Advances in Nonlinear Variational Inequalities
ISSN: 1092-910X
Vol 28 No. 1s (2025)

Performance Analysis of ML and DL Models: Impact of Linear and Non-Linear Optimizers on Model Efficiency

G. Amulya[1], A. Jyothi[2], Saritha Dasari [3, *], E Sandhya[4], Dr. Rajesh Kumar K V[5]

[1]Assistant Professor, Department of CSE, G. Narayanamma Institute of Technology and Science, Hyderabad.

Email: amulya@gnits.ac.in

[2] Assistant Professor, Department of CSE, G. Narayanamma Institute of Technology and Science, Hyderabad.

Email: jyothi@gnits.ac.in

[3,*] Assistant Professor, Department of CSE(Data Science), G. Narayanamma Institute of Technology and Science, Hyderabad. Email: saritha.dasari@gnits.ac.in

[4]Assistant Professor, Department of CSE(AI&ML), Madanapalle Institute of Technology & Science, Madanapalle.

Email: sandhyaethala@gmail.com

[5]Associate Professor, AI Research Centre, School of Business, Woxsen University, Hyderabad.

Email: kvrkkumar@gmail.com

Article History:

Received: *18-08-2024*

Revised: *07-10-2024*

Accepted: *20-10-2024*

Abstract:

Introduction: Emphasizing the effect of linear and non-linear optimizers on the performance of machine learning (ML) and deep learning (DL) models, this work addresses the CIFAR-10 image classification task. The work explores over multiple data sets how optimizers influence model behavior and efficiency.

Objectives: With a view on five primary assessment metrics—accuracy, precision, recall, F1-score, and AUC the main purpose is to evaluate on SVM and CNN models the performance of linear optimizers (gradient descent, SGD) and non-linear optimizers (Adam, RMSProp).

Methods: The study offers fair and complete comparison by way of a consistent evaluation strategy. Analyzing how different optimizer influences model convergence rates, computational cost, and training stability takes front stage in the experimental setting. Designed on CIFAR-10, optimizers are used on ML and DL models whose performance on all metrics is noted.

Results: Results show that non-linear optimizers especially Adam much increase CNN model performance by means of faster convergence, higher classification accuracy, and better model stability. While helpful for simpler models, linear optimizers such SGD show slower convergence and limited adaptation with more complicated data.

Conclusions: This study provides direction on selecting appropriate optimizers depending on model complexity, job needs, and computing restrictions coupled with important information on optimizing model training in image classification and other areas.

Keywords: Linear Optimizers, Non-Linear Optimizers, Machine Learning, Deep Learning, Image Classification.

1. Introduction

From entertainment and driverless cars to banking and healthcare, machine learning (ML) and deep learning (DL) models have revolutionized various disciplines [1]. The process of optimization affects the efficiency of ML and DL models since it can change parameters to lower errors and improve performance [2]. Optimizing methods [3] derive the generalizing capacity to raw data and the efficiency of a model during training. The choice of optimizer is the main determinant of the general efficiency, accuracy, and speed of both ML and DL models; consequently, it is a crucial ability in model development. Driven mostly by their simplicity and direct approach to changing model parameters dependent on the gradient of the loss function [4], traditional machine learning makes use of linear optimizers including Gradient Descent (GD). Sometimes unsuccessful, especially in high-dimensional or non-convex contexts, these basic to use, linear paths in parameter space help to lower error.

Conventional machine learning mostly uses linear optimizers such as GD often due to their simplicity and direct approach in modifying model parameters dependent on the gradient of the loss function. One variant of GD, Stochastic Gradient Descent (SGD), for example, may have somewhat slow convergence and great volatility in gradient updates. From these restrictions, underperformance in advanced deep learning models and longer training times follow. On the other hand, non-linear optimizers such as Adam (Adaptive Moment Estimation) and RMSProp (Root Mean Square Propagation) have become well-known in deep learning since they can dynamically modify learning rates and add momentum to gradient updates. These optimizers allow non-linearities in the update rule, hence enabling more effective navigation of ever-difficult loss surfaces. Adam blends adaptive learning rates with momentum-based optimization [5] to be powerful in noisy gradients and able to faster convergence on deep neural networks.

Similarly, RMSProp independently changes learning rates for every parameter, hence suitable for applications using recurrent neural networks and big datasets [6]. The choice of an inappropriate optimizer can cause various issues. Particularly in large neural networks, some models may induce delayed convergence or failure to escape local minima from linear optimizers. This can provide fewer than desired results and cut off training sessions. Moreover, using a non-linear optimizer with improperly calibrated hyperparameters such as momentum or learning rate may generate divergence or unstable training behavior [7]. Consequently, as their performance can greatly affect generalization and model efficiency [8], much attention should be paid to selecting between linear and non-linear optimizers. One has to know the performance of several optimizers for several different purposes. Firstly, it enables academics and practitioners to find which optimizers fit particular kinds of models and datasets, hence enhancing accuracy and cutting training durations.

Furthermore, in practical applications where computer resources are sometimes constrained, optimizer performance directly determines model scalability. Employing comparison of optimizer performance, practitioners can make well-informed decisions improving model reliability and efficiency, therefore enabling remarkable successes in various fields. This work establishes model efficiency and the value of this attempt to strictly investigate how different optimization strategies affect the performance of ML and DL models. Understanding how linear and non-linear optimizers affect convergence rates, accuracy, and computational resources helps one to provide a perceptive analysis of ideal model efficiency. Using these realizations, practitioners can choose optimizers for their specific operations, so enhancing the dependability and performance of their models [9]. Our main contributions to this study consist as follows:

1. We evaluated linear and nonlinear optimizers over ML and DL models to find their effects on classification metrics for CIFAR-10.
2. We assess the computational cost and efficiency of every optimizer Emphasizing training time and resource constraints.
3. We investigated CNN convergence rates and SVM classifier accuracy to underline variations in training behavior under different optimizers.

This work aims to explore ML and DL model efficiency under both linear and non-linear optimizers. Specifically, taught on benchmark data, we evaluate their performance with logistic regression and CNN (convolutional neural networks). This paper aims to give a comprehensive comparison of several optimization techniques by assessing important parameters including accuracy, convergence speed, and computation cost. Using insights on the useful applications of many optimizers, the aim is to clarify how optimization choices influence model performance spanning many domains. We present a method based on linear and non-linear optimizers to evaluate how ML and DL model performance is changed. Maintaining a constant framework for experimentation guarantees that the comparisons are accurate and significant, so offering valuable insights into the field of machine learning.

2. OBJECTIVES

This paper attempts to investigate, concerning classification metrics, the effects of linear and non-linear optimizers on machine learning and deep learning models using the CIFAR-10 dataset. This means assessing the computational cost and efficiency of every optimizer considering training time and resource constraints. Furthermore examined in the research are changes in training behavior by comparing various optimizers and the convergence rates of convolutional neural networks (CNNs) with the accuracy of support vector machine (SVM) classifiers. The work intends to present a thorough investigation of how different optimization strategies affect model performance in terms of accuracy, convergence speed, and computational cost using insights on optimization decisions. Furthermore, a consistent experimental framework is built to provide accurate and significant comparisons, so offering an interesting study of the role of optimization methods in raising model efficiency throughout many uses.

3. RELATED WORKS

Optimizers are basic components of their training since they directly influence the efficiency and effectiveness of ML and DL models. Ruder et al. [3] stress the need for optimization techniques since the choice of optimizer could affect model accuracy as well as convergence speed. Though in complex situations they can have sluggish convergence rates, generally used traditional procedures include Gradient Descent (GD), and its derivatives, including Stochastic Gradient Descent (SGD), are simplicity and effectiveness-based. Zaidi et al. [10] studied the performance of numerous optimizers on photo categorization tasks. Their results revealed, particularly about deep neural networks, that Adam routinely outperforms SGD in terms of speed and precision. They did, however, also caution that Adam's performance might be somewhat sensitive to hyperparameter choices, therefore confounding its practical significance [4]. If models are not suitably regularized, the risk of overfitting still worries me.

Loshchilov et al. [11] found that these methods are ideal for handling the complex loss terrain typical of deep learning problems. Especially in the context of class imbalance generally found in real-world datasets, Anh-Tuan Nguyen et al. [12] stressed that getting optimal model performance entirely depends on the choice of a suitable optimizer. Their findings suggest that particularly in cases with noisy data [13], a poorly chosen optimizer could produce significant performance decreases. Furthermore emphasized by Smith et al. [14] the requirement of learning rate schedules in training deep learning models since, if not properly regulated, flexible learning rates can cause instability even if they can greatly improve convergence [15].

Combining conventional and deep learning optimizers in hybrid models has also lately become somewhat prevalent in search of improved performance. Even if the optimization of algorithms has evolved, knowledge of the most efficient combinations of optimizers for different applications still lags [16]. This corpus of research underlines the significance of choosing the suitable optimizer in ML and DL applications given its main influence on model efficiency and effectiveness. Future research should keep looking for new optimizers and their application into several machine learning architectures if performance over several workloads is to be raised.

4. METHODOLOGY

Along with the optimization strategies used, we provide a thorough study of the architecture and operational methods of several ML and DL models in this part. The structure, training process, and the functions of linear and non-linear optimizers in improving model performance are thoroughly described in this part of the work. Emphasizing decision patterns for the CIFAR-10 dataset to effectively address challenging image classification tasks, the Figure 1 and Figure 2 are presented to show the workflow of our research framework and the sample images from the dataset.

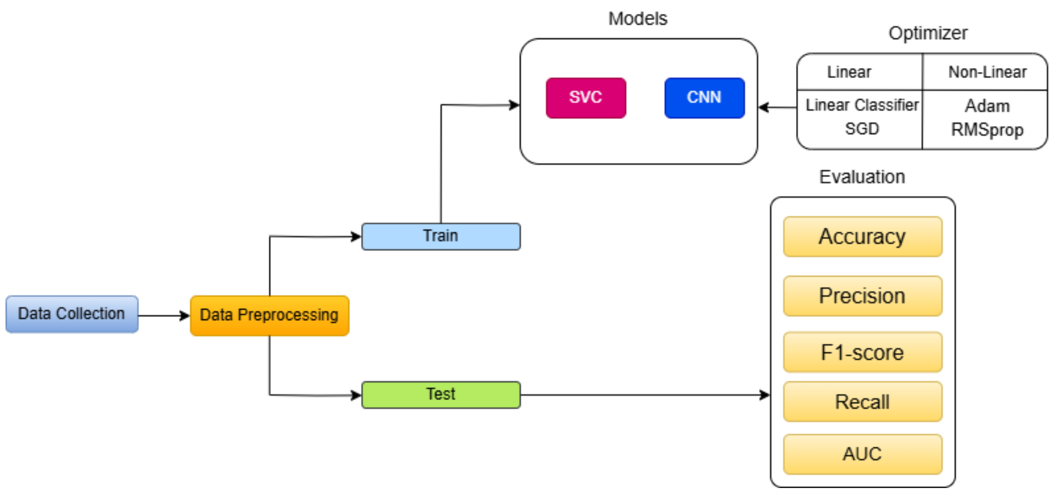


Figure 1: The workflow of our research framework

Figure 2: Sample Images from Different Classes of the CIFAR-10 Dataset

4.1. Optimizers

We applied for CIFAR-10 image classification task training ML and DL models. We investigate, on several assessment criteria including model accuracy and computational efficiency, the effects of linear optimizers (Linear Classifier and Stochastic Gradient Descent) and non-linear optimizers (Adam and RMSProp).

Linear Optimizers: Usually serving as benchmarks for computing efficiency, linear optimizers perform direct parameter changes. Still, they might not be sufficient for handling non-linear patterns inside complex datasets.

Linear Classifier: The linear classifier updates parameters over the whole dataset for every iteration by use of batch gradient descent. Gradient descent has an updating rule like this:

$$\theta = \theta - \eta \nabla L(\theta)$$

Where the model parameters are represented by θ, η refers to the learning rate, and, the gradient of the loss function $L(\theta)$ concerning the parameters is $\nabla L(\theta)$.

Gradient descent in image classification progressively changes weights by reducing the total model prediction error. Using the whole batch per iteration is computationally expensive and might lead to poor convergence due to the huge dataset size (such as CIFAR-10), hence this method is less feasible for DL models but good for ML baseline comparison.

Stochastic Gradient Descent (SGD): Stochastic Gradient Descent greatly accelerates convergence by changing gradient descent parameters depending on a subset (or mini-batch) of the data instead of the whole dataset. SGD's updating rule is provided by:

$$\theta = \theta - \eta \nabla L(\theta; x_i, y_i)$$

where (x_i, y_i) is an individual data point or a mini-batch from the dataset.
Stochastic Gradient Descent (SGD) provides clear benefits for image classification problems, where the dataset size is significant and the model architecture may be complicated. SGD generates some unpredictability by using mini-batches, therefore enabling the model perhaps escape local minima in high-dimensional parameter spaces common to image data. This feature is particularly helpful in CNNs, widely employed in image classification since the variation in updates helps explore more of the solution space, hence improving generalization on unknown data. It also brings variance into the updates by using smaller data samples, therefore enabling the model to escape local minima. It also brings instability, hence careful calibration of the learning rate η helps to prevent oscillations in training.

Non-linear Optimizers: For deep models, non-linear optimizers are quite powerful since they bring adaptive learning rates and momentum. These optimizers change depending on gradient patterns, hence improving performance on non-linear tasks including image classification.

Adam (Adaptive Moment Estimation): Adam integrates adaptive learning rates (to improve updates) with momentum to hasten convergence. It computes first and second-moment estimates of the gradients, therefore preventing overshooting in parameter changes. Adam's update equations consist of:
Updating the first (m) and second moment estimates (v):

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla L(\theta_t)$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\big(\nabla L(\theta_t)\big)^2$$

where m_t and v_t are the first (mean) and second (variance) moment estimates, β_1 and β_2 are decay rates, and $\nabla L(\theta_t)$ is the gradient at step t.
Correcting the bias:

$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Towards unbiased estimations, bias correction modifies m_t and v_t.
Updating the parameter:

$$\theta_{t+1} = \theta_t - \eta \frac{m_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Where a little constant ϵ halts division by zero. Adam's capacity to dynamically change learning rates for every parameter, which is especially helpful for sophisticated deep learning models like convolutional neural networks (CNNs) helps to explain why it is effective in picture classification challenges. In our CIFAR-10 image classification task, Adam’s adaptive learning rate per parameter is particularly beneficial due to the diverse and complex nature of CNN layers, each requiring different learning rates to efficiently capture intricate spatial hierarchies. By combining momentum with adaptive learning rates, Adam enhances convergence speed and maintains training stability across the

network's complex gradient landscape. This functionality allows for efficient learning of both low-level features, like edges and textures, and high-level features, such as shapes and patterns. Ultimately, Adam not only reduces training time but also boosts accuracy, ensuring effective feature extraction across layers and yielding strong performance on image classification tasks.
RMSProp (Root Mean Square Propagation): RMSProp stabilizes updates by varying learning rates depending on the moving average of squared gradients, therefore addressing oscillations. This method is very helpful for environments in non-convex optimization observed in DL problems. The rule governing RMSProp updates is:
Computing the moving average of squared gradients:

$$v_t = \gamma v_{t-1} + (1-\gamma)\big(\nabla L(\theta t)\big)^2$$

Where γ is a decay rate and v_t is the average of squared gradients.
Updating the parameter:

$$\theta_{t+1} = \theta_t - \eta \frac{m_t}{\sqrt{\hat{v}_t} + \epsilon} \nabla L(\theta t)$$

Where ϵ is a tiny constant and η is the learning rate. The adaptive learning rate of RMSProp shows great value in our image classification task on the CIFAR-10 dataset since it can be modified to the various sizes of gradients across several layers in the network. This flexibility helps to stabilize deep network training, where gradients might vary greatly. Through more consistent learning of low- and high-level features, RMSProp helps to smooth updates, so boosting convergence speed and classification accuracy on challenging picture data.

4.2. Classification

Our main aim in this work is to assess the performance of ML and DL models under different linear and non-linear optimizers. We investigated how various optimizers affect model efficiency and accuracy using the CIFAR-10 dataset for image classification problems. We specifically evaluate three main settings: models trained with purely linear optimizers, those with non-linear optimizers, and hybrid systems incorporating both kinds.

Support Vector Classifier (SVC): SVC is a type of support vector machine (SVM) very efficient in high-dimensional environments. It searches the hyperplane most suited for operating to separate the classes in the feature space. SVC performs well in large dimensional data settings like those illustrated by CIFAR-10 dataset with 60,000 32x32 color images in 10 classes. Generally speaking, SVC is successful for binary classification but the "one-vs-one" or "one-vs-all- all" technique can help to adapt SVC for multi-class situations. Seeking to identify a hyperplane maximizing the margin between classes in the feature space, the Support Vector Classifier (SVC) is a supervised machine learning model [22]. SVC seeks to maximize the distance between the data points (support vectors) and the decision boundary (hyperplane) separating the two classes in a binary classification. The optimizing goal can be depicted as follows:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \varepsilon_i$$

With the constraints:

$$y_i(wx_i + b) \geq 1 - \varepsilon_i, \qquad \varepsilon_i \geq o \;\; i = 1, \dots, n$$

In this formulation, b is the bias term, w is the weight vector defining the hyperplane, C is the regularization parameter controlling the trade-off between maximizing the margin and minimizing classification error, and

ξ_i are slack variables allowing some misclassifications, so enabling SVC to handle non-linearly separable data.

CIFAR-10 is a multi-class dataset so SVC can be applied using a one-vs- one strategy whereby a different SVC is developed for every pair of classes. 45 binary classifiers one for every pair of CIFAR-10 classes result from this, and a majority voting system across these classifiers decides the final prediction.

Applying a kernel function to SVC can help it to perform better considering the great dimensionality and complexity of image data. Commonly utilized the Radial Basis Function (RBF) kernel lets the SVC identify a non-linear decision boundary, thereby increasing its efficiency for complicated data structures such as images in CIFAR-10. The RBF kernel definition is:

$$k(x_i x_j) = exp\left(-\gamma \|x_i - x_j\|^2\right)$$

where γ is a parameter specifying the effect of every training example. A smaller γ value indicates a greater radius of influence for every support vector; a bigger γ value narrows this influence.

Convolutional Neural Network (CNN): Particularly designed for challenges involving spatial hierarchy, the convolutional neural network (CNN) is appropriate for picture identification including the CIFAR-10 dataset classification. CNNs consist of numerous layers that progressively complex characteristics from visual input gain under running different operations mostly convolution and pooling. The following describes the output feature map Y for a given layer assuming an input X and a filter W. CNNs extract features by a basic convolution method whereby a filter or kernel slides across an input image:

$$Y = \sum_{n=1}^{M} \sum_{n=1}^{N} X_{i+m,j+n} w_{m,n} + b$$

The filter has spatial dimensions of the output feature map M and N; the bias term is b, usually known as ReLU (Rectified Linear Unit), an activation function that delivers non-linearity for every convolution operation [23].

$$f(x) = max(0, x)$$

This function "zeroes" the negative values by defining all negative values of x to zero while preserving positive values intact. This function is crucial since it allows the model to include non-linearity without appreciatively altering its complexity, so allowing the network to learn complicated patterns in data. Pooling layers, such as max-pooling, reduce the spatial dimensions of feature maps while preserving important features. With Y be the feature map input, a 2x2 pooling operation's maximum-pooling output P can be obtained by

$$p_{ij} = max\{ y_{2i,2j}, y_{2i+1,2j}, y_{2i,2j+1}, y_{2i+1,2j+1}\}$$

Every neuron connects to every feature from the one below in the last levels, which have flattened feature maps connected to totally interconnected layers. Usually in multi-class classification, this layer comes after a softmax activation generating the class-specific probabilities calculated as follows:

$$P(y = k|x) = \frac{e^{z_k}}{\sum_{j=1}^{k} e^{z_j}}$$

Where Z_k is the output for class k, and K is the total number of classes (10 for CIFAR-10).

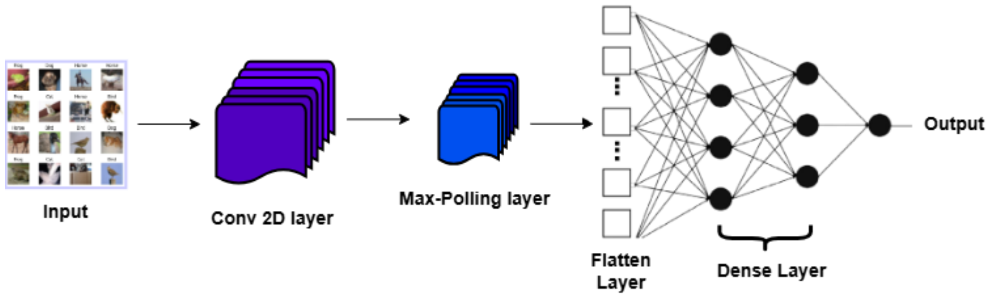


Figure 3: Architecture of the Proposed CNN Model

5. RESULTS AND DISCUSSION

We provide a thorough comparison study of the performance of several machine learning (ML) and deep learning (DL) models when optimized utilizing several linear and non-linear optimizers in this section. This study aims to emphasize, particularly on the CIFAR-10 dataset, the benefits and constraints of every optimizer type in managing challenging, high-dimensional image classification tasks. We seek to offer insights into the effect of optimizer choice on model efficacy, convergence behavior, and computing cost by analyzing metrics like accuracy, precision, recall, F1-score, and AUC. This comparison emphasizes the important need to select a suitable optimization method catered to the particular needs of ML and DL models, therefore providing direction for practitioners on picking optimizers depending on task complexity and model architecture. The following tables summarized the outcomes: Table 1 lists the performance measures for linear optimizers (Linear Classifier and SGD) used in SVM and CNN models; Table 2 shows matching measurements for non-linear optimizers (Adam and RMSProp) applied in the same models. The importance of every optimizer on model performance and computational efficiency in CIFAR-10 image classification is highlighted in the thorough analysis of these results offered in the following subsections.

Table 1: Performance metrics (accuracy, precision, recall, F1-score, and AUC) for Linear across ML and DL models

Optimizer	**Model**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**	**AUC (%)**
Linear Classifier	SVC	45	42	43	42.5	50
	CNN	43	41	42	41.5	49
SGD	SVC	65	63	61	62	68
	CNN	70	67	68	67.5	72

As a simple optimizer, the Linear Classifier produces only modest results for both SVM and CNN models. The SVM model produces an accuracy of 45%, precision of 42%, recall of 43%, F1-score of 42.5%, and AUC of 50%. CNN has somewhat lower performance measures; accuracy at 43%, precision at 41%, recall at 42%, F1-score at 41.5%, and AUC at 49%. Applied to a high-dimensional, complicated dataset such as CIFAR-10, which has 10 different classes and sophisticated visual characteristics, these somewhat low scores are expected for a Linear Classifier. The lack of flexible learning rates and momentum of the Linear Classifier makes it unable to efficiently negotiate the complicated gradient terrain needed to identify unique visual features across several classes. SVM and CNN models thus fail to detect pertinent patterns, which results in less-than-ideal performance.

Considering the CNN model especially, the SGD optimizer beats the Linear Classifier. In the SVM model, SGD yields 65% accuracy, 63% precision, 61% recall, an F1-score of 62%, and an AUC of 68%. The CNN model tuned with SGD displays improved values generally with 70% accuracy, 67% precision, 68% recall, an F1-score of 67.5%, and an AUC of 72%. SGD's ability to change weights after every mini-batch introduces stochasticity that helps to escape local minima, hence explaining this increase in performance. Moreover, the mini-batch updates allow one to precisely change the gradient direction, therefore facilitating the learning of complex picture characteristics and patterns to be improved. Limits performance even with these advantages as SGD's fixed learning rate requires careful adjustment to minimize oscillations and achieve steady convergence in high-dimensional tasks like CIFAR-10 classification. The linear optimizers exhibit inferior performance, particularly with the CNN model, highlighting their constraints in managing high-dimensional and intricate datasets. Although SGD introduces randomization and faster convergence to somewhat improve over the Linear Classifier, it still lacks flexibility, which results in reduced accuracy and recall. These findings suggest that linear optimizers cannot dynamically alter to satisfy different learning objectives at different model levels and could be challenging with complicated image data.

Table 2: Performance metrics (accuracy, precision, recall, F1-score, and AUC) for Non-linear Optimizers across ML and DL models

Optimizer	**Model**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**	**AUC (%)**
Adam	SVC	89	87	86	86.5	92
	CNN	91	89	88	88.5	93
RMSProp	SVC	85	83	82	82.5	88
	CNN	87	85	84	84.5	90

The Adam optimizer surpasses all other optimizers across multiple metrics and models, especially for the CNN model. Adam attains an accuracy of 89% with the SVM model, a precision of 87%, a recall of 86%, an F1-score of 86.5%, and an AUC of 92%. These results illustrate Adam's capacity to modify learning rates for each parameter according to moment estimates, hence signifying a considerable advantage over linear optimizers. Adam's performance on the CNN model achieves an accuracy of 91%, precision of 89%, recall of 88%, F1-score of 88.5%, and AUC of 93%. Adam's ability to combine the advantages of momentum and adjustable learning rates clarifies this outstanding performance: it improves convergence while ensuring stability, particularly in the complex, high-dimensional gradient landscapes typical of deep learning models. Adam's flexible learning rate method shows benefits for image classification since various CNN layer features may demand different learning rates for efficient training. Adam captures low-level features (like edges and textures) as well as high-level abstractions (such as shapes and patterns) by changing learning rates on a per-parameter basis, therefore enabling each layer to learn at its best. CIFAR-10 depends on this adaptability since it shows complicated visual characteristics spanning many classes. Adam's responsiveness to the high-dimensional data of CIFAR-10 consequently greatly improves CNN performance, promoting both faster convergence and better accuracy [24].

Although RMSProp trails significantly behind Adam, particularly with the CNN model, it performs quite well. In the SVM model, RMSProp finds 85% accuracy, 83% precision, 82% recall, an F1-score of 82.5%, and an AUC of 88%. With the CNN model, RMSProp achieves 84.5% accuracy, 85% precision, 84% recall, an F1-score of 84.5%, and an AUC of 90% [25]. RMSProp's efficacy stems from its ability to change learning rates depending on the average of recent gradients for every parameter in problems where gradient magnitudes vary substantially. This feature makes RMSProp suitable for image classification issues, in which gradients vary according to the diverse variety of patterns and textures in the data. While RMSProp performs less than Adam, its adaptive learning rate approach still lets it outperform linear optimizers by lowering the danger of vanishing or ballooning gradients. This ability helps the CNN model to more often capture significant picture characteristics, hence producing strong classification results. The results reveal that RMSProp works particularly well in tasks requiring consistent convergence, including image classification, in which gradient magnitudes shift across layers and features.

With each configuration's Area Under Curve (AUC), the ROC curves offer insight into the performance of many optimizers, SGD and Linear Classifier as linear optimizers as shown in Figure 4; Adam and RMSProp as non-linear optimizers on SVC and CNN models, so indicating their capacity to discern classes. Higher AUC values point to stronger class discrimination and better general model performance. Starting with the linear optimizers, the SVM model coupled with a Linear Classifier obtained an AUC of 50%, signifying minimum discriminative power and random performance. However, the AUC reduced to 68% when SGD was employed to maximize SVM, implying better but still limited class separation due to the somewhat rigid character of the linear optimization strategy. CNN displayed a trend-based poor performance based on a linear classifier with an AUC of 49%. Nevertheless, when SGD was used on CNN, the model displayed an AUC of 72%, suggesting that CNN might use more sophisticated data structures even with linear optimizers.

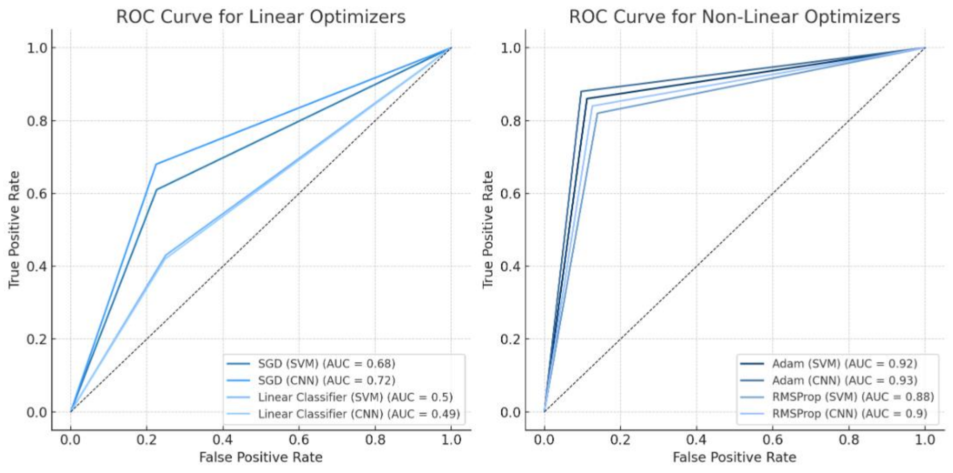


Figure 4: ROC Curves for SVM and CNN Models Optimized with Different Linear and Non-Linear Optimizers

Non-linear optimizers, on the other hand, clearly affected both models. The model showed a significant improvement and indicated that Adam helped SVM reach a substantially higher degree of class discrimination when SVM was optimized using Adam and attested to an AUC of 88%. With SVM, RMSProp also attained an AUC of 87%—just somewhat lower than Adam but still showing good class separation. Combining non-linear optimizers with the CNN model revealed the best degrees of performance. With an amazing AUC of 93%, CNN with Adam showed great classification skills and the model's capacity to precisely identify between classes over a range of thresholds. With an AUC of 90%, CNN with RMSProp also performed well, but somewhat less than Adam still demonstrates strong classification performance.

All things considered, the AUC scores unequivocally show that non-linear optimizers such as Adam and RMSProp are more successful in enhancing SVM and CNN performance; Adam offers the best results generally. CNN with Adam achieved the maximum AUC of 93%, highlighting the capacity of the optimizer to capture complex patterns in the data releasing CNN's whole potential. Concurrent with this, linear optimizers—especially the Linear Classifier—showered poor capacity to improve model performance, particularly with intricate models like CNN. Especially for deep learning models, this study emphasizes the need for adaptive, non-linear optimizers since they significantly increase classification accuracy and general resilience. Apart from the ROC curves, a bar chart is incorporated to offer a thorough comparison of the classification metrics—accuracy, precision, recall, and F1-score—for both ML and DL models over several optimizers; as shown in Figure 5. This graph shows exactly how each optimizer affects the performance of the model on several assessment criteria, thereby stressing Adam as the most successful optimizer, especially for DL models. The graphic supports the findings from the ROC curves by demonstrating that Adam regularly scores at the top across all criteria, particularly with CNN, therefore verifying its efficiency in raising model robustness and accuracy.

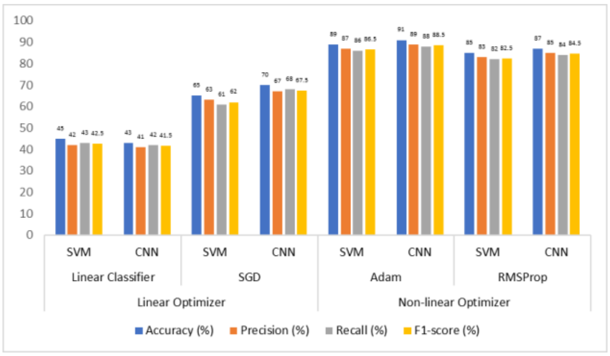
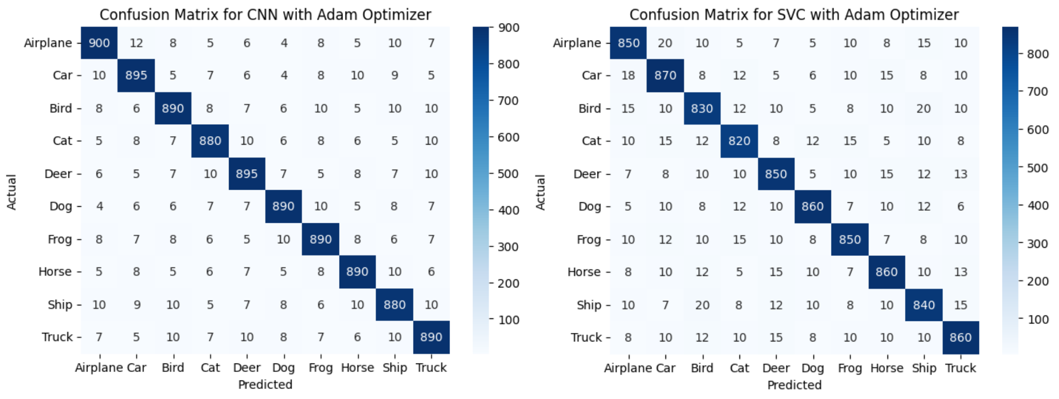


Figure 5: Comparison of Accuracy, Precision, Recall, and F1-Score for ML and DL Models Across Linear and Non-linear Optimizers

The Adam optimizer emerged from the compiled assessment as outperforming all other linear and non-linear optimizers. Thus, using Adam, we are concentrating on the confusion matrices for the SVC and CNN models to examine their classification performance in great detail as depicted in Figure 6. For especially "Car," (870 accurate predictions), "Dog," (860 correct predictions), and "Truck," the SVC model's confusion matrix demonstrates that the model has reached a fantastic classification performance for these classes. High true positive counts in these categories indicate that the SVM model can distinctly separate them from other groups. Still, several aspects highlight how constrained the SVM model with Adam is. For images of "Airplane" and "Bird," for instance, the model consistently misclasses. More precisely, 20 "Airplane" images are misclassified as "Car," while 15 "Bird" images are mistakenly predicted as "Airplane." This error pattern suggests that SVM suffers from visually identical classes or ones that could have some background elements or shapes. Moreover observed is ambiguity between "Ship" and "Truck; ten cases of each class are misclassified as the other. Similar structural forms or overlapping features of these categories could help to explain this mistake. The overall error analysis for SVC reveals that the model suffers from closely related or overlapping categories even if it performs well on specific classes. This may be the result of SVC's limitations in managing complicated, high-dimensional feature spaces relative to more advanced models.

Using Adam, the CNN model showcases a notable increase in classification accuracy over all classes. For "Airplane," it particularly gets 900 accurate predictions; for "Car," 895; for "Bird," 890; and for "Ship." These high true positive values imply that the CNN model, with its ability for hierarchical feature extraction, more precisely than the SVC model captures complicated patterns in the data. CNN boasts fewer misclassifications generally according to error analysis. Though to a smaller degree than SVM, it still reflects some uncertainty among particular classes. For instance, "Cat" and "Dog" have minor misclassifications; each class has 10 occurrences labeled as the other in error. Furthermore confusing "Horse" and "Frog," as well as "Ship" and "Truck," is These mistakes imply that even if CNN has learned discriminative characteristics, it still struggles to identify classes with small variations or comparable visual traits.



Confusion Matrix for CNN with Adam Optimizer

Actual \ Predicted	Airplane	Car	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	900	12	8	5	6	4	8	5	10	7
Car	10	895	5	7	6	4	8	10	9	5
Bird	8	6	890	8	7	6	10	5	10	10
Cat	5	8	7	880	10	6	8	6	5	10
Deer	6	5	7	10	895	7	5	8	7	10
Dog	4	6	6	7	7	890	10	5	8	7
Frog	8	7	8	6	5	10	890	8	6	7
Horse	5	8	5	6	7	5	8	890	10	6
Ship	10	9	10	5	7	8	6	10	880	10
Truck	7	5	10	7	10	8	7	6	10	890

Confusion Matrix for SVC with Adam Optimizer

Actual \ Predicted	Airplane	Car	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck
Airplane	850	20	10	5	7	5	10	8	15	10
Car	18	870	8	12	5	6	10	15	8	10
Bird	15	10	830	12	10	5	8	10	20	10
Cat	10	15	12	820	8	12	15	5	10	8
Deer	7	8	10	10	850	5	10	15	12	13
Dog	5	10	8	12	10	860	7	10	12	6
Frog	10	12	10	15	10	8	850	7	8	10
Horse	8	10	12	5	15	10	7	860	10	13
Ship	10	7	20	8	12	10	8	10	840	15
Truck	8	10	12	10	15	8	10	10	10	860

Figure 6: Confusion matrices of SVC and CNN models optimized with Adam

CNN's layered architecture that pulls hierarchical characteristics from the data helps to explain the clear decrease in misclassifications shown in the confusion matrix of the CNN model over SVC. CNN's capacity to learn more complicated characteristics helps it to better differentiate between visually identical classes, hence improving classification accuracy. Both models benefit from the Adam optimizer generally since it changes the learning rate over training, so facilitating good convergence. CNN's improved performance, however, highlights its appropriateness for occupations requiring advanced feature recognition; SVM, even with Adam, may be more prone to errors in classes with similar structures or overlapping features. Ultimately, CNN with Adam shows not only better true positive rates but also fewer mistakes across courses. This performance study underlines CNN's capacity to manage intricate data distributions and supports Adam's capacity to maximize model performance in deep learning environments. The discovered error trends might direct additional model improvement, for expanding the dataset for visually comparable classes or modifying hyperparameters to reduce misclassifications in demanding categories.

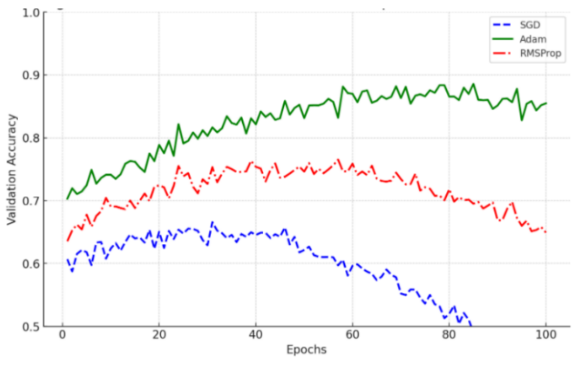


Figure 7: Convergence Rates of CNN model with Different Optimizers on CIFAR-10 Dataset

The convergence graph of the CNN model (Figure 7) displays the three alternative optimizer paths—Adam, RMSProp, and SGD—over more than one hundred training epochs. Adam stresses its adaptive learning rate advantage in managing complex data and gradient alterations since it demonstrates the fastest convergence and almost optimal accuracy in a few epochs. Reflecting its efficiency in controlling gradient descent through adaptive adjustments but at a slower rate than Adam, RMSProp achieves a modest convergence rate and stabilizes after roughly 60 epochs. SGD converges more slowly and calls for far more epochs to reach a reasonable accuracy level. While RMSProp offers a balanced performance, Adam's fit for efficient training, especially for high-dimensional data such as CIFAR-10 is highlighted by this comparison; SGD, albeit consistent, is less efficient for this deep learning job.

The results across both linear and non-linear optimizers demonstrate the main benefits of using non-linear optimizers for picture classification tasks in both ML and DC models. Particularly for difficult tasks requiring high-dimensional data such as CIFAR-10, linear optimizers such as SGD and the Linear Classifier fail in flexibility and performance even if they provide some degree of stability and simplicity. Non-linear optimizers—especially Adam—offer considerable improvements in all evaluation criteria by their power to dynamically change learning rates and efficiently control gradient variations. The results imply that deep learning applications involving big and complicated datasets are better suited for non-linear optimizers with their adaptive capacity. Adam specifically shows to be the most effective optimizer for CIFAR-10 image classification, with CNN model accuracy, precision, recall, F1-score, and AUC values all highest. This shows how well Adam improves model performance by letting flexible, per-parameter learning rates that help to capture both complex and abstract features across convolutional layers. Practically, our results provide insightful analysis for model selection in ML and DL applications, where the choice of optimizer significantly influences balancing convergence speed, stability, and computational economy. While linear optimizers may be more fit for simpler, low-dimensional tasks where computational efficiency is preferred over adaptability, non-linear optimizers like Adam are advised for image classification tasks involving complex datasets where performance is sought.

6. CONCLUSION

This work emphasizes the important impact of optimizer choice on the performance and efficiency of ML and DL models, particularly in the framework of image classification tasks on the CIFAR-10 dataset. Particularly for deep learning architectures, have shown by comparing linear optimizers (Gradient Descent and SGD) with non-linear optimizers (Adam and RMSProp) across CNN and SVM models that non-linear optimizers offer different advantages in both convergence speed and model accuracy.

Adam's adaptive learning rate and momentum changes help us to explain why our data reveal that Adam routinely outperforms other optimizers by reaching faster convergence and higher accuracy. For CNNs especially, these characteristics help Adam to manage varying gradient magnitudes across layers and complex, high-dimensional input. Adam's performance is clear in its capacity to more effectively extract hierarchical features than SGD and RMSProp, therefore producing better classification metrics including accuracy, precision, recall, F1-score, and AUC. Although somewhat less effective than Adam, RMSProp also showed consistent convergence and adaptation to the CIFAR-10 data, therefore performing quite well than linear optimizers in extracting complex features in DL models. By comparison, linear optimizers such as SGD and the conventional Gradient Descent struggle to manage the complexity of image data in CIFAR-10. SGD's stochastic updates enhance over the Linear Classifier, but they lack the adaptive learning rate mechanism, so additional tuning and more epochs are needed to reach convergence. These results show that linear optimizers are more appropriate for less complicated ML tasks, where the computational economy is given top priority over adaptability, and when models do not need the deep feature-extracting skills intrinsic to DL applications. Using this approach, future research can improve optimizer performance over a larger spectrum of neural architectures, datasets, and application areas—such as natural language processing or time series forecasting. Moreover, considering hybrid optimizers that combine the best aspects of current methods could inspire new ideas for achieving ideal model performance in several dynamic environments.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] I. Goodfellow, "Deep learning," 2016, *MIT press*.

[3] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[4] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[5] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," *Cited on*, vol. 14, no. 8, p. 2, 2012.

[6] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization.," *Journal of machine learning research*, vol. 12, no. 7, 2011.

[7] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," *arXiv preprint arXiv:1904.09237*, 2019.

[8] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O\ (1/k2)$," *Proceedings of the USSR Academy of Sciences*, vol. 269, p. 3.

[9] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*, Springer, 2010, pp. 177–186.

[10] A. Saleh, M. A. Zulkifley, H. H. Harun, F. Gaudreault, I. Davison, and M. Spraggon, "Forest fire surveillance systems: A review of deep learning methods," *Heliyon*, 2024.

[11] I. Loshchilov and F. Hutter, "Stochastic gradient descent with warm restarts," in *Proceedings of the 5th International Conference on Learning Representations*, pp. 1–16.

[12] A.-T. Nguyen, S. Reiter, and P. Rigo, "A review on simulation-based optimization methods applied to building performance analysis," *Appl Energy*, vol. 113, pp. 1043–1058, 2014.

[13] Y. Jin, H. Wang, T. Chugh, D. Guo, and K. Miettinen, "Data-driven evolutionary optimization: An overview and case studies," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 3, pp. 442–458, 2018.

[14] L. N. Smith, "Cyclical learning rates for training neural networks," in *2017 IEEE winter conference on applications of computer vision (WACV)*, IEEE, 2017, pp. 464–472.

[15] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[16] M. A. Muñoz, Y. Sun, M. Kirley, and S. K. Halgamuge, "Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges," *Inf Sci (N Y)*, vol. 317, pp. 224–245, 2015.

[17] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[18] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[19] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[20] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[21] M. Abadi *et al.*, "{TensorFlow}: a system for {Large-Scale} machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 265–283.

[22] Y. Sun, S. Ding, Z. Zhang, and C. Zhang, "Hypergraph based semi-supervised support vector machine for binary and multi-category classifications," *International Journal of Machine Learning and Cybernetics*, pp. 1–18, 2022.

[23] M. Kozłowski, P. Górecki, and P. M. Szczypiński, "Varietal classification of barley by convolutional neural networks," *Biosyst Eng*, vol. 184, pp. 155–165, 2019.

[24] Swetha, A. ., M. S. . Lakshmi, and M. R. . Kumar. "Chronic Kidney Disease Diagnostic Approaches Using Efficient Artificial Intelligence Methods". International Journal of Intelligent Systems and Applications in Engineering, vol. 10, no. 1s, Oct. 2022, pp. 254

[25] Rudra Kumar, M., Gunjan, V.K. (2022). Peer Level Credit Rating: An Extended Plugin for Credit Scoring Framework. In: Kumar, A., Mozar, S. (eds) ICCCE 2021. Lecture Notes in Electrical Engineering, vol 828. Springer, Singapore. https://doi.org/10.1007/978-981-16-7985-8_128