dynamically based on gradient variation. The approach makes certain that the optimization method stays stable and effective across varied optimization landscapes by adaptively modifying the learning rate. This adaptive adjustment aids the methodology to react to variations in the optimization environment, resulting in improved convergence and generalization.

B. Leveraging Gradient Variance

The variance of gradients is an effective indication of how the optimization process will behave. It captures the volatility or stability of gradients, offering insights into the optimization landscape's complexity. The suggested approach may alter its optimization strategy based on the unique characteristics of the issue at hand by including gradient variance in the learning rate adjustment. This allows the approach to fine-tune the learning rate and optimize the deep learning model more effectively.

In various areas, the suggested variance adaptive optimization strategy differs from existing optimization methods. While classic methods such as Stochastic Gradient Descent (SGD) and its derivatives employ static learning rates or adaptive approaches based on past gradients, gradient variances are not explicitly included in the learning rate modification process. Furthermore, the proposed variance adaptive optimization approach facilitates fine-grained learning rate modification. The strategy provides an improved and focused optimization approach by modifying the learning rate for each parameter depending on its gradient variance. This fine-grained change improves optimization efficiency, especially in complicated deep neural network models with many parameters. In comparison to current adaptive optimization methods such as AdaGrad, RMSprop, and Adam, the proposed strategy has a notable benefit in that it incorporates gradient variance directly into learning rate adjustment. While these adaptive approaches consider previous gradients, they may accrue too many variations over time, resulting in overfitting. The suggested approach, on the other hand, concentrates on gradient instantaneous variance, offering a more up-to-date estimate while avoiding possible difficulties associated with accumulated variances.

Let's consider a general optimization objective for deep learning: Minimize: $E(w) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(x_i; w))$ (1)

where: E(w): is the objective function to be minimized, w: represents the model parameters, N: is the total number of training samples, L(yi, f(xi; w)): is the loss function that measures the discrepancy between the predicted output f(xi; w) and the ground truth label yi. Consider existing optimization algorithm as Adaptive Moment Estimation (ADAM): Adam optimizer is a combination of momentum and Root Mean Square propagation (RMSprop) algorithm. The key idea behind Adam is to calculate two moving averages of parameters. The first moment i.e., means and the second moment i.e., an uncentered variance of gradients. These moving averages are utilized adaptive update of learning rate during training. However, Adam applies bias correction to moments because during initialization moving averages are biased towards zero. The Adam update equations are given as follows:

ADAM Optimization:

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \quad (2)$$

$$v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2 \quad (3)$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad (4)$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \quad (5)$$

$$w_t = w_{t-1} - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon} \quad (6)$$

where, m_t and v_t represent the first and second moments of the gradients at time step t, β_1 and β_2 are the decay rates for the first and second moments, respectively, g_t represents the gradient at time step t, $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected estimates of the moments, η is the learning rate, ϵ is a small constant for numerical stability.

The proposed variance adaptive optimization is given below,

Proposed Variance Adaptive ADAM (VADAM):

$$\text{var}_t = \beta_{\text{var}} \cdot \text{var}_{t-1} + (1-\beta_{\text{var}}) \cdot g_t^2 \quad (7)$$

$$\hat{\text{var}}_t = \frac{\text{var}_t}{1-\beta_{\text{var}}^t} \quad (8)$$

$$\text{lr}_t = \frac{\eta}{\sqrt{\hat{\text{var}}_t}+\epsilon} \quad (9)$$

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \quad (10)$$

$$v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2 \quad (11)$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad (12)$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \quad (13)$$

$$w_t = w_{t-1} - \text{lr}_t \cdot \frac{\hat{m}_t}{\sqrt{\hat{\text{var}}_t}+\epsilon} \quad (14)$$

Where, var_t and $\hat{var}_t$ represents the variance of gradients at time step t and its bias-corrected estimate, respectively, $\beta_v ar$ is the decay rate for the variance, lr_t is the adapted learning rate based on the variance, the rest of the terms are the same as in the ADAM update equations. The flow diagram depicting the working of the VADAM is shown in Figure 3. The pseudo-code of the VADAM is shown in the following figure 4.

The suggested VADAM differs from previous optimization algorithms in the learning rate adaption procedure. Unlike previous approaches such as ADAM, which change the learning rate based on the first and second moments

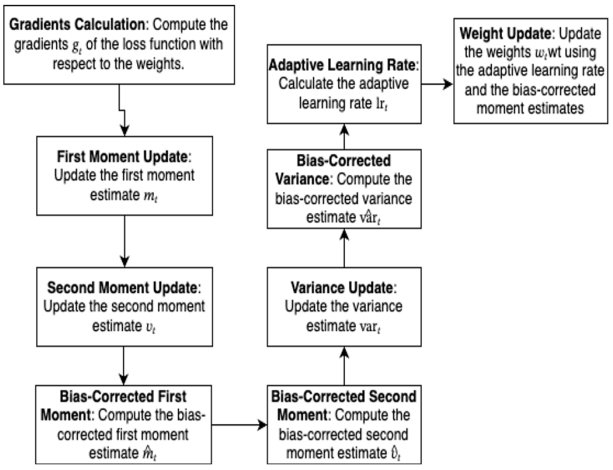


Figure 3. Workflow for VADAM

```
def _prepare_local(self, var_device, var_dtype, apply_state):
    super(VAdam, self)._prepare_local(var_device, var_dtype, apply_state)
    apply_state[(var_device, var_dtype)]["learning_rate_t"] = tf.identity(self.learning_rate)
    apply_state[(var_device, var_dtype)]["beta1_t"] = tf.identity(self.beta1)
    apply_state[(var_device, var_dtype)]["beta2_t"] = tf.identity(self.beta2)
    apply_state[(var_device, var_dtype)]["epsilon_t"] = tf.identity(self.epsilon)
    apply_state[(var_device, var_dtype)]["ω_t"] = tf.identity(self.ω)

def _create_slots(self, var_list):
    for var in var_list:
        self.add_slot(var, "m")
        self.add_slot(var, "v")

def _resource_apply_dense(self, grad, var):
    var_dtype = var.dtype.base_dtype
    lr_t = self._get_hyper("learning_rate", var_dtype)
    beta1_t = self._get_hyper("beta1", var_dtype)
    beta2_t = self._get_hyper("beta2", var_dtype)
    epsilon = self._get_hyper("epsilon", var_dtype)
    ω = self._get_hyper("ω", var_dtype)

    m = self.get_slot(var, "m")
    v = self.get_slot(var, "v")

    m_t = m.assign(beta1_t * m + (1. - beta1_t) * grad)
    v_t = v.assign(beta2_t * v + (1. - beta2_t) * tf.square(grad))

    r = m_t / (tf.sqrt(v_t + ω * v) + epsilon)
    var_update = tf.assign_sub(var, lr_t * r)

    return tf.group(*[var_update, m_t, v_t])
```

Figure 4. Pseudo code for VADAM

of gradients, VADAM adds gradient variation directly into the learning rate adaption. The addition of variance-based adaptation enables VADAM to dynamically change the learning rate depending on gradient variability. This can contribute to better convergence and generalization performance, especially in circumstances with complicated optimization landscapes or noisy gradients. VADAM provides a more fine-grained and adaptable optimization technique by considering instantaneous variance. Through experimental assessment, comparing the convergence time, generalization performance, and other important metrics of VADAM with standard ADAM and other current optimization methods, the distinctive influence of variance adaptation on the optimization process can be detected. The efficacy of the proposed VADAM may be objectively measured and compared with existing approaches using these studies. Overall, the mathematical comparison demonstrates VADAM's distinguishing feature of utilizing gradient variance for adaptive learning rate modification, which distinguishes it from typ-

ical optimization methods. Table 2 shows the comparison of VADAM against other optimizers.

- Learning Rate Adaptation: VADAM and RMSprop use adaptive learning rate adaptation, which dynamically modifies the learning rate during training. In contrast, ADAM and SGD have fixed or manually controlled learning rates.
- Adaptive Moment Estimation: VADAM, ADAM, and RMSprop use adaptive moment estimation for adaptive learning rate modification, which covers first and second-moment estimates. SGD is devoid of adaptive moment estimation.
- Handling Variance: VADAM integrates variance-based adaptation, taking gradient variability into account. In their learning rate adaptation, ADAM, RMSprop, and SGD do not directly manage variation
- Convergence Speed: Because of their adaptive learning rate modifications and use of moment estimates, VADAM and ADAM tend to have higher convergence speeds than RMSprop and SGD.
- Generalization Performance: When compared to SGD, VA ADAM, ADAM, and RMSprop have better generalization performance. However, the addition of variance-based adaptation to VADAM may improve its generalization capabilities.

It's vital to remember that the efficiency of various optimization techniques varies depending on the issue, dataset, and model design. The table compares them in general terms based on their major properties.

4. EXPERIMENTATION AND RESULTS

For the experimentation, we have developed deep neural networks from scratch using TensorFlow and sklearn libraries. We have utilized multilayer perceptron architectures for Breast Cancer, PIAMA Indian Diabetes and Cancer datasets. Following diagram 5 shows a deep feed-forward neural network. The input value is subject to the number of features in respective datasets. We have not utilized transfer learning approaches exploiting existing pre-trained models like VGG16, ResNet50 or InceptionNet. For image datasets like MNIST, CIFAR10 and Fashion-MNIST we have created a convolutional neural network (CNN) from the scratch. The illustration of the CNN blocks for the CIFAR10 dataset is shown in Figure 6. To validate the variance adaptive optimization, we have utilized various datasets from the UCI repository as well as from the Kaggle. The following table shows the various datasets and their category.

Table 3 discusses the datasets utilized for experimentation. We have considered both binary and multiclass classification datasets. Also, to study the robustness of algorithms across deep learning frameworks, we have considered standard image datasets. The characteristics of datasets are discussed in Table 4. The experiment is conducted in a Google collaboratory environment with GPU support for the training and testing of the optimizer. The baseline configurations of the hyperparameters required are shown in Table 5.The experimentation has been performed to analyze the functioning of the suggested variance adaptive methodology against Adam and the stochastic gradient descent algorithm. For experimentation purposes, we have kept the batch size at 32 and the epochs to 10 except MNIST dataset. For the MNIST dataset, the epochs is set to 5. The dataset considered for the experimentation is small and requires less time to train the model. The intention of considering the small quantity of epochs is based on the dataset size. However, parameters are like a number of epochs, and learning rates are not dataset or experiment specific. These are tunable parameters and set to some value based on the experiment's requirements. Table 5 demonstrates the training time taken by each optimizer across various datasets. Training loss on each dataset is shown in Table 6. During experimentation, we observed training loss for all the optimizers. The details of training loss are shown in Table 7 and Figure 8. The convergence rate for the VADAM is measured using Equation 15. Given a sequence of loss values L_t where L_t is the loss at iteration t, the rate of convergence at iteration t is defined as:

$r_t = \frac{L_{t-1} - L_t}{L_{t-1}}$ (15) The figure 7 shows the convergence of the VAdam and Adam on the breast cancer dataset. Various models are developed utilizing variance adaptive Adam, Adam and SGD optimizers. The test performance of each optimizer on the test dataset is presented in Figure 9. Table 8 shows the performance of the optimizer on the test dataset. From the obtained results we can observe that VAdam converges rapidly in contrast to the existing contemporary

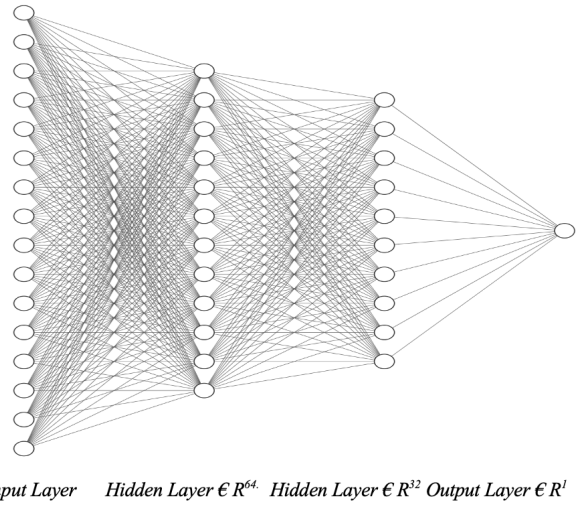


Figure 5. Deep Neural Network for Breast Cancer, PIAMA Indian Diabetes and Cancer datasets

TABLE II. Comparison of Optimization Algorithms

Optimization Algorithm	Learning Rate Adaptation	Adaptive Moment Estimation	Handling Variance	Convergence Speed	Generalization Performance
VADAM	Variance-based	Yes	Yes	Fast	Improved
ADAM [25]	Momentum-based	Yes	No	Fast	Good
RMSprop	Adaptive learning rate	No	No	Moderate	Moderate
SGD	Fixed learning rate	No	No	Slow	Moderate

optimizers. The following tables present classification reports for experiments to observe the performance of VAdam in case of data imbalance. Tables 9, 10 and 11 demonstrate the classification report for Breast Cancer, MNIST and CIFAR10 datasets.

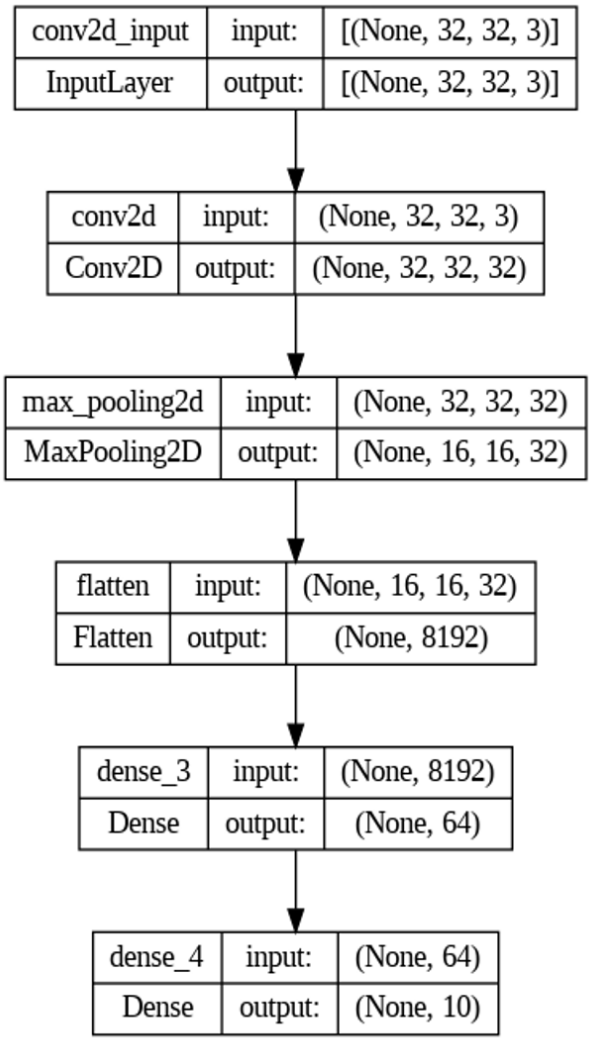
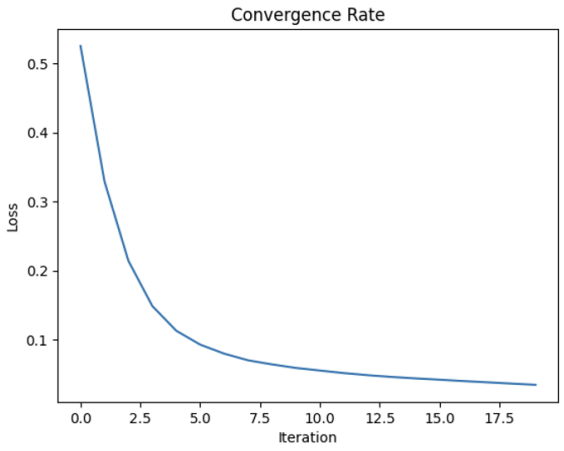
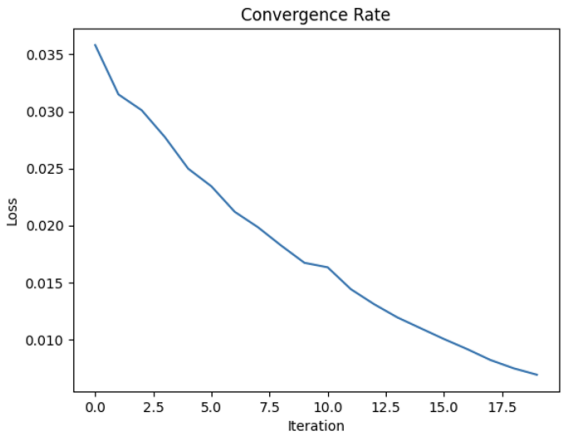


Figure 6. CNN model for CIFAR10 dataset



a) VADAM convergence



b) ADAM Convergence

Figure 7. Convergence rate for Breast Cancer Dataset

5. EXPERIMENT LIMITATIONS AND CONSTRAINTS

During the experimentation, VAdam is trained and validated on the existing toy datasets. Also, we have observed the effectiveness of VAdam for classification problems. Some of the experiments are performed on image datasets using VAdam and CNNs to demonstrate the scale and variations in applicability. However, VAdam is not validated on huge datasets and sequential datasets for its performance. We propose to perform high computational experiments in the near future.

TABLE III. Experimentation Datasets

Sr. No.	Dataset	Type
1	Breast Cancer Dataset [26]	Binary classification
2	MNIST Dataset [27]	Multi-class classification
3	CIFAR10 Dataset [28]	Multi-class classification
4	PIAMA Indian Diabetes Dataset [29]	Binary classification
5	Cancer Dataset	Binary classification
6	Fashion MNIST Dataset [30]	Multi-class classification

TABLE IV. Dataset Characteristics

Dataset	Features	Classes	Instances
Breast Cancer	30 numeric features	2 classes	569 instances
MNIST	28x28 grayscale images	digits 0-9	60,000 training and 10,000 testing images
CIFAR-10	32x32 color images	10 classes	50,000 training and 10,000 testing images
PIMA Indian Diabetes	8 numeric features	2 classes	768 instances
Cancer Dataset	28 numeric features	2 classes	857 instances
Fashion MNIST	28x28 grayscale images	10 classes	60,000 training and 10,000 testing images

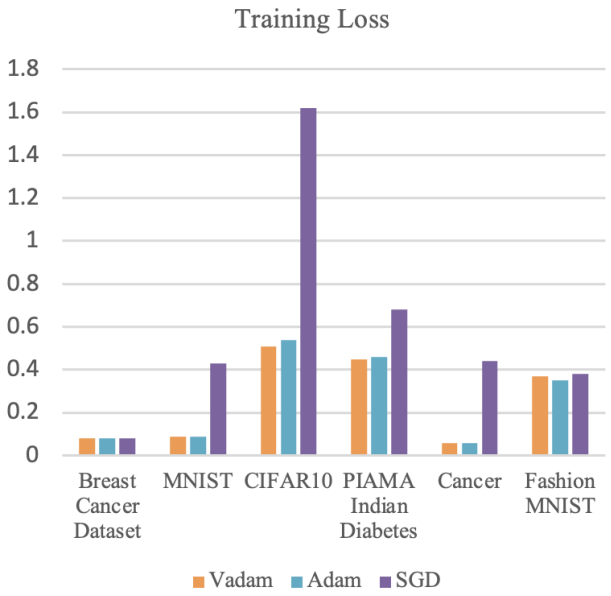


Figure 8. Training Loss

TABLE V. Hyperparameter Settings

Hyperparameters	**Value**
Learning rate	0.001
β_1	0.9
β_2	0.999
Epsilon (ε)	10^{-8}
ω	0.5

Figure 9. Optimizer performance - Test Accuracy

TABLE VI. Training time

Dataset	Optimizer	Training Time (in seconds)
Breast Cancer Dataset	VAdam	2.65
	Adam	3.69
	SGD	4.27
MNIST	VAdam	30.95
	Adam	31.89
	SGD	32.30
CIFAR10	VAdam	84.30
	Adam	67.50
	SGD	84.03
PIAMA Indian Diabetes	VAdam	3.76
	Adam	2.62
	SGD	1.75
Cancer	VAdam	1.94
	Adam	3.46
	SGD	1.92
Fashion MNIST	VAdam	62.90
	Adam	82.75
	SGD	65.20

TABLE VII. Training Loss

Dataset	**Vadam**	**Adam**	**SGD**
Breast Cancer Dataset	0.08	0.08	0.08
MNIST	0.09	0.09	0.43
CIFAR10	0.51	0.59	1.62
PIAMA Indian Diabetes	0.45	0.46	0.68
Cancer	0.06	0.06	0.44
Fashion MNIST	0.37	0.35	0.38

TABLE VIII. Test Accuracy

Dataset	Optimizer	Test Accuracy
Breast Cancer Dataset	Vadam	96.5
	Adam	98.2
	SGD	98.2
MNIST	Vadam	87.18
	Adam	86.74
	SGD	88.7
CIFAR10	Vadam	64.21
	Adam	61.7
	SGD	43.01
PIAMA Indian Diabetes	Vadam	81.1
	Adam	77.9
	SGD	50.6
Cancer	Vadam	97.4
	Adam	96.5
	SGD	92.9
Fashion MNIST	Vadam	87.41
	Adam	87.7
	SGD	86

TABLE IX. Classification Report for Breast Cancer Dataset

	Precision	Recall	F1-Score	Support
0	1.00	0.98	0.99	43
1	0.99	1.00	0.99	71
Accuracy		0.99		114
Macro Avg	0.99	0.99	0.99	114
Weighted Avg	0.99	0.99	0.99	114

TABLE X. Classification Report for MNIST Dataset

	Precision	Recall	F1-Score	Support
0	0.98	0.99	0.99	980
1	0.99	0.98	0.99	1135
2	0.95	0.99	0.97	1032
3	0.97	0.98	0.97	1010
4	0.98	0.97	0.98	982
5	0.98	0.98	0.98	892
6	0.98	0.98	0.98	958
7	0.99	0.93	0.96	1028
8	0.94	0.98	0.96	974
9	0.97	0.96	0.97	1009
Accuracy		0.97		10000
Macro Avg	0.97	0.97	0.97	10000
Weighted Avg	0.97	0.97	0.97	10000

TABLE XI. Classification Report for CIFAR10 Dataset

	Precision	**Recall**	**F1-Score**	**Support**
0	0.90	0.79	0.84	1000
1	0.95	0.92	0.93	1000
2	0.90	0.66	0.76	1000
3	0.62	0.78	0.69	1000
4	0.80	0.85	0.82	1000
5	0.76	0.78	0.77	1000
6	0.81	0.91	0.86	1000
7	0.93	0.86	0.89	1000
8	0.95	0.89	0.92	1000
9	0.87	0.94	0.90	1000
Accuracy		0.84		10000
Macro Avg	0.85	0.84	0.84	10000
Weighted Avg	0.85	0.84	0.84	10000

6. CONCLUSION AND FUTURE WORK

Throughout the paper, we intend to advance the optimization process for deep neural networks by proposing consideration of gradient variance during learning rate adaption. We have compared our results against popular optimization algorithms such as Adam and stochastic gradient descent. The existing Adam algorithm is modified to adapt gradient variance for improved convergence and generalization. Proposed variance adaptive Adam outperforms stochastic gradient descent as well as Adam optimizers in overall training accuracy and convergence time. To conclude, experimental results indicate that the proposed VAdam optimizer is efficient as well as effective contrasted with the existing contemporary optimizers. The effectiveness of VAdam can be used in deep learning applications to tackle the common issues of convergence and generalization. In future work, we intend to work on time series applications to check the effectiveness of VAdam. Also, the effectiveness of the VAdam is to be validated against the vanishing gradient and exploding gradient issues.

REFERENCES

[1] F. Mehmood, S. Ahmad, and T. K. Whangbo, "An efficient optimization technique for training deep neural networks," *Mathematics*, vol. 11, no. 6, p. 1360, 2023. [Online]. Available: https://doi.org/10.3390/math11061360

[2] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[4] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[5] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[6] Y. Zhang, D. Zhou, M. Chen, and X. Yuan, "An overview of optimization methods for deep learning," *Complexity*, p. 6670487, 2021.

[7] A. Mustapha, L. Mohamed, and K. Ali, "Comparative study of optimization techniques in deep learning: Application in the ophthalmology field," *Journal of Physics: Conference Series*, vol. 1743, no. 1, p. 012002, jan 2021. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/1743/1/012002

[8] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[10] D. Misra, "Mish: A self regularized non-monotonic activation function," *arXiv*, 2019.

[11] Y. Wang, P. Zhou, and W. Zhong, "An optimization strategy based on hybrid algorithm of adam and sgd," in *MATEC Web Conf.*, vol. 232, 2018, p. 03007.

[12] I. Karabayir, O. Akbilgic, and N. Tas, "A novel learning algorithm to optimize deep neural networks: Evolved gradient direction optimizer (evgo)," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 2, pp. 685–694, 2021.

[13] K.-S. Kim and Y.-S. Choi, "Hyadamc: A new adam-based hybrid optimization algorithm for convolution neural networks," *Sensors*, vol. 21, no. 12, p. 4054, 2021.

[14] M. Liu, D. Yao, Z. Liu, J. Guo, and J. Chen, "An improved adam optimization algorithm combining adaptive coefficients and composite gradients based on randomized block coordinate descent," *Computational Intelligence and Neuroscience*, p. 4765891, 2023.

[15] M. Reyad, A. Sarhan, and M. Arafa, "A modified adam algorithm for deep neural network optimization," *Neural Comput Applic*, vol. 35, p. 17095Äì17112, 2023.

[16] Z. Manzil, R. Sashank, S. Devendra, K. Satyen, and K. Sanjiv, "Adaptive methods for nonconvex optimization," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 9793–9803.

[17] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," *arXiv*, 2019.

[18] J. Bai, Y. Ren, and J. Zhang, "BGADAM: Boosting based Genetic-Evolutionary ADAM for neural network optimization," *arXiv*, 2019.

[19] L. Yang and D. Cai, "Adadb: An adaptive gradient method with data-dependent bound," *Neurocomputing*, vol. 419, pp. 183–189,

2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231220311784

[20] W. Yuan and K.-X. Gao, "EAdam optimizer: How ϵ impact adam," *arXiv*, 2020.

[21] A. Yazdinejad, A. Dehghantanha, R. M. Parizi, and G. Epiphaniou, "An optimized fuzzy deep learning model for data classification based on nsga-ii," *Neurocomputing*, vol. 522, pp. 116–128, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222015260

[22] J. Sakhnini, H. Karimipour, A. Dehghantanha, A. Yazdinejad, T. R. Gadekallu, N. Victor, and A. Islam, "A generalizable deep neural network method for detecting attacks in industrial cyber-physical systems," *IEEE Syst. J.*, pp. 1–9, 2023.

[23] S. Wang, J. Sun, and Z. Xu, "HyperAdam: A learnable Task-Adaptive adam for network training," *arXiv*, 2018.

[24] S. Smith and A. Johnson, "Variance adaptive optimization technique for deep learning," in *Neural Computing and Applications*, 2023.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[26] M. S. Iqbal, W. Ahmad, R. Alizadehsani, S. Hussain, and R. Rehman, "Breast cancer dataset, classification and detection using deep learning," *Healthcare (Basel, Switzerland)*, vol. 10, no. 12, p. 2395, 2022. [Online]. Available: https://doi.org/10.3390/healthcare10122395

[27] E. Plesovskaya and S. Ivanov, "Hierarchical classification on the mnist dataset using truncated svd and kernel density estimation," *Procedia Computer Science*, vol. 212, pp. 368–377, 2022.

[28] A. Ganguly, A. U. Ruby, and G. C. J. Chandran, "Evaluating cnn architectures using attention mechanisms: Convolutional block attention module, squeeze, and excitation for image classification on cifar10 dataset," 2023, published on 13 August 2023.

[29] V. Chang, J. Bailey, Q. A. Xu *et al.*, "Pima indians diabetes mellitus classification based on machine learning (ml) algorithms," *Neural Computing and Applications*, vol. 35, pp. 16 157–16 173, 2023. [Online]. Available: https://doi.org/10.1007/s00521-022-07617-1

[30] N. Snehith, M. T. N. D. S. Harsha, S. Bodempudi, L. R. L. Bailundo, S. Shameem, and J. V. Namgiri, "Reconstructing noised images of fashion-mnist dataset using autoencoders," in *2023 International Conference on Ambient Intelligence, Knowledge Informatics and Industrial Electronics (AIKIIE)*, Ballari, India, 2023, pp. 1–6.