Thesis for the Degree of Master of Science in Computer Engineering

Performance Analysis of Adam Optimization on Introducing Energy Index

A thesis submitted in partial fulfillment of the requirement for the degree of Master of Science In Computer Engineering

Mohan Bhandari

2017-1-39-0011

Nepal College of Information Technology

Faculty of Science and Technology

Pokhara University

July 3,2021

Thesis for the Degree of Master of Science in Computer Engineering

Performance Analysis of Adam Optimization on Introducing Energy Index

Supervised By Dr. Pramod Parajuli

A thesis submitted in partial fulfillment of the requirement for the degree of Master of Science In Computer Engineering

Mohan Bhandari
2017-1-39-0011

Nepal College of Information Technology
Faculty of Science and Technology
Pokhara University

July 3 2021

DECLARATION

This thesis, named ”PERFORMANCE ANALYSIS OF ADAM OPTIMIZATION ON INTRODUCING ENERGY INDEX,” is based on my own research. Other researchers' related work on this issue has been recognized. I am responsible for all liabilities pertaining to the data correctness and validity, as well as any other material contained herein

Mohan Bhandari
2017-1-39-0011
July 3 2021

RECOMMENDATION

This is to certify that this thesis submitted by MOHAN BHANDARI under the title ”PERFORMANCE ANALYSIS OF ADAM OPTIMIZATION ON INTRODUCING ENERGY INDEX,” was prepared under my supervision in partial fulfillment of the requirements for the degree of Master of Science (M. Sc.) in Computer Engineering awarded by Pokhara University.
I recommend the same for Pokhara University’s admittance.

Pramod Parajuli, Ph.D
Principal, CG Institute of Management
Chief Innovation Officer, Nepal Payment
July 3 2021

CERTIFICATE

This thesis entitled ”PERFORMANCE ANALYSIS OF ADAM OPTIMIZATION ON INTRODUCING ENERGY INDEX”, prepared and submitted by MOHAN BHANDARI, has been reviewed and is certified for the award of the degree of Master of Science in Computer Engineering by Pokhara University.

Prof. Dr. Purushottam Kharel
External Examiner — Signature — Date
Nepal College of Information and Technology

Suresh Pokhrel
External Examiner — Signature — Date
Nepal College of Information and Technology

Dr. Pramod Parajuli
Supervisor — Signature — Date
Principal,CG Institute of Management

Assoc. Prof. Saroj Shakya
Head Of Masters Department — Signature — Date
Nepal College of Information and Technology

Acknowledgment

I express my gratitude towards Department of Computer Engineering at Nepal College of Information Technology for accepting my thesis, "PERFORMANCE ANALYSIS OF ADAM OPTIMIZATION ON INTRODUCING ENERGY INDEX". I am grateful to the assistance provided by the department with all helpful ideas in selecting this thesis topic.

My supervisor, Dr. Pramod Parajuli, has been really helpful in providing me with information and direction on my thesis. He continuously let me do my own work on this assignment, but led me in the proper way when he believed I needed it.

I'm also grateful to Mr. Saroj Shakya, our program coordinator, for coordinating and supporting this thesis.

I would also want to convey my deep gratitude to respected professors, my families, and friends who have directly and indirectly assisted and supported me during the thesis process.

Mohan Bhandari
2017-1-39-0011
July 3 2021

Abstract

The adjustment of learning rate(η), bias and different other parameters during back propagation are crucial for the performance of machine learning based algorithms. In support for better performance, by utilizing the exponential decay of past gradients and their squares, Adam Optimization technique tries to tune different learning parameters. Still, the algorithm lacks involvement and calculations of frequently occurring features in individual neurons of the network which play significant role in adjusting the learning rate for the performance improvement of the algorithms.

The study focuses on Energy Index Based Optimization Method which introduces the significance of frequently occurring features in Adam Optimization. Energy model of neuron is designed to calculate the energy index being based on the frequency of feature and Energy Index is calculated. Energy Index was taken into account along with Adam Optimizer. The performance of energy introduced Adam optimizer was experimented on Logistic Regression, Multi-Layer Perceptron and Support Vector Machine.

Keywords : *Machine Learning Algorithms, Stochastic Gradient Descent, Learning Rate, Adaptive Moment Estimation, Energy Index, Logistic Regression, Convolution Neural Network, Multilayer Perceptron, Support Vector Machine*

Contents

List of Figures

List of Tables

List of Abbreviations

ADAM	Adaptive Moment Estimation
ANN	Adaptive Moment Estimation
AUC	Area Under Curve
BoW	Bag of Words
CIFAR	Canadian Institute For Advanced Research
CNN	Convolution Neural Network
EI	Energy Index
FMNIST	Fashion Modified National Institute of Standard and Technology
GD	Gradient Descent
LR	Logistic Regression
MLP	Multi-Layer Perception
MNIST	Modified National Institute of Standard and Technology
NIST	National Institute of Standard and Technology
RMS	Root Mean Square
RMSProp	Root Mean Square Propagation
RProp	Resilient Propagation
RSES	Rough Set Exploration System
ROC	Receiver Operator Characteristic

SGD	Stochastic Gradient Descent
SVM	Support Vector Machine

Chapter 1

Introduction

1.1 Background

Inspired from the way that biological nervous system process information,Artificial Neural Network is neural structure that constitutes highly organized processing nodes to work together called neurons to solve a specific problem. Using mathematical or computational model for processing information, ANNs are taken under consideration for applications like image recognition, object detection, face recognition, speech recognition, data classification and other aspects, neural networks learn by example. Almost all ANNs have some sort of learning rule that adjusts the weights of nodes in neural networks based on input patterns that are calculated after learning from a supervised process that occurs with each epoch over a forward activation flow of outputs and adjusting weights with back-propagation.

With the help of initial inputs provided, a neural network makes an arbitrary 'guess' about what the input might be and finally compares it's guess with the real input and concludes the differences i.e. loss function. Fundamentally, a loss function is a performance metric on how sound the Neural Network succeeds to grasp its goal of generating outputs, likely to the desired values. Hence, on support of loss function, the network attempts to make fitting adjustment to its weights. During this process, different activation functions help the network to polarize the activities for stability. Further, we can take support from the different optimization techniques to modify the weights so as to minimize the total loss function. While training a Neural Network over a M x N array of inputs, simple brute force is not imaginable for optimization. As a consequence, differentiation is introduced to deal with loss function as loss function gives the rate of loss in while learning. In simple words, positive derivative means the error increases if we increase the weight, negative derivative means the error decreases if we increase the weights and if it is zero, we reached our stable point. At this point, random initialization of weight does not affect as the error function curve always drags to minimum. Beside these all, there are different methods for updating weights, commonly called optimizers. The overall process is iterated until the convergence.

Figure 1.1 shows the use of optimizer in Neural Network. In Neural Network, optimization means to find the alternative with low cost function by increasing the required factors and decreasing the undesired ones under given constraints. Maximizing, here, means attempting to attain stable convergence.

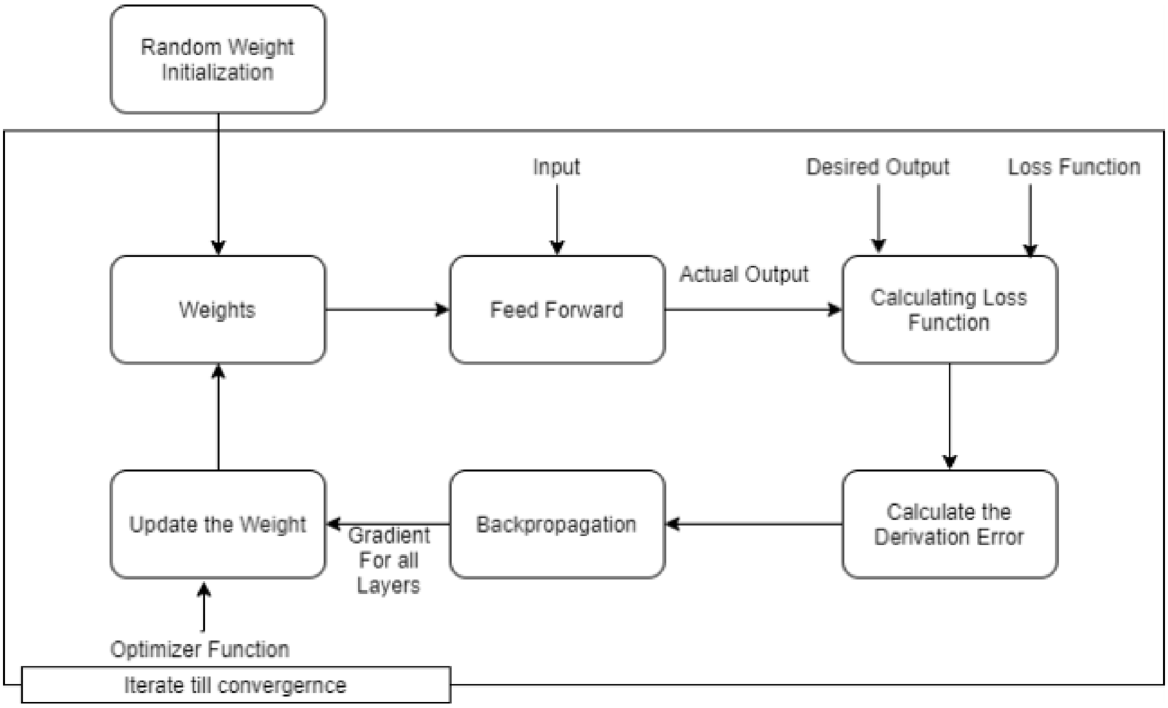


Figure 1.1: Use of Optimizers in Neural Network

During training, network adjusts parameters and weights of model and try to minimize the loss function and make the estimates as correct as possible. The loss function guides the optimizer for the right direction.

The first order optimization algorithm called gradient descent is used to find local minimum of a function, steps proportional to the negative of gradient is taken at a particular point. Gradient ascending occurs when steps are proportional to the positive of gradient.

Different types of descent algorithm used on the basis of gradient are:

(i) Batch Gradient Descent

(ii) Stochastic Gradient Descent

(iii) Mini Batch Gradient Descent

Designing and training neural network models is an enhancement issue because one should consider that to improve the objective function, boundary value should be considered. As of now, as the most mainstream strategy, stochastic angle plummet is moderately proficient and powerful since the calculation of first-request fractional subordinates regarding every one of the boundaries has a similar computational intricacy as assessing the capacity. In any case, back-propagation with gradient descents requires some manual change in default learning rate. Learning rate is reduced if proper selection of learning rate is not done. In this manner, tuning the learning rate and tracking down an ideal learning rate consequently are significant difficulties in the preparation for SGD [1].

In addition to adapting the learning algorithm to the training dataset, optimization is an important element of a machine learning project. The process of preparing data before fitting a model and adjusting a model that has been chosen may both be viewed as optimization problems. In fact, a predictive modeling project as a whole may be viewed as a big optimization issue.

The author in [2] introduced a technique for proficient stochastic improvement that utilizes simply first order gradients with low memory. Individual learning rates for different parameters are determined utilizing assessments of the first and second moments of the gradients.

Here, the study is focused on improving the performance of adam optimization.

1.2 Problem Statement

Using Gradient Descent (GD) as an optimization in machine learning algorithms requires some adjustment in initial values of different machine parameters like learning rate and inappropriate adjustment which will minimize the convergence or may even make training algorithm to diverge.

Adam Optimization, one of the variants of Gradient Descent, is most common algorithm to perform optimization, considers the exponential decaying average of past gradient and its square with the help of hyper parameters.

Features occurring frequently have significant effect to adjust the learning rate of the algorithms during back propagation. Adam considers this fact on layers but not for individual neurons.

Hence, We can increase the performance of Adam if we consider those features on every single neuron in a network.

1.3 Research Objective

Though Adam solves the problem of exponential decay of past gradients and the square gradients by adjusting the learning rate only in layers, the optimizer do not consider the significance of frequently occurring features in each neuron.

Considering this fact, the performance of this optimizer can be improved on introducing the Energy Index.

Hence, the goal of this research is to :

(i) Introduce the Energy Index in Adam Optimization to tune the learning rate in individual neuron of every layer of neural network and analyze the performance in different machine learning models

1.4 Significance of Study

Experiments are concentrated to analyze the performance of ADAM optimizer on introducing the Energy Index. The newly introduced behavior of ADAM optimizer is implemented in different variants of neural networks like Logistic Regression (Single Neuron), Multilayer Perceptron (Multiple Neurons in different layers) and Support Vector Machine (Support Vector Based Classification). For all the variants, sigmoid function is used and every task is done from scratch.

Different standard datasets like CIFAR10, MNIST and Fashion MNIST are used for analyzing the result of introducing the calculation of recurrent feature calculation during learning and different parameters like cost, test error, training error and confusion matrix evaluate the performance.

1.5 Limitations of Study

During the study, ‘Adam with Energy Index’ model was tested in three different algorithms (Logistic Regression, Multilayer Perceptron and Supoort Vector Machine) each with three different datasets, but the study was not focus on other datasets and algorithms.

Being specific to MLP, Experiments were taken under a single hidden layer, the study is unknown about the results for multiple hidden layers.

Further, all the experiments were done for image classifications.The performance of ‘ADAM with EI’ is not studied in other algorithm of Machine Learning trends like text classification, speech recognition etc.

Chapter 2

Literature Review

2.1 Background Study

2.1.1 Neural Network

Artificial Neural Networks (ANNs) are a data-processing paradigm inspired by the way biological nerve systems process the information. They are biologically inspired connection models that take in data, transform it through a succession of hidden layers, and then calculate the result. In regular NN, each neuron in every layer is coupled to all neurons in preceding layers. Figure 2.1 shows a general structure of biological neural network. ANN comprises of large number of nodes, that work to impersonate natural neurons of human cerebrum. For any problem, these neurons connect with one another by means of a connection to take input information and perform straightforward procedure to generate the result. The consequence of these

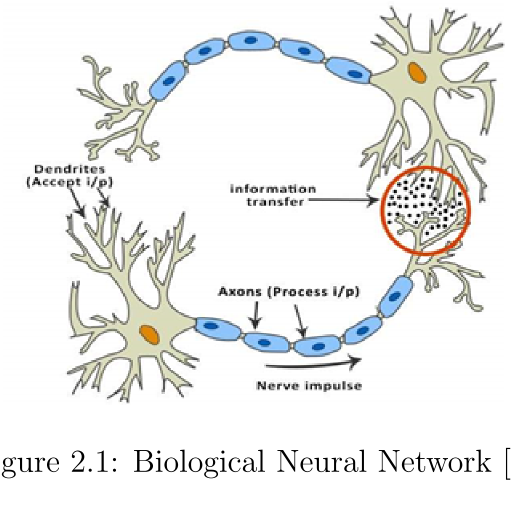


Figure 2.1: Biological Neural Network [3]

tasks is then passed to next layers of neurons. The yield at every node is called its activation. Each connection is related with weight. ANNs are fit for learning, which happens by changing weights during back propagation[3]. Figure 2.2 shows a simple ANN.

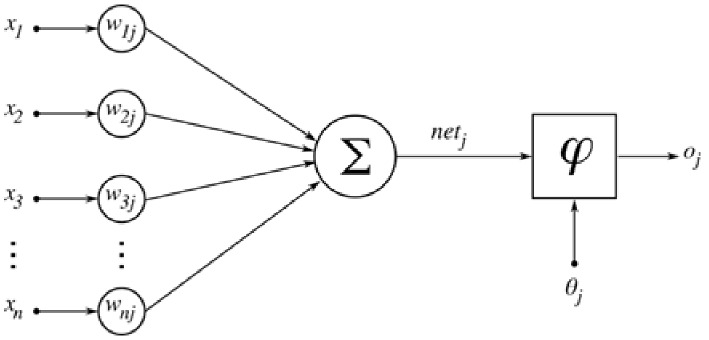


Figure 2.2: Artificial Neural Network [3]

2.1.2 Sigmoid Activation Function

A mathematical function with a characteristic "S"-shaped curve, also known as a sigmoid curve, is sigmoid function. As sigmoid function translate the entire number line into 0 and 1, or - 1 and 1, one of its use is to convert an actual value into one that may be analyzed as a probability.

As sigmoid function converges faster than any other activation function, for probabilistic classification, it was selected as activation function in the all the algorithms used. As standard choice for a sigmoid function is logistic function. Overall , the sigmoid function is defined as[4]:

$$S(x) = \frac{1}{1+e^{-z}} = \frac{e^x}{e^x+1} \tag{2.1}$$

2.1.3 Logistic Regression

Under the Supervised Learning approach, One of the most prominent Machine Learning algorithms is Logistic Regression. It's a technique for predicting a categorical dependent variable from a collection of independent factors. Logistic regression is used to predict the output of a categorical dependent variable. As a result, the outcome must be either discrete or categorical. It produces probabilistic values between 0 and 1, and instead of fitting a regression line, we use logistic regression to construct a "S" shaped logistic function, which predicts two maximum values (0 or 1)[5].

Logistic regression is employed in this study is to categorize observations based on a variety of data sources and to swiftly determine the most effective factors for classification. Figure 2.3 depicts the logistic function.

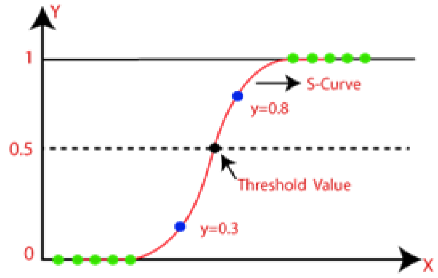


Figure 2.3: Logistic Regression[5]

(a) Decision Boundary

To forecast which class a piece of data belongs to, a threshold might be set. Based on this threshold, the calculated probability is divided into groups.

if $(predicted\_value)$ >0.5

classify email as spam else as not spam.

(b) Cost Function

A cost function is a metric for comparing the performance of a neural network to the training sample and expected output. Factors such as weights and biases may also have an impact. A cost function is a single number rather than a vector since it evaluates how well the neural network performed as a whole. In logistic regression, cost function is defined as[4]:

$$cost = \begin{cases} -\log h(x) & ify = 1 \\ -\log(1 - h(x)) & ify = 0 \end{cases} \tag{2.2}$$

2.1.4 Multi Layer Perceptron

The simplest basic feed-forward network is the multi layer perceptron (MLP). The units are organized into levels, each of which has a specific number of identical units. Every unit in one layer must be connected to every unit in the next tier for the network to be complete. The input layer is the initial layer, and its units are the input feature values.

The output layer is the final layer, and it contains one unit for each value that the network produces.

Because we don't know what these units can compute ahead of time and must figure it out while learning, all of the levels in between are known as hidden layers. The units in these levels are input units, output units, and hidden units, as depicted in Figure 2.4. The number of layers is represented by the depth, and the number of units in each layer is represented by the breadth.

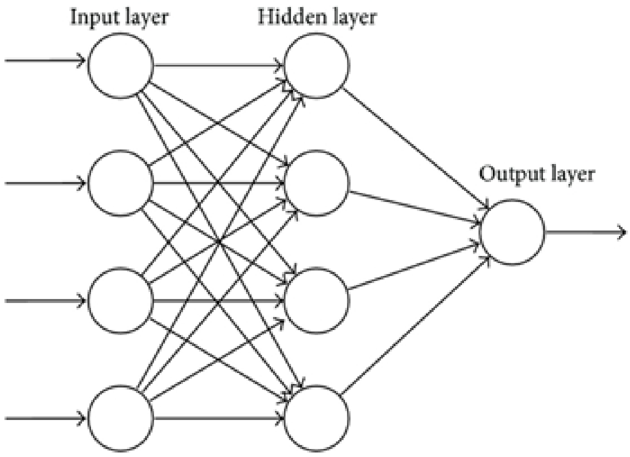


Figure 2.4: Multilayer Perceptron [4]

Currently in this study, 3072 input layers, 64 hidden layers and 1 output layer is defined.

2.1.5 Support Vector Machine

Support Vector Machines (SVMs), also called support vector network [6] are supervised learning models that evaluate data for classification and regression analysis. They come with related learning algorithms.

A support-vector machine creates a hyperplane in a high-dimensional space that may be used for classification and regression. The hyperplane with largest distance to the nearest training-data point of each class (known as functional margin) produces good separation intuitively, since the larger the margin, lower the classifier's generalization error.

As, result of SVM depends on kernel function used, the main problem in SVM is to find the best kernel function for data classification.

The initial purpose of linear support vector machines was to classify binary data. There are n sets of training datasets. [7],

$$(x_1, y_1), ..., (x_n, y_n) \tag{2.3}$$

where the $\mathrm{y_i}$ is either 1 or -1, indicating the class to which $\mathrm{x_i}$ belongs to. Each $\mathrm{x_i}$ is a p-dimensional vector. We have to find now the ”maximum-margin hyperplane” to dividethe group of $\mathrm{x_i}$ for which $\mathrm{y_i}$=1 from the group of point which $\mathrm{y_i}$= -1. We can write the hyperplane as:

$$W^T x - b = 1 \tag{2.4}$$

$$W^T x - b = -1 \tag{2.5}$$

Using the so-called one-vs-rest technique is the easiest way to expand SVMs for multiclass problems[6]. The negative instances will be data from the other classes, and K linear SVMs will be trained separately for K class problems. The objectives of Softmax and multiclass SVMs are the same, and they are parametrized by all of the weight matrices W. The Softmax layer lowers cross-entropy or maximizes log-likelihood, whereas SVMs simply seek to find the largest margin between data points of distinct classes.

The output of k-th SVM is denoted as:

$$a_k(x) = W^T x \tag{2.6}$$

The predicted class is:

$$arg_k max a_k(x) \tag{2.7}$$

2.1.6 Adaptive Moment Estimation

Instead of using standard stochastic gradient descent technique, Adam optimization algorithm is used to repeatedly update network weights depending on training data. It also keeps track of the moving average of the gradient’s first and second moments. For all weight updates, stochastic gradient descent retains a single learning rate, which does not change during training.

Adam, according to the authors [2], is a stochastic gradient descent extension that combines the advantages of two existing stochastic gradient descent extensions.

(a) Adaptive Gradient Algorithm

By maintaining a per-parameter learning rate, the Adaptive Gradient Algorithm (AdaGrad) improves performance on problems with sparse gradients.

(b) Root Mean Square Propagation

Root Mean Square Propagation (RMSProp) additionally keeps per-parameter learning rates that are modified according to the weights' average recent gradient magnitudes.

The basic algorithmic steps of ADAM optimizer are [2]:

Step 1: Initialize first moment vector (m=0), second moment vector(v=0) and hyper-parameters (β_1=0.9 and β_2=0.999)

Step 2: For every iteration (t)

(a) Get the gradients with respect to t

$$g_t = grad(\theta_{t-1}) \tag{2.8}$$

(b) Update first moment($\mathrm{m_t}$)

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)m_t \tag{2.9}$$

(c) Update second moment($\mathrm{v_t}$)

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)_t^2 \tag{2.10}$$

(d) Computer bias corrected first moment

$$\widehat{m} = \frac{m_t}{1-\beta_1^t} \tag{2.11}$$

(e) Compute bias corrected second moment

$$\widehat{v} = \frac{v_t}{1-\beta_2^t} \tag{2.12}$$

(f) Update Parameters

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\widehat{v}_{t-1}}+\epsilon}\widehat{m}_{t-1} \tag{2.13}$$

2.1.7 Energy Index

Inputs, outputs, activation function, weights, and biases are the essential components of a neuron in a layer of an artificial network, all of which must be learnt and are collectively known to as parameters. Figure 2.5 with 6-tuples depicts the energy neuron model in the neural network based on these values. Figure 2.5 shows an energy neuron model and can be defined by a 6-tuple[1].

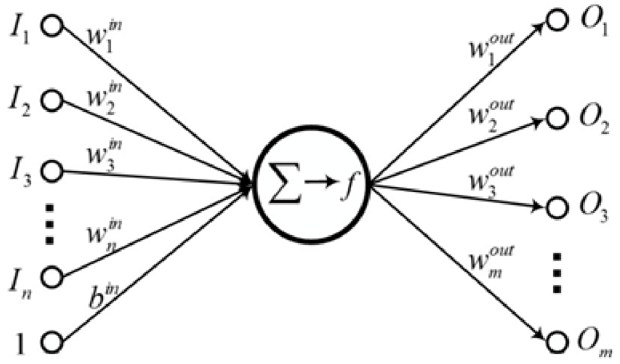


Figure 2.5: The Structure of Energy Neuron Model[1]

$$(I\ ,\ O,\ P,\ f,\ E_{\text{in}},\ E_{\text{out}})$$

where,

$I = (I_1, I_1, \ldots\ldots I_n)$: input vector for the particular neuron

O : output of the neuron

P : parameter to be learned by the network (Biases and Weights)

f : Activation Function

E_{in} : input energy for the particular neuron

E_{out} : output energy for the particular neuron

2.2 Related Works

Since the performance of SGD depends on the careful choice of the learning rate, many scholars have conducted several researches on selection of the learning rate. Authors in [8] presented **AdaGrad** adaptive learning approach, which adjusts the learning rate according to the parameters, with greater updates for infrequent parameters and lower updates for frequent values. AdaGrad is built on recent advances in stochastic optimization and online learning, both of which employ proximal functions to control the gradient measures of the method.

The findings indicated that adaptively changing the proximal feature makes determining a learning rate much easier, and that the outcomes are almost as good as the optimal proximal function that can be chosen in retrospect.

For every parameter θ_{i} in each iteration t during learning, we have:

$$\theta_{t,i} = \theta_{t-1,i} - \frac{\eta}{\sqrt{g_{t-1} + \epsilon}} g_{t-1,i} \tag{2.14}$$

where ϵ is smoothing to avoid division by zero and $\mathrm{g}_{\mathrm{t-1}}$ is objective function's gradient with respect to the parameter $\theta_{\mathrm{t-1}}$ at iteration t-1. Being based on past gradients for θ, Adagrad modifies the general learning rate η at each time step t for every parameter θ_{i}

The ImageNet image database [9], the Reuters RCV1 text classification [9], the MNIST handwriting datasets were used as the dataset by the authors. According to the authors' experiment. The AdaGrad family of algorithms incorporates regularization by default and produces very sparse solutions that perform similarly to dense solutions. In the adaptive methods experiments, a diagonal approximation to the matrix was obtained by taking the outer products of sub gradients computed along the algorithm's run.

As rare words require significantly bigger updates than frequent words,authors in [10] utilized AdaGrad to train GloVe word embeddings.

With innovations like Diagonal Adaptation and Full Matrix Implementation, Adagrad eliminates the need to manually manage the learning rate; nevertheless, the accumulation of squared gradients in the denominator leads the learning rate to drop, resulting in a sluggish convergence speed. The benefit of AdaGrad is that it eliminates the need to modify the learning rate manually, being 0.01 as common. The accumulation of squared gradients in the denominator is its biggest flaw. The cumulative total grows through training when each added term is positive, causing the learning rate to shrink and become infinitesimally small.

In [11], author has addressed an Adagrad modification that aims to slow down the program's aggressive, monotonically declining learning pace. The concept presented in the paper was extracted from [8] in order to address the method's two major flaws:

(a) The need for a manually selected global learning rate

(b) The continual decay of learning rates during training

Rather than collecting all previous squared gradients, **AdaDelta** restricts the window of cumulative past gradients to a predetermined size 'w'. Rather than keeping 'w' prior squared gradients inefficiently, the number of gradients is recursively determined as a declining average of all past squared gradients.. The update of AdaDelta is:

$$\theta_t = \theta_{t-1} - \frac{RMS[\Delta\theta]_{t-2}}{RMS[g]_{t-1}} g_{t-1} \tag{2.15}$$

Where $\mathrm{RMS[g]}_{\mathrm{t\text{-}1}}$ is the RMS error criterion of the gradient $_{\mathrm{t\text{-}1}}$
Beyond the standard stochastic gradient descent, the approach dynamically adapts over time using only first-order knowledge and has a low computational overhead. The method appears to be robust to noisy gradient knowledge, various model architecture choices, various data modalities, and hyperparameter selection, and does not require manual tuning of a learning rate.
Authors experimented the optimizer in MNIST handwriting, Throughout the training collection, the system was trained on mini-batches of 100 images per batch for 6 epoch As compared to SGD, the approach had a low computational overhead and had a per-dimension learning rate. Despite a wide range of input data formats, hidden unit numbers, nonlinearities, and distributed replica numbers, the hyperparameters do not need to be modified, indicating that AdaDelta is a resilient learning rate system that can be utilized in a variety of circumstances.
The weighted average would rise at a slower pace due to the limiting term, causing the learning rate to slowly decrease to reach the global minima.

Authors in Coursera slide 29, Lecture 6 [12], have stated that **RMSProp** is a mini-batch version of RProp, which is equivalent to using the gradient but dividing by the size of the gradient. RMSProp algorithm makes mini-batch learning more efficient The learning rate in RMSProp adjusts based on a moving average of the magnitudes of latest past gradients. RMSProp, in other words, keeps a moving average of the squares of recent gradients, denoted by (v). As a result, recent gradients are given more weight. The word Beta (β) is used to refer to the forgetting factor (just like in SGD with Momentum).

$$v_t = \beta V_{t-1} + (-\beta) \cdot [grad(\theta)_{t-1}]^2 \tag{2.16}$$

Standing for Resilient Propagation, RProp only uses the signs of gradients to compute the updates[13]. The motivation is that the magnitude of gradients varies by weight and changes over time and making difficult to choose a single learning rate. This is addressed by RMSProp by holding a moving average of the squared gradient and changing the weight updates by this magnitude. The gradient changes are carried out as follows:

$$E[G^2]_t = 0.9[g^2]_{t-1} + 0.1g^2_{t-1} \tag{2.17}$$

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{E[g^2_{t-1}] + \epsilon}} g_{t-1} \tag{2.18}$$

Another technique given by authors in [2] is to compute adaptive learning rates for each parameter. The method proposed by authors named **Adam** retained an exponentially decaying average of previous gradients $\mathrm{m_t}$, comparable to momentum, in addition to an exponentially decaying average of past squared gradients $\mathrm{v_t}$, as in Adadelta and RMSProp. We compute the decaying averages of past and past squared gradients $\mathrm{m_t}$ and $\mathrm{v_t}$respectively as Equation 2.9 and Equation 2.10. Initializing $\mathrm{m_t}$ and $\mathrm{v_t}$ as vectors, authors observed that they were biased towards zero, basically during the initial steps and during the decay rates being small (i.e. $\beta 1$ and $\beta 2$ are close to 1).

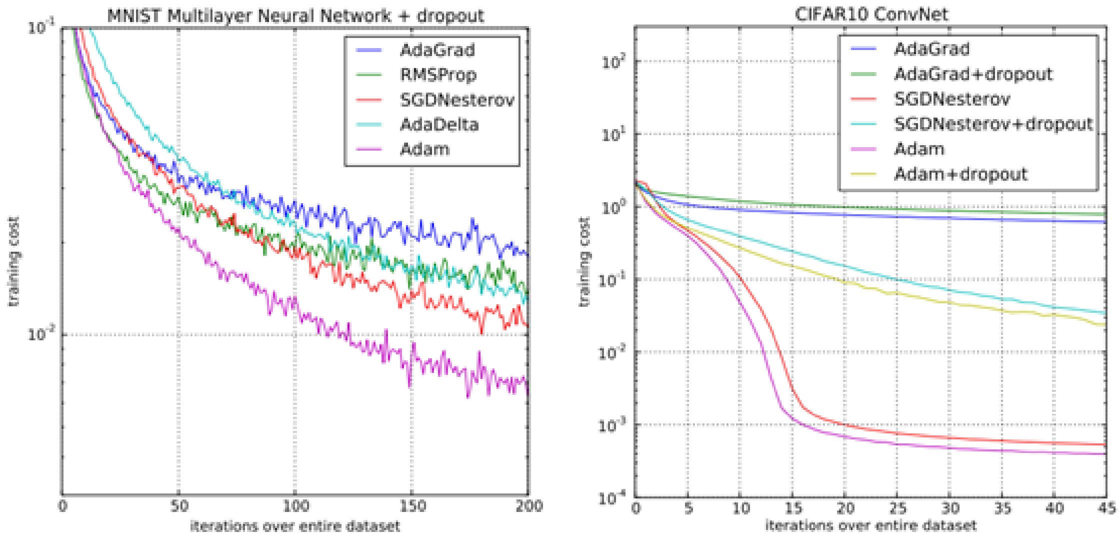
Finally the parameters are updated as in Equation 2.13.

The authors experimented the algorithm in Logistic Regression for IMDB movie review dataset [2]. Bag-of-words (BoW) feature vectors were created using the first 10,000 most prevalent terms in IMDB movie reviews. For each study, the 10,000-dimensional BoW feature vector is highly sparse; to avoid over-fitting, 50 percent dropout noise was introduced to the BoW features during training to avoid over-fitting. Adam's empirical success is in line with the author's theoretical results. Adam, like Adagrad, may choose to use sparse features to obtain a higher rate of convergence than a typical SGD with momentum..

On MLP networks trained with dropout noise, the authors compare Adam to other stochastic first order approaches. The result of authors is shown in Figure 2.6(a) where Adam has a higher convergence rate than the other approaches.

The authors also used CIFAR10 datasets in their CNN experiments. Interestingly, despite the fact that both Adam and Adagrad make quick progress in decreasing the cost early in the training, as shown in Figure 2.6(b), Adam and SGD converge more faster for CNNs than Adagrad.

Features that occur more frequently have significant effect on learning rate of the algorithm but Adam does not consider this fact. Although Adam considers the features on every layers of neural network.

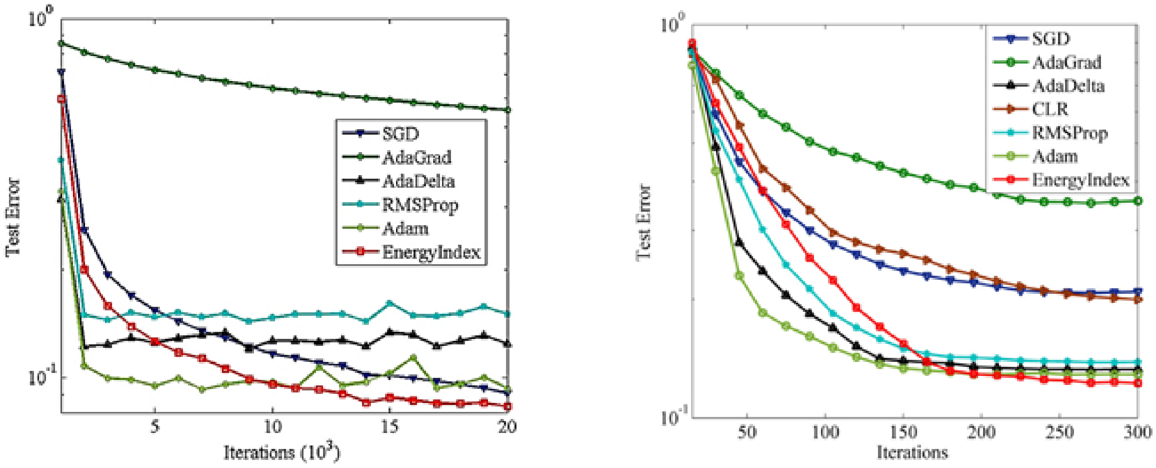


(a) Training of multilayer neural networks on(b) Convolution neural networks training cost MNIST

Figure 2.6: Training Cost of different optimizer in MLP and CNN [12]

The Energy Index Based Optimization Method **EIOM** was proposed by authors in [1] to automatically adjust the learning rate in back propagation. The research focus on the concept that frequently occurring features contributes more in the learning rate during back propagation. The author suggested a novel learning rate technique that dynamically combines knowledge about the neuron's feature frequencies to perform more descriptive learning. EIOM has been suggested in several machine learning models, including LR and MLP with the MNIST dataset and CNN with the CIFAR 10 dataset. The parameters were set to the same values when comparing various optimization techniques in the same model, with the exception of the parameters linked to the different learning rate approaches, which were set according to their original publication.

The authors found that the EIOM-based machine learning model beat the other optimization approaches in terms of classification accuracy while needing no manual adjustment beyond selecting a default value in their testing. By finding the appropriate SGD learning rate, the EIOM was able to alleviate the bottleneck in deep learning. For finding the Energy based learning rate, authors have first initialized the weights.



(a) Test Results of different optimizer on LR

(b) Test Results of different optimizer on MLP

Figure 2.7: Test Results of different optimizer on LR and MLP [1]

Input energy, output energy, energy index and learning rate calculated respectively.While updating the weight, the authors lack considering the past and square gradients. In this study, inclusion of those gradients is considered.

Author in [14] has looked at the concept of multiclass categorization and its need in scientific research. Various classification methods are briefly described. SVMs (Support Vector Machines) have better performance in Binary Classification. The study of author in this paper was to determine the suitability of Support Vector Machines in multiclass categorization.

Different classification algorithm like K-Nearest Neighbor, Decision Tree, Bayesian Classification were compared against SVM. It was found that KNN proved themselves as lazy learners and perform slowly. Decision Tree showed the better result but were highly complex whenever the depth of the tree was on higher side. Next, Bayesian Classifiers needed a good sample of data having more probable states.

The major problem in SVM that the author found was to find the optimal margins for separating the hyperplane w^T+b=0. Hyperplane in SVM ensures the lowest rate of misclassification.

The theoretical finding of the author concluded that Support Vector Machine is better for classification. And this study was focused on implementing Support Vector Machine with Adam and its performance was compared with “ADAM with EI” on all three datasets under consideration.

Another author in [7] have also demonstrated that replacing the softmax layer with a linear support vector machine has a tiny but constant benefit.

Chapter 3

Research Methodology

3.1 Experiment and Overflow

The system model in Figure 3.1 explains the introduction of Energy Index in every neuron of every layer of the neural network. Different three standard datasets are fed as data with the initialization of initial weights, bias, initial learning rate and the decay rate.

The model is introduced in algorithms like Logistic Regression, Multi Layer Percep-

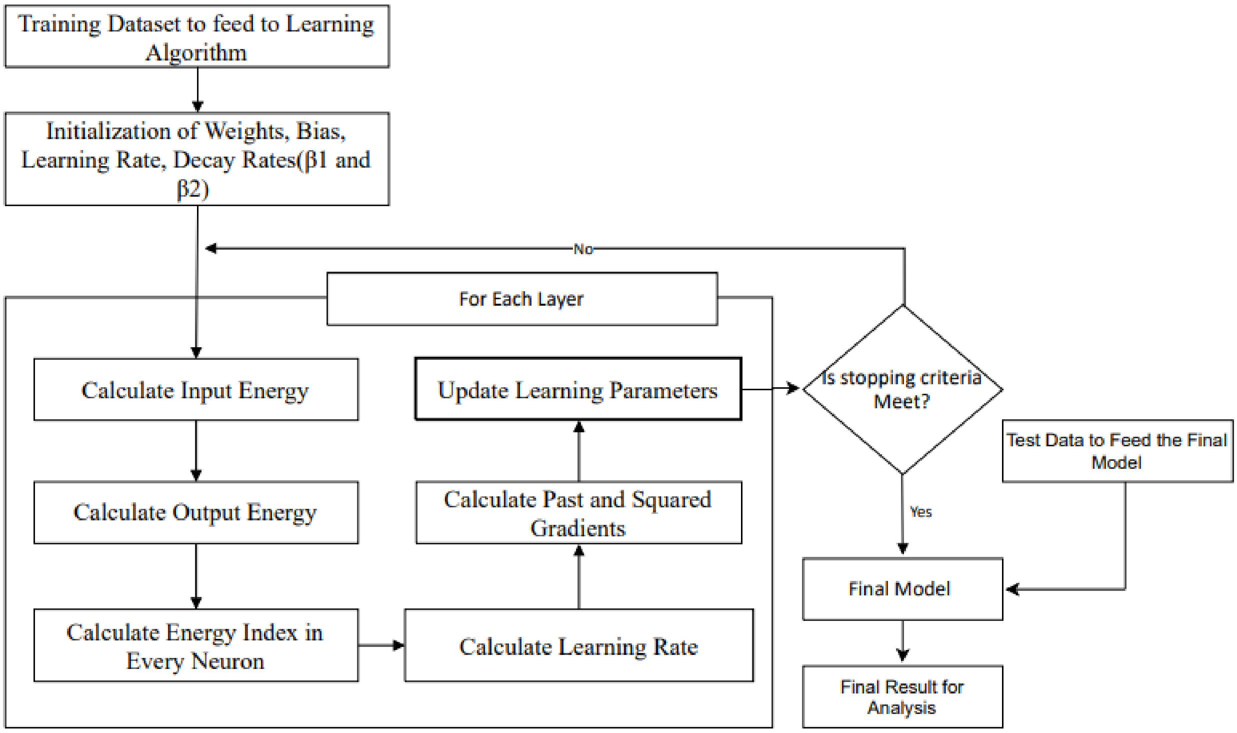

Training Dataset to feed to Learning Algorithm
Initialization of Weights, Bias, Learning Rate, Decay Rates(β1 and β2)
No
For Each Layer
Calculate Input Energy
Update Learning Parameters
Is stopping criteria Meet?
Test Data to Feed the Final Model
Calculate Output Energy
Calculate Past and Squared Gradients
Yes
Final Model
Calculate Energy Index in Every Neuron
Calculate Learning Rate
Final Result for Analysis

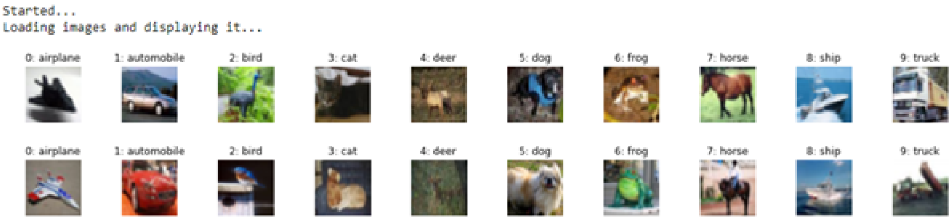
Figure 3.1: System Model

tron and Support Vector Machine.During the learning cycle if the stopping criteria is met, learning is stopped and final model for analysis is achieved. The final system model improved the performance of Adam optimization on introducing energy index.

3.1.1 Datasets

(a) CIFAR 10

The CIFAR-10 dataset (Canadian Institute for Advanced Research) is a set of images used to train machine learning and computer vision algorithms. It's one of the most used datasets for machine learning studies. The CIFAR-10 dataset includes 60,000 32x32 color pictures divided into ten categories. Each class has 6,000 photographs[15]. Different classes of CIFER-10 dataset are defined in Table 3.1.



```
Started...
Loading images and displaying it...
```

Figure 3.2: Display of CIFAR-10 image Datasets

The python format images are extracted using pickle library [16] of python and converted into 32*32 size images as shown in Figure 3.2. With the help of pickle library, the data set file was separated into data and associated label. After the separation train data, train labels, test data and test labels were defined.

(b) MNIST

The MNIST database is a simple data set of handwritten digits that is used to train and evaluate supervised machine learning algorithms. It's a supplement to the NIST database. There are 70,000 black-and-white 28x28 pictures in the collection that depict the numerals 0 through 9.

With 60,000 photos in the training set and 10,000 images in the testing set, the data is split into two groups. The separation of pictures guarantees that a model that has been properly trained may efficiently categorize relevant photos that have not been examined previously based on what it has learned previously.

Table 3.1: CIFAR 10: Labels and Classes

Label	Class
0	airplane
1	automobile
2	bird
3	cat
4	deer
5	dog
6	frog
7	horse
8	ship
9	truck

The information is saved in a simple file format intended for vectors and multi-dimensional matrices. [17].

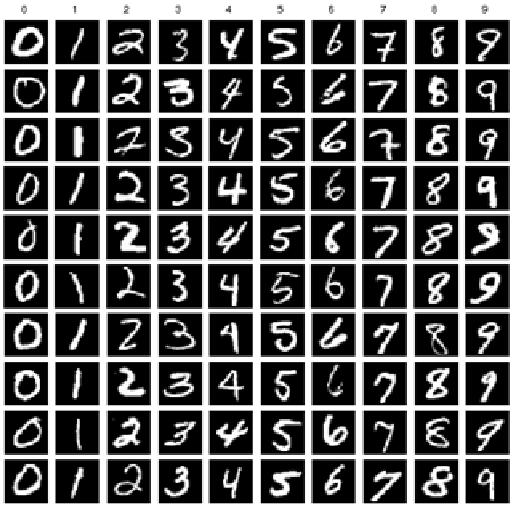


Figure 3.3: Display of MNIST image Datasets [4]

There are four files:

(a) Training set images: train-images-idx3-ubyte

(b) Training set labels: train-labels-idx1-ubyte

(c) Test set images: t10k-images-idx3-ubyte

(d) Test set labels: t10k-labels-idx1-ubyte

The training set has 60000 examples, whereas the test set has 10,000. The test set's first 5000 samples are from the original NIST training set. The final 5000 are drawn from the NIST test set. The first 5000 are more organized and straightforward than the latter 5000.

(c) Fashion MNIST

Fashion-MNIST is a Zalando article image dataset with 60,000 training examples and 10,000 test samples. Each sample is a grayscale picture of 28x28 pixels with a label from one of 10 different categories. According to Zalando, Fashion-MNIST is designed to be a drop-in replacement for the original MNIST dataset for assessing machine learning algorithms[18].

Table 3.2: Fashion MNIST: Labels and Classes

Label	Class
0	T-shirt/Top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankel Boot

Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels. Each pixel has a single pixel value that specifies whether it is bright or dark, with higher values indicating darker. The pixel value is an integer between 0 and 255. The training and testing data sets each include 785 columns.

Different classes of Fashion MNIST dataset defined in Table 3.2

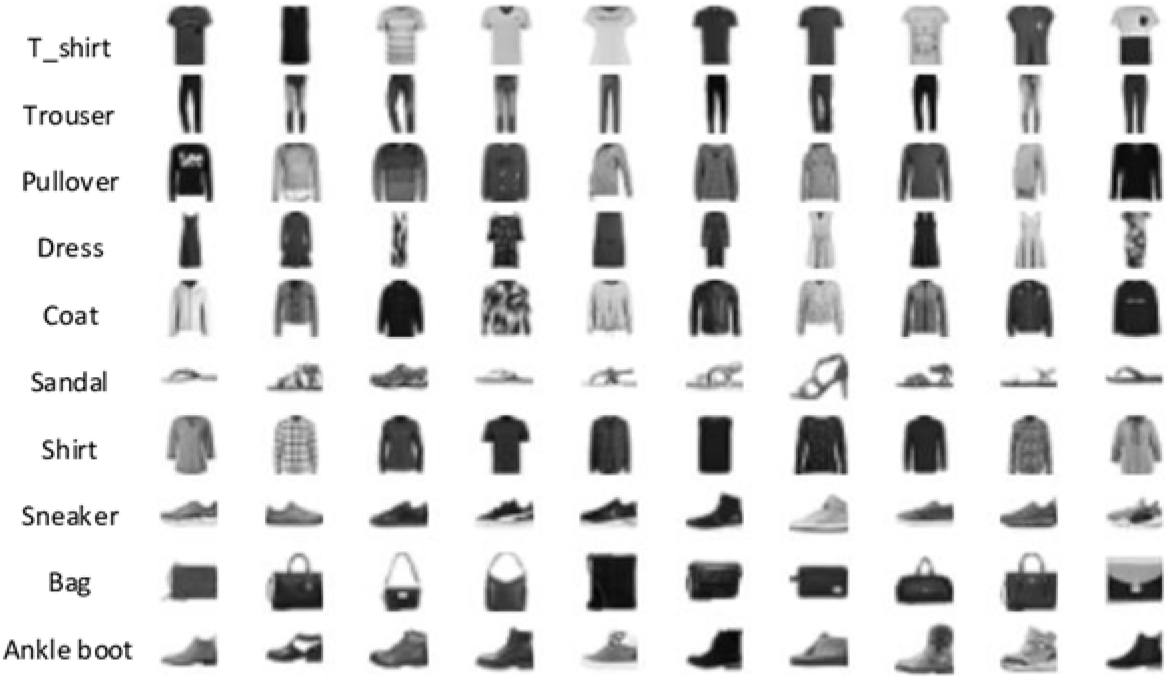


Figure 3.4: Display of Fashion MNIST image Datasets [9]

3.1.2 Initialization

(a) Initialization for CIFAR 10

The initial weight is initialized as the input size of image (32*32*3) random weights and bias as 0. Learning rate is kept as per the algorithm performance and decay rates as beta1 = 0.9, beta2 = 0.999 and eps = 1e-8

(b) Initialization for MNIST and Fashion MNIST

Both the datasets are converted into csv file where the first cell of every column specify its label. Rest of the values in the cells in a row holds hold the information of that label.

Since, the size of the image is 28*28, the initial weight is initialized as the input size of image (28*28) random weights and bias as 0. Learning rate is kept as per the algorithm performance and decay rates as beta1 = 0.9, beta2 = 0.999 and eps = 1e-8

3.1.3 Calculate Input Energy

The input energy of all individual neurons in every layer l is defined as:

$$E^l_{in} = \Sigma^m_{j=1} \mid w^l_j \mid \tag{3.1}$$

where E_{in} is the input energy , w_i^l is the j^{th} column vector of w_l

3.1.4 Calculate Output Energy

The output energy of all individual neurons in every layer is defined as:

$$E^l_{out} = \Sigma^m_{j=1} \mid w^{l+1}_j \mid \tag{3.2}$$

where E_{out} is the output energy, w_i^{l+1} is the i^{th} row vector of w^{l+1}

3.1.5 Calculate Energy Index

Energy Index of individual neurons in l^{th} layer EI^l is define as:

$$EI^l = \lambda \frac{E^l_{in}}{\bar{E}^l_{in}} + (1 - \lambda) \frac{(E^l_{out})^T}{\bar{E}^l_{out}} \tag{3.3}$$

where $\lambda \epsilon [0,1]$ is EI factor that controls the contribution of input and output energy. $\bar{E}^l{}_{in}$ and $\bar{E}^l{}_{out}$ are the expectations of $E^l{}_{in}$ and $E^l{}_{out}$ respectively, and $(E^l{}_{out})^T$ is the transpose of $E^l{}_{out}$ A neuron with a higher input and output energy has a higher energy index in every layer l. A neuron with a lesser input energy and output energy, on the other hand, has a reduced energy index. Finally, EI^l is normalized to a range of [0, 1] as:

$$EI^l = \frac{EI^l}{max(EI^l)} \tag{3.4}$$

3.1.6 Calculate Learning Rate

The learning rate η^l of the neuron in l^{th} layer is defined as:

$$\eta^l = \eta^* \times (1 - EI^l) \tag{3.5}$$

where, η^* is the default value as in section 3.1.2

Here, we can see that if a characteristic is more significant and happens more frequently, the energy index of that feature will be higher, and the learning rate will be lower. A feature that is less common than other characteristics, on the other hand, will have a lower energy index and a higher learning rate.

3.1.7 Calculate Past Gradients and Square Gradients

First moment (past gradients) is calculated from equation 2.9 and second moment (square gradients) is calculated from equation 2.10.

3.1.8 Update the Parameter

Now, the parameters are updated from equation 3.6

$$\theta_t = \theta_{t-1} - \frac{\eta^l}{\sqrt{\widehat{v}_{t-1}} + \epsilon} \widehat{m}_{t-1} \tag{3.6}$$

Where, $\hat{m}$ is calculated from equation 2.9 and $\hat{v}$ is calculated from equation 2.10. Here, η^1 (energy index based learning rate) is introduced in Adam Optimization.

3.2 Evaluation Criteria

The performance of all the algorithms implemented under different datasets will be evaluated under the following criteria.

1. Confusion Matrix

   A confusion matrix is a table that displays the number of correct and erroneous predictions produced by a classifier. It's a measure for assessing the performance of a classification model. To evaluate the performance of a classification model, it is utilized to produce performance metrics such as accuracy, precision, recall, and F1-score. The confusion matrix for the binary classification model will look like Figure 3.5. [19].

   The basic terminologies used by confusion matrix are

   (a) True Positives (TP)

   When the actual value and the predicted value are both positive.

   (b) True negatives (TN)

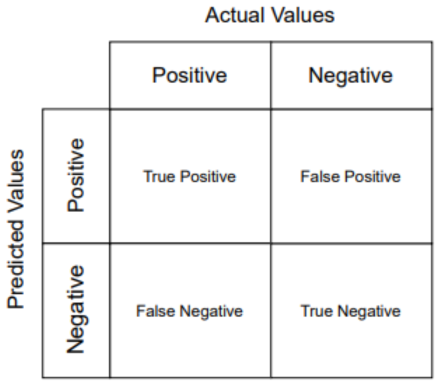
   When the actual value and the predicted value are both negative.

   (c) False positives (FP)

   When the actual value is Negative and the predicted value is Positive.

   (d) False negatives (FN)

   When the actual value is Positive and the predicted value is Negative.



		Actual Values	
		Positive	Negative
Predicted Values	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Figure 3.5: Confusion Matrix [19]

2. Cost

   The cost of a neural network is a measure of its performance in relation to the training sample and expected output. Weights and prejudices, for example, may have an influence.

   The method with lowest cost is considered better.

3. Training Error

   This is the error that is used to train the model back with the help of training data. It is the difference between the model's prediction and the training data sets.

$$TrainingError = PredictedResults - TrainingResults \tag{3.7}$$

4. Test Error

   After the model has been trained, test data is fitted to the model to ensure that it is accurate. A test error is a sort of mistake like this.

$$TestError = PredictedResults - TrainingResults \tag{3.8}$$

Chapter 4

Result and Discussion

4.1 Environment Setup and Tools Used

4.1.1 Environmental Setup

In the Windows 10 operating system, the whole study was carried out and the findings were examined. The Intel(R) Core(TM) i5-8250U CPU was utilized at 1.60GHz. The RAM on the machine was 8GB.

4.1.2 Tools Used

(a) Programming Language

Python is a multi-paradigm, high-level, dynamically typed programming language. The main feature of this language is that the statements it provides are almost like pseudocode, because it allows us to express very strong ideas in a small number of lines of code while still being very readable.

Only fundamental libraries like NumPy, Math for mathematical calculations and Pickle for extracting the cifar-10-batches-py dataset. All the calculations for Energy Index, Neural Networks and Optimizer are done from scratch

(b) IDE

A web-based interactive computational environment, Jupyter Notebook was used to write the python codes. The jupyter used was under the installation of Anaconda3-2021.05 for Python 3.8

4.1.3 Time Complexity

Let, the number of training sample be n, the total number of features be f, the number of neurons at layer i in a neural network be n_i and the number of support vector be n_{sv}, the following expectation for time complexity was calculated [20] :

(a) Neural Network

(i) Adam

The prediction time complexity for adam optimization is [20] :

$$O(pn_{l1} + n_{l1} \times n_{l2} + ...) \tag{4.1}$$

(ii) Adam with EI

The prediction time complexity for adam optimization with energy index is :

$$O(pn_{l1} + i \times n_{l1} \times n_{l2} + ...) \tag{4.2}$$

(b) Support Vector Machine

(a) Adam

The prediction time complexity for adam optimization is [20] :

$$O(n_{sv} \times p) \tag{4.3}$$

(b) Adam with EI

The prediction time complexity for adam optimization is:

$$O(n_{sv} \times p) \tag{4.4}$$

4.2 Result Analysis

Different machine learning models like including Logistic Regression and Multilayer Perceptron were used to evaluate the performance of ”Adam with EI” model. CIFAR 10, MNIST and Fashion MNIST were the datasets used as the foundation for all learning models. When comparing ADAM with ’ADAM with EI’, most parameters are assigned to the identical values, with the exception of those linked to the different learning rate techniques, which are set according to their original report by respected authors.

4.2.1 Logistic Regression

Local minima have no effect on Logistic Regression since its goal function is convex. As a consequence, three distinct datasets were compared.

Since Logistic Regression is single layer, input energy does not exist, so the energy factor λ is set to 0 [1].

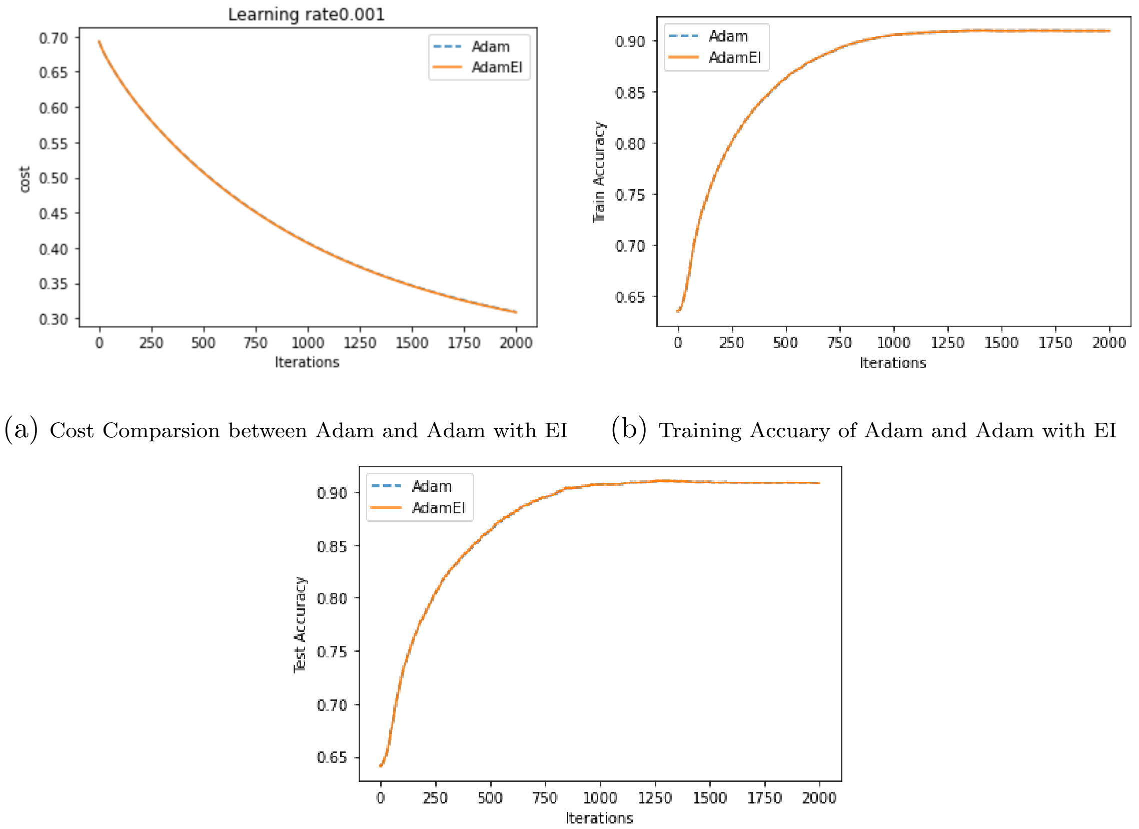
After several test on different learning rates, the best learning rate found was 0.001 for 2000 iterations on all three datasets as the initial learning rate.

(a) CIFAR10 Dataset

CIFAR10, an RGB color dataset with 60,000 images of 10 different categories and a resolution of 32 x 32, was used to test the Logistic Regression model with Adam and ”Adam with EI”. The input for the algorithm to classify the CIFAR10 dataset is 3072 – dimensional data (32 x 32 x 3)[16]. As the size of training image is 3072 x 50000, the new learning rate calculated from ”Adam with EI” was 1 x 3072. 10,000 images were used as test data. On Every iteration , the learning rate of same size is computed to learn the algorithm throughout the layer. Instead, Adam calculated the learning rate of size 1 x 1 for every iteration, and the same learning rate is used through out the layer. Cost function, Train Accuracy and Test Accuracy between ”Logistic Regression with Adam” and ”Logistic Regression with Adam and Energy Index” was compared. The result of comparison is shown in Figure 4.1 (a), Figure 4.1(b) and Figure 4.1(c) respectively. These figures show that both the models attained the same results. All the evaluation metrics like cost, training accuracy, test accuracy, ROC, confusion matrix are identical and similarity is also shown in Table 4.1.

Table 4.1: Comparative Analysis of Logistic Regression in CIFAR10

	Adam	Adam With EI
Cost	0.3091	0.3085
Test Accuracy	90.79 %	90.79 %
Train Accuracy	90.90 %	90.79 %
ROC	0.58	0.58



(a) Cost Comparsion between Adam and Adam with EI

(b) Training Accuary of Adam and Adam with EI

(c) Test Accuary of Adam and Adam with EI

Figure 4.1: Cost, Train and Test Accuracy of Logistic Regression in CIFAR10

Table 4.2: Confusion Matrix for Logistic Regression with CIFAR10

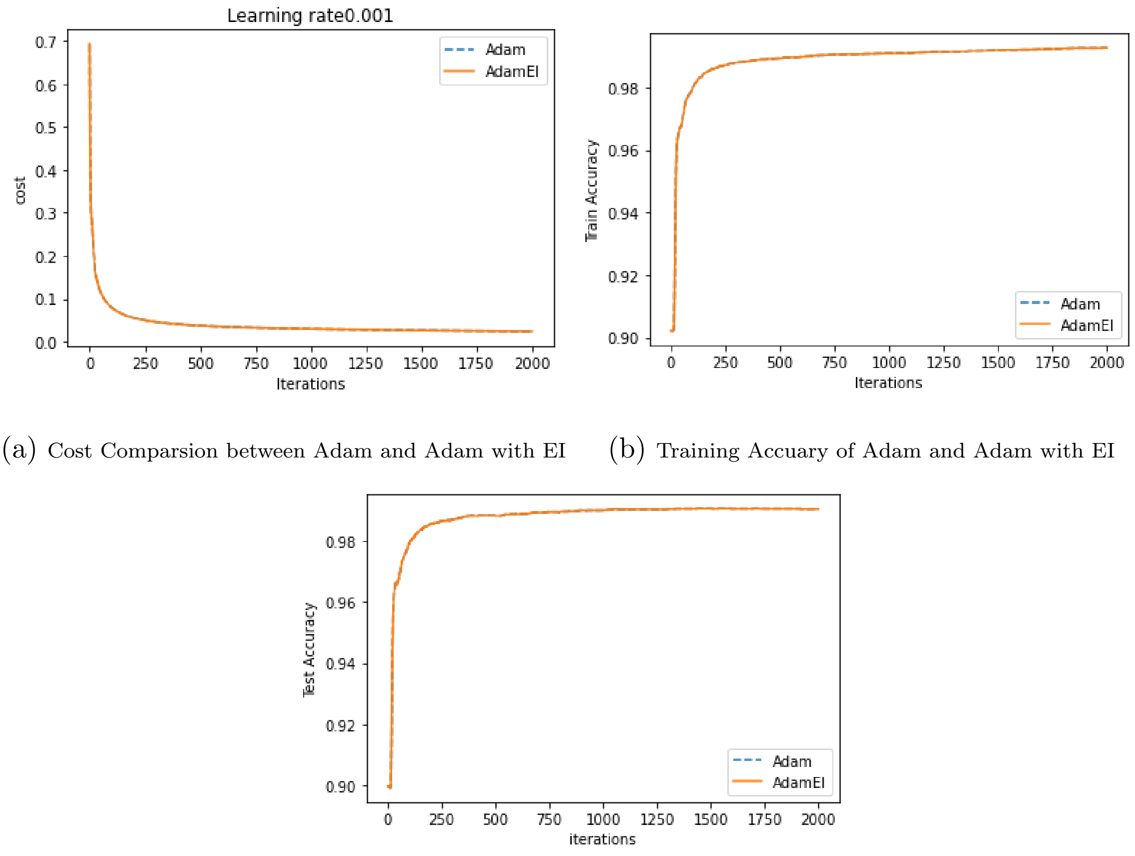
		Actual Value	
		Positive	Negative
Predicted Values	Positive	8911	89
	Negative	168	832

Confusion matrix for the model is tabled in Table 4.2.

Although, both the method have slower convergence speed, they have train and test accuracy of 90.90% and 90.79% respectively in only 2000 iterations. Further it is notice that, both the algorithm have reached the stability after 1000 iterations, minimizing the impact of noise and outliers.

(b) MNIST Dataset

MNIST, black and white dataset with 70,000 images of 10 different categories and a resolution of 28 x 28, was used to test the Logistic Regression model with Adam and "Adam with EI". The input for the algorithm to classify the MNIST dataset is 784 – dimensional data (28 x 28)[17]. Dataset is stored in



(a) Cost Comparsion between Adam and Adam with EI

(b) Training Accuary of Adam and Adam with EI

(c) Test Accuary of Adam and Adam with EI

Figure 4.2: Cost, Train and Test Accuracy of Logistic Regression in MNIST

CSV designed to store the vectors and multidimensional matrices. To reduce the computational complexity, the dimension of training data used was 784 x 32000, the new learning rate calculated from "Adam with EI" was 1 x 784. 10,000 images were used as test data. On Every iteration , the learning rate of same size is computed to learn the algorithm throughout the layer. Instead, Adam calculated the learning rate of size 1 x 1 for every iteration, and the same learning rate is used through out the layer. Cost function, Train Accuracy and Test Accuracy between "Logistic Regression with Adam" and "Logistic Regression with Adam and Energy Index" was compared. The result of comparison is shown in Figure 4.2 (a), Figure 4.2(b) and Figure 4.2(c) respectively. These figures show that both the models attained the same results. All the evaluation metrics like cost, training accuracy, test accuracy, ROC, confusion matrix are

Table 4.3: Comparative Analysis of Logistic Regression in MNIST

	Adam	Adam With EI
Cost	0.02	0.02
Test Accuracy	99.02 %	99.02 %
Train Accuracy	99.26 %	99.26 %
ROC	0.97	0.97

same and similarity between the models is also shown in Table 4.3. Further, Confusion matrix in Table 4.4.

Both the method have faster convergence speed, they have overall accuracy of 99% in 2000 iteration. Further it is notice that, both the algorithm have started achieving stability only after 500 iterations, minimizing the impact of noise and outliers.

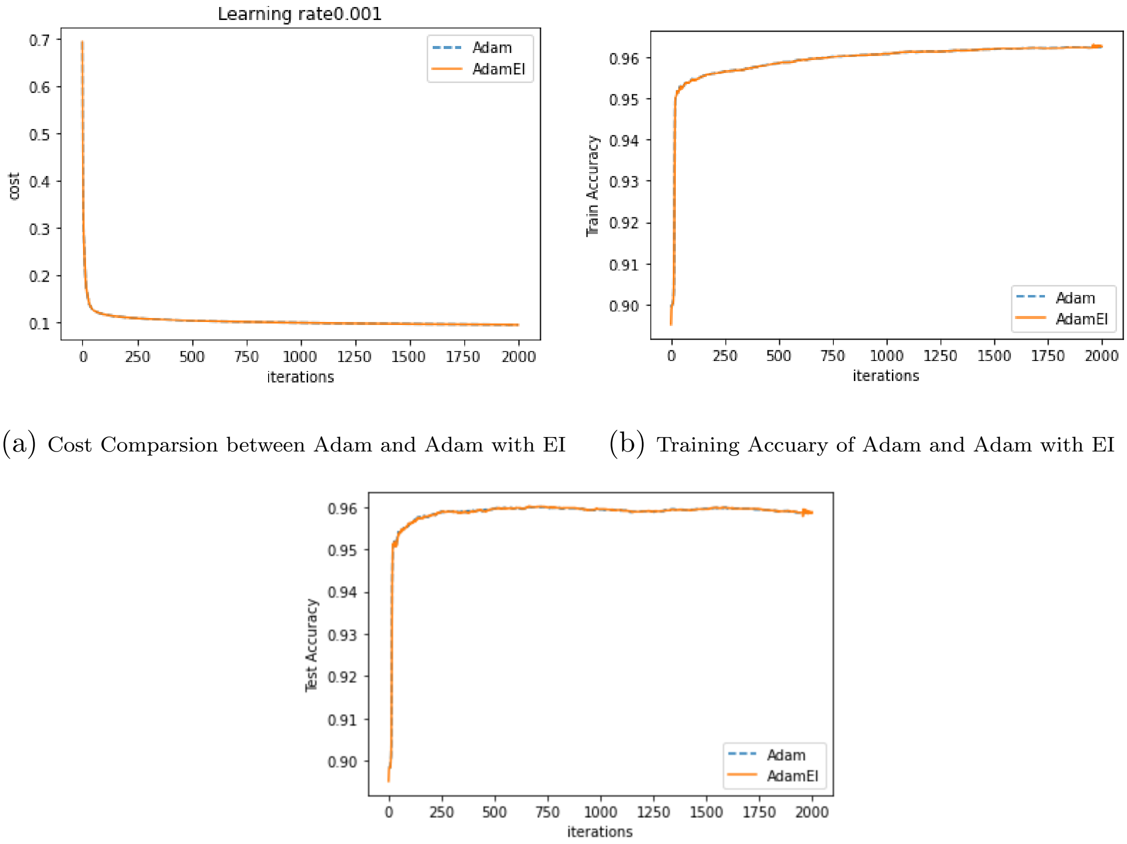
Table 4.4: Confusion Matrix for Logistic Regression with MNIST

		Actual Value	
		Positive	Negative
Predicted Values	Positive	8958	40
	Negative	58	944

(c) Fashion MNIST

Fashion MNIST, black and white dataset with 70,000 images of 10 different categories and a resolution of 28 x 28, was used to test the Logistic Regression model with Adam and "Adam with EI". The input for the algorithm to classify the MNIST dataset is 784 – dimensional data (28 x 28)[18].

Dataset is stored in CSV designed to store the vectors and multidimensional matrices. To reduce the computational complexity, the dimension of training data used was 784 x 50000, the new learning rate calculated from "Adam with EI" was 1 x 784. 10,000 images were used as test data. On Every iteration , the learning rate of same size is computed to learn the algorithm throughout the layer. Instead, Adam calculated the learning rate of size 1 x 1 for every itera-



(a) Cost Comparsion between Adam and Adam with EI

(b) Training Accuary of Adam and Adam with EI

(c) Test Accuary of Adam and Adam with EI

Figure 4.3: Cost, Train and Test Accuracy of Logistic Regression in Fashion MNIST

tion, and the same learning rate is used through out the layer. Cost function, Train Accuracy and Test Accuracy between ”Logistic Regression with Adam” and ”Logistic Regression with Adam and Energy Index” was compared. The result of comparison is shown in Figure 4.3 (a), Figure 4.3(b) and Figure 4.3(c) respectively. These figures show that both the models attained the same results. All the evaluation metrics like cost, training accuracy, test accuracy, ROC, con-

Table 4.5: Comparative Analysis of Logistic Regression in Fashion MNIST

	Adam	Adam With EI
Cost	0.09	0.09
Test Accuracy	95.85 %	95.87 %
Train Accuracy	96.23 %	96.25 %
ROC	0.87	0.87

fusion matrix are same and similarity between the models is also shown in Table 4.5 and confusion matrix in Table 4.6.

Both the method have faster convergence speed, they have test and training accuracy of 95.85% and 96.23% respectively in 2000 iterations. Further it is notice that, both the algorithm have started achieving stability only after 250 iterations, minimizing the impact of noise and outliers.

Table 4.6: Confusion Matrix for Logistic Regression with Fashion MNIST

		Actual Value	
		Positive	Negative
Predicted Values	Positive	8812	177
	Negative	238	773

4.2.2 Multi Layer Perceptron

The MLP model employed in the study comprises 784 input units, 64 completely linked hidden units, and 1 output unit. One hot encoding is modelled for binary classification since the output unit is 1. The sigmoid activation function is employed. As in [21], the algorithm is trained for 1000 iterations.

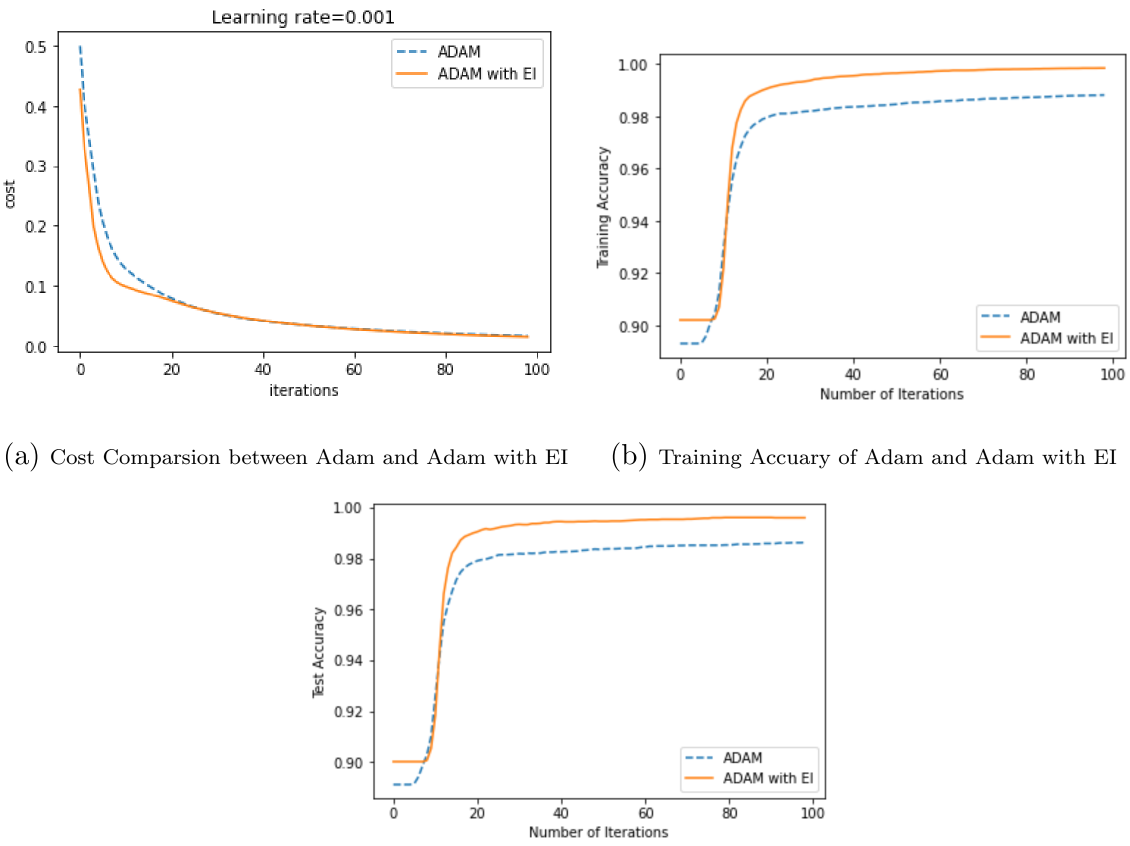
As MLP has multiple layers, energy factor λ is set 0.1 [1].

(a) MNIST Dataset

During the study, different experiments on initial learning rate was tested on different iterations, and the best initial learning rate found was 0.001 for 1000 iteration. The dimension of training data used was 784 x 32000.

On using Adam as the optimizer, only every iteration, the learning rate was calculated and for every layers, the same learning rate is used.

On using Enery Index model, the learning rate calculated was 1 x 784 for the first layer and 1 x 64 for the second layer.In Every iterations new learning rate was calculated. As seen in Figure 4.4, MLP with "Adam with EI" has the better classification accuracy. Figure 4.4(a) shows that the convergence MLP with introduction of EI is better than the normal Adam. Figure 4.4(b) and Figure 4.4(c) show both algorithms during earlier iterations did not perform



(a) Cost Comparsion between Adam and Adam with EI (b) Training Accuary of Adam and Adam with EI

(c) Test Accuary of Adam and Adam with EI

Figure 4.4: Cost, Train and Test Accuracy of MLP in MNIST

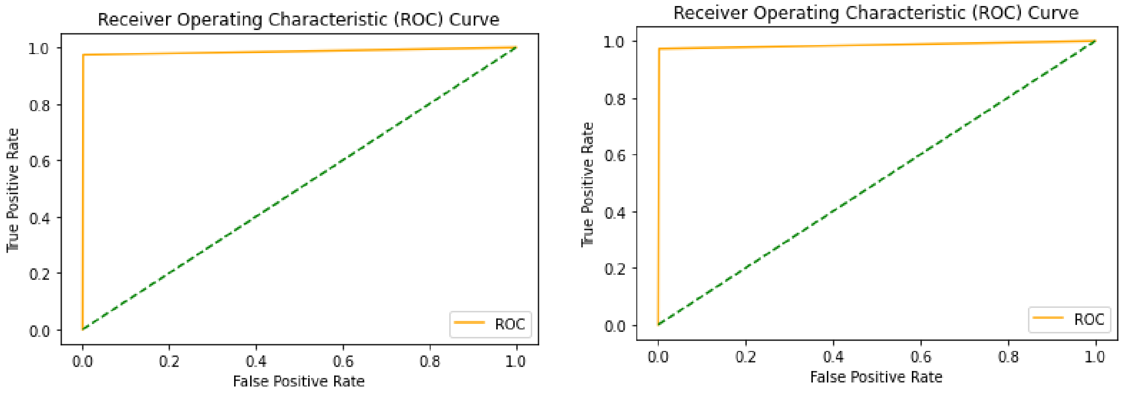
better, but they have good performance and achieved the stability after 200 iterations. From Table 4.7, result show that all the performance metrics like

Table 4.7: Comparative Analysis of MLP in MNIST

	Adam	Adam With EI
Cost	0.016	0.01
Test Accuracy	98.63 %	99.60 %
Train Accuracy	98.81 %	99.84 %
ROC	0.98	0.99

cost, test accuracy, training accuracy show that MLP has better performance with the Energy Index along with Adam. Figure 4.5 further supports the better result of ROC value in Table 4.7.

Comparing the confusion matrix Table 4.8 and Table 4.9, we can see that "MLP



(a) ROC of MLP with Adam and EI for MNIST

(b) ROC of MLP with Adam for MNIST

Figure 4.5: ROC Curve of MLP for MNIST

with EI” is better than ”MLP with Adam” on MNIST dataset.

Table 4.8: Confusion Matrix for MLP with Adam for MNIST

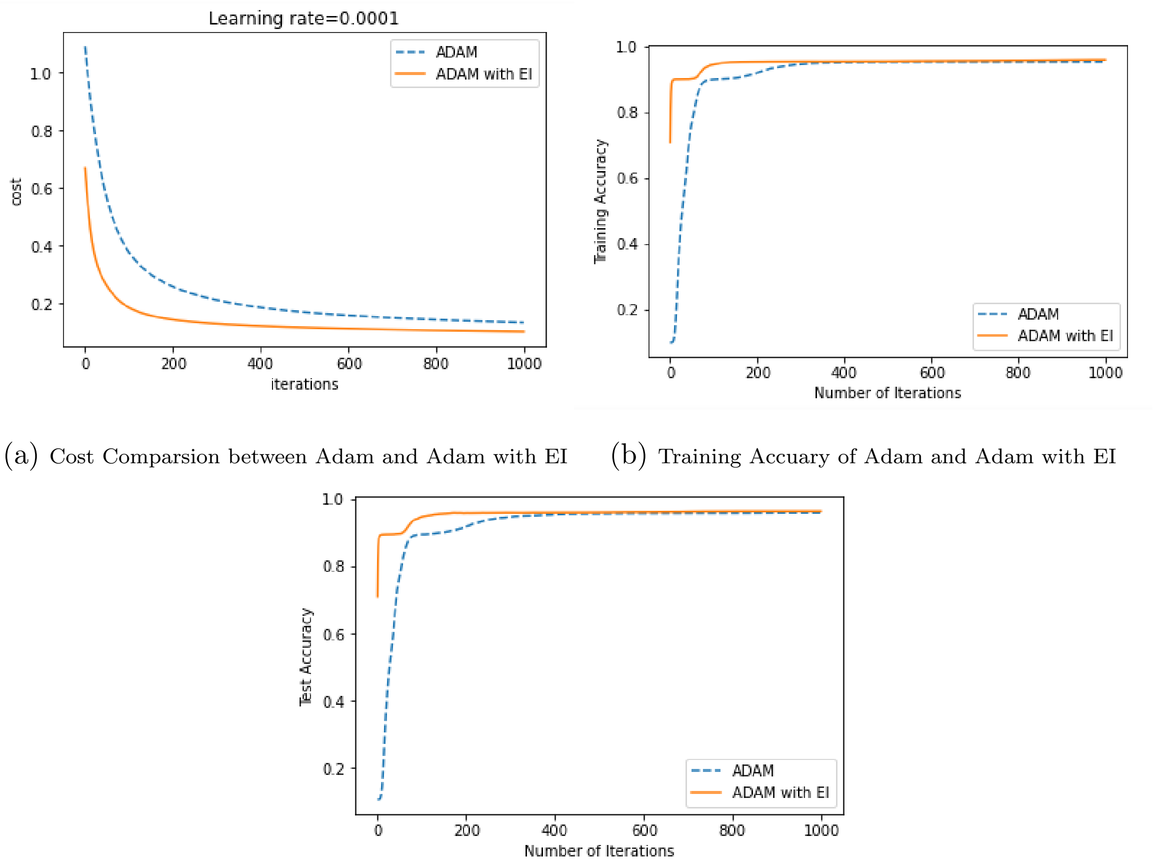
		Actual Value	
		Positive	Negative
Predicted Values	Positive	8982	20
	Negative	28	970

Table 4.9: Confusion Matrix for MLP with Adam and EI for MNIST

		Actual Value	
		Positive	Negative
Predicted Values	Positive	8988	14
	Negative	26	972

(b) Fashion MNIST

During the study, different experiments on initial learning rate was tested on different iterations, and the best initial learning rate found was 0.001 for 1000 iteration. The dimension of training data used was 784 x 50000. On using Adam as the optimizer, only every iteration, the learning rate was calculated and for every layers, the same learning rate is used. On using Enery Index model, the learning rate calculated was 1 x 784 for the first layer and 1 x 64 for the second



(a) Cost Comparsion between Adam and Adam with EI (b) Training Accuary of Adam and Adam with EI

(c) Test Accuary of Adam and Adam with EI

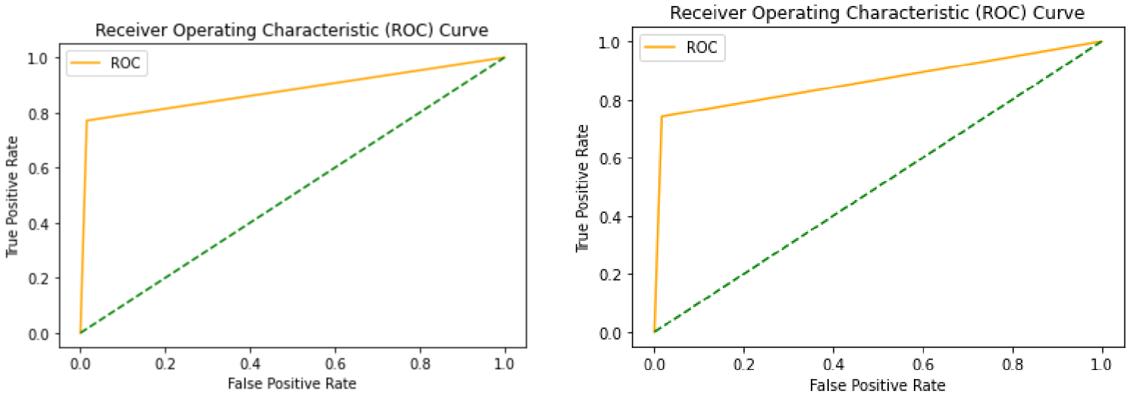
Figure 4.6: Cost, Train and Test Accuracy of MLP in Fashion MNIST

layer.In Every iterations new learning rate was calculated. As seen in Figure 4.6, MLP with ”Adam with AI” has the better classification accuracy. Figure 4.6(a) shows both initial and final cost MLP with introduction of EI is less than the normal Adam. Figure 4.6(b) and Figure 4.6(c) show both algorithms since the earlier iterations, Adam with EI have better performance and achieved the stability after 500 iterations.

Table 4.10: Comparative Analysis of MLP in Fashion MNIST

	Adam	Adam With EI
Cost	0.13	0.009
Test Accuracy	95.82 %	96.23 %
Train Accuracy	95.44 %	96.03 %
ROC	0.86	0.88

From Table 4.10, result show that all the performance metrics like cost, test accuracy, training accuracy show that MLP has better performance with the Energy Index along with Adam. Figure 4.7 further supports the better result of ROC value in Table 4.10.Comparing the confusion matrix Table 4.11 and Table 4.12,we can see that ”MLP with EI” is better than ”MLP with Adam” on MNIST dataset.



(a) ROC of MLP with Adam and EI for Fashion MNIST

(b) ROC of MLP with Adam for Fashion MNIST

Figure 4.7: ROC Curve of MLP for Fashion MNIST

Table 4.11: Confusion Matrix for MLP with Adam and EI for Fashion MNIST

		Actual Value	
		Positive	Negative
Predicted Values	Positive	8805	134
	Negative	243	818

Table 4.12: Confusion Matrix for MLP with Adam for Fashion MNIST

		Actual Value	
		Positive	Negative
Predicted Values	Positive	8795	144
	Negative	274	787

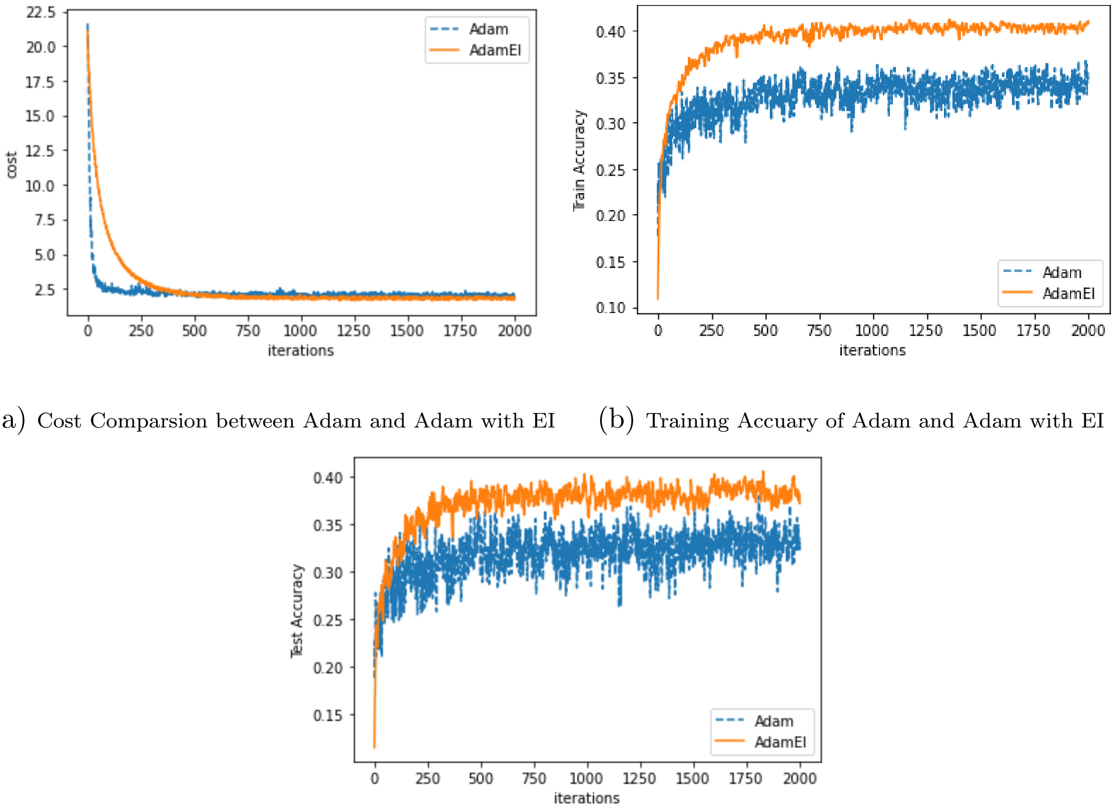
4.2.3 Support Vector Machine

Hinge loss and linear kernel was used to study the performance of support vector machine in both the optimizer. With the help of cross validation, the best learning rate, regularization parameter calculated was 0.0001 and 1.000000e+03 for 2000 iteration for all the datasets.

(a) CIFAR10 Dataset

Mean of training and test data was calculated and datasets were normalized by subtracting the mean image from the original datasets.

The training dimension was 50000 X 3072 and test dimension was 10000 x 3072. For Adam with EI, the newly calculated learning rate is of size 3072 x 10 since there are 10 classification to be done.



(a) Cost Comparsion between Adam and Adam with EI (b) Training Accuary of Adam and Adam with EI

(c) Test Accuary of Adam and Adam with EI

Figure 4.8: Cost, Train and Test Accuracy of SVM in CIFAR10

Cost function, Train Accuracy and Test Accuracy between "SVM with Adam" and "SVM with Adam and Energy Index was compared. The result of comparison is shown in Figure 4.8(a), Figure 4.8(b) and Figure 4.8(c) respectively.

These figures show that among, Among the two models, "Adam with EI" has the better classification.

Though SVM with Adam only converged faster, it could not perform better on higher iterations on term of cost. Adam with EI started with very low train and test accuracy, but these parameters also performed better on higher iterations. However, the the results are better than standard sklearn libray[22].From Figure 4.8(a) and Figure 4.8(b), we can conclude that Adam with EI has more resistivity. Overall Accuracy of Adam was 32 % where as accuracy of "Adam with EI" was 37 %. Table 4.13 further clarifies that Adam with EI performed better than Adam.

Table 4.13: Comparative Analysis of SVM in CIFAR10

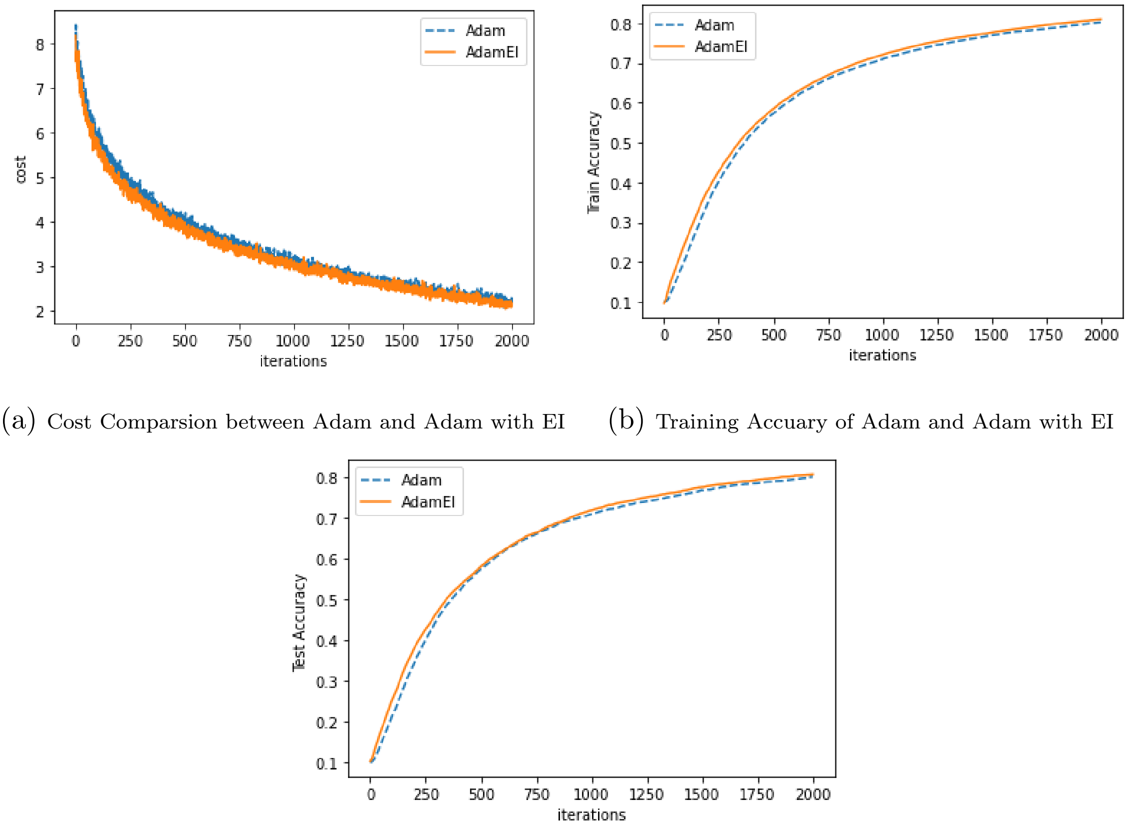
	Adam	Adam With EI
Cost	1.94	1.74
Test Accuracy	34.48 %	39.04 %
Train Accuracy	34.95 %	40.26 %
Overall Accuracy	32 %	37 %
ROC	0.65	0.68

(b) MNIST Dataset

The training dimension was 32000 X 784 and test dimension was 10000 x 3072. For Adam with EI, the newly calculated learning rate is of size 784 x 10 since there are 10 classification to be done.

Cost function, Train Accuracy and Test Accuracy between "SVM with Adam" and "SVM with Adam and Energy Index was compared. The result of comparison is shown in Figure 4.9(a), Figure 4.9(b) and Figure 4.9(c) respectively. These figures show that among the two models, "Adam with EI" has the better classification accuracy.

Both the models could not converge better, could find the stability even upto 2000 iterations, we can see from Table 4.14 that "Adam with EI" has,however, better performance than Adam.



(a) Cost Comparsion between Adam and Adam with EI

(b) Training Accuary of Adam and Adam with EI

(c) Test Accuary of Adam and Adam with EI

Figure 4.9: Cost, Train and Test Accuracy of SVM in MNIST

Table 4.14: Comparative Analysis of SVM in MNIST

	Adam	Adam With EI
Cost	2.32	2.17
Test Accuracy	80.25 %	80.8 %
Train Accuracy	80.16 %	80.92 %
Overall Accuracy	80 %	81 %
ROC	0.83	0.83

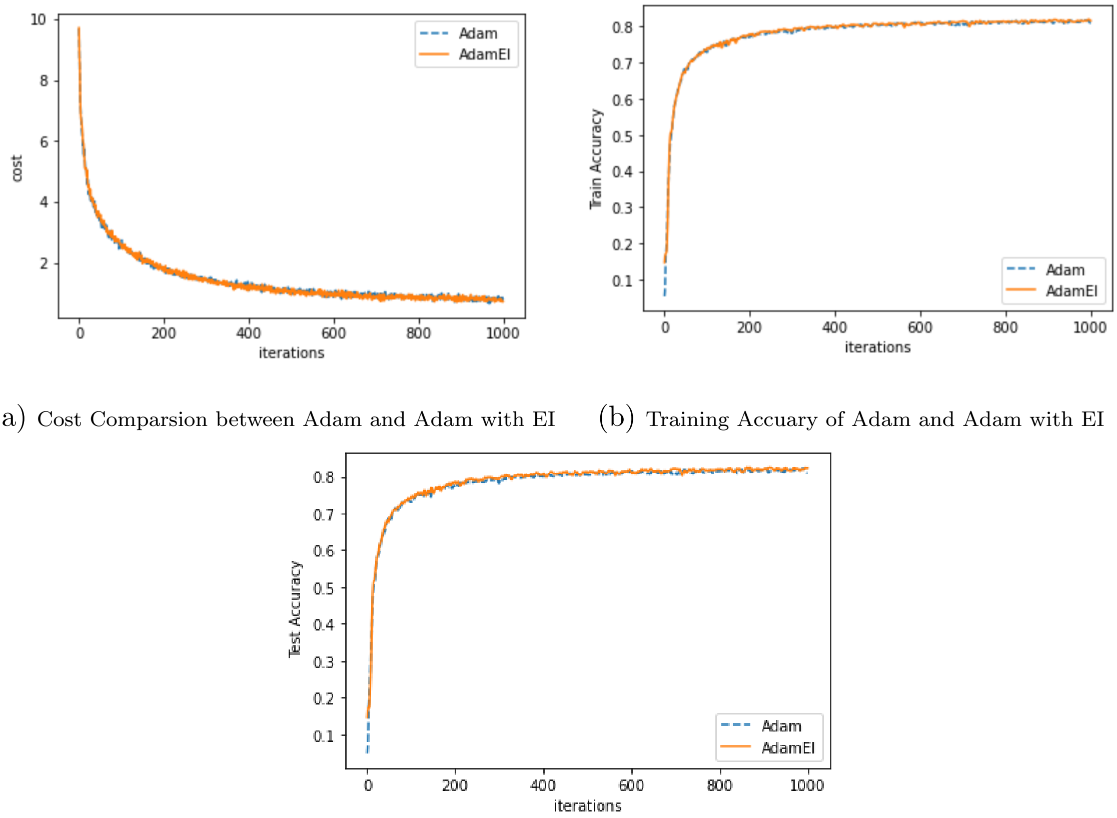
(c) Fashion MNIST

The training dimension was 50000 X 784 and test dimension was 10000 x 3072. For Adam with EI, the newly calculated learning rate is of size 784 x 10 since there are 10 classification to be done.

Cost function, Train Accuracy and Test Accuracy between ”SVM with Adam”

Table 4.15: Comparative Analysis of SVM in Fashion MNIST

	Adam	Adam With EI
Cost	0. 83	0.73
Test Accuracy	81.01 %	82.29 %
Train Accuracy	80.82 %	81.49 %
Overall Accuracy	81 %	82 %
ROC	0.97	0.97



(a) Cost Comparsion between Adam and Adam with EI

(b) Training Accuary of Adam and Adam with EI

(c) Test Accuary of Adam and Adam with EI

Figure 4.10: Cost, Train and Test Accuracy of SVM in Fashion MNIST

and ”SVM with Adam and Energy Index was compared. Figures 4.10(a), 4.10(b), and 4.10(c) illustrate the results of comparison, accordingly. These graphs indicate that both models have a higher degree of convergence.

After 200 iterations, both models had a minimal influence of noise and outliers. Table 4.15 shows that ”Adam with EI” has a superior overall accuracy than Adam, with a 1% increase in overall accuracy.

Chapter 5

Conclusion

5.1 Implications

5.1.1 Theoretical Contributions

In this study, with the support of Adam optimizer, learning rate to individual neurons in different variants of machine learning algorithms was introduced. The performance of energy introduced model was compared to that of the conventional Adam optimizer and assessed. The introduction of a faster learning rate to individual neurons resulted in a better categorization accuracy.

5.1.2 Practical Implications

Outcomes from all the datasets under consideration showed that Logistic Regression produced almost identical results. In MNIST datasets, LR fared better with an accuracy of 99.26 %. MNIST had superior outcomes in terms of convergence than the other datasets. The algorithm, on the other hand, achieved accuracy of more than 90% in all datasets studied. Though, Multi Layer Perceptron had the better accuracy than Logistic Regression, the latter had accuracy more than Support Vector Machine. In the MNIST dataset, a multilayer perceptron with a single hidden layer produced a very good result of 99.84. Adam, on the other hand, had a score of 98.81%. The suggested model achieved smooth stability better than the other algorithms under consideration due to its high convergence speed. Energy introduced adam obtained 96.03% using a low cost value for the Fashion MNIST dataset and was more accurate despite being unstable throughout the inner learning period.

Though the introduction of an energy index to a support vector machine did not increase performance on CIFAR10 datasets, it still outperformed traditional Adam in terms of accuracy. Support vector machine have better results for MNIST and Fashion MNIST Datasets, but it could not classify better than linear regression or multilayer perceptron. On the basis of these results, we can say that, SVM are better for high dimensional space and could not perform better on low dimensional datasets for image classification.

During the study, it was also found that calculation for learning rate for individual neuron may accumulate the time complexity in adam optimization.

5.2 Recommendation for future works

As the study is focused in analyzing the performance of Adam on introducing the energy index, the study is limited only to three algorithms and three datasets. Performance of the "Adam with EI" can be further studied in other algorithms and datasets. Further, the optimizer can also be implemented in other applications of Artificial Intelligence like text processing and speech recognition.

References

[1] H. Zhao, F. Liu, H. Zhang, and Z. Liang, “Research on a learning rate with energy index in deep learning,” 2019.

[2] J. L. B. Diederik P. Kingma, “Adam: A method for stochastic optimization,” 2015.

[3] Neural network. Accessed On: 6/16/2021. [Online]. Available: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks.htm

[4] T. Wood. Sigmoid function. Accessed on 6/16/2021. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/sigmoid-function

[5] S. Jaiswal. Logistic regression in machine learning. Accessed on 6/16/2021. [Online]. Available: https://www.javatpoint.com/logistic-regression-in-machine-learning

[6] C. Cortes and V. Vapnik, “Support-vector networks,” 1995.

[7] Y. Tang, “Deep learning using linear support vector machines,” *arXiv preprint arXiv:1306.0239*, 2013.

[8] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, vol. 12, 2011.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” pp. 248–255, 2009.

[10] J. Pennington, R. Socher, Christopher, and D. Manning, “Glove: Global vectors for word representation,” 2014.

[11] M. D. Zeiler, "Adadelta: An adaptive learning rate method," 2012.

[12] G. Hinton, N. Srivastava, and K. Swersky, "Lecture 6.5-rmsprop, overview of mini-batch, coursera: Neural networks for machine learning," 2012.

[13] T. D. Manfred Mannle, Alain Richard, "Identification of rule-based fuzzy models using the rprop optimization technique," 1995.

[14] S. K. Y. Yashima Ahuja, "Multiclass classification and support vector machine," vol. 2, 2012.

[15] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[16] Python object serialization. Accessed On: 6/16/2021. [Online]. Available: https://docs.python.org/3/library/pickle.html

[17] Mnist dataset. Accessed on 6/16/2021. [Online]. Available: https://deepai.org/dataset/mnist

[18] Fashion mnist. Accessed on 6/16/2021. [Online]. Available: https://www.kaggle.com/zalando-research/fashionmnist

[19] A. Suresh. Confusion matrix. Accessed On: 6/16/2021. [Online]. Available: https://medium.com/analytics-vidhya/what-is-a-confusion-matrix-d1c0f8feda5

[20] Computational complexity of machine learning algorithms. Accessed On: 6/16/2021. [Online]. Available: https://www.thekerneltrip.com/machine/learning/computational-complexity-learning-algorithms

[21] D. A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units," 2016.

[22] Z. Khan. Accessed on 6/16/2021. [Online]. Available: https://www.kaggle.com/zaiyankhan/vgg-16-with-svm-on-cifar-10