ORIE4741 Final Report - Gradient Descent Optimization Methods

Henry C. Ogworonjo (hco23), Shishi Zhang (sz559)

***Abstract*—In this project, we investigate gradient decent method which is one of the key concepts covered in the course "Learning with big messy data". Specifically, we study the various gradient descent optimizer and propose a novel strategy for optimizing gradient descent. To accomplish this, we perform three sub-tasks. First, we did a mini literature survey of the commonly used algorithms for optimizing the gradient descent method. Next we design some computational experiments to explore the performance of some of these algorithms and evaluate their performance using accuracy and speed as metrics. Finally, inspired by three of these algorithms, namely RMSprop, Adams and Stochastic Gradient descent, we propose a new method which we call the RADAMS algorithm. The experiment on the existing algorithms was carried out on a multi-class color image data (CIFAR-10). We compared the speed of both the existing and proposed algorithm using the MNIST data. Finally, to validate the proposed algorithm, we compare its accuracy to the stochastic gradient descent and Adam algorithm using an image dataset (CIFAR-100) and a stock dataset (GOOGLE). The proposed algorithm shows promising results on all the tested data.**

I. Introduction

Gradient descent is one of the most popular optimization algorithms and arguably the most common optimization algorithm for the neural network. Stochastic gradient descent, a variant of the gradient descent in particular has emerged as one of the most used training algorithms for deep neural networks [1]. The accuracy and efficiency of trained neural networks are greatly affected by the choice of the activation function and hyperparameters. To optimize the deep neural networks, several hyperparameters, including learning rate, weight decay and dropout rate, are manually adjusted. Among these hyper-parameters, the learning rate is generally considered to be the most important. Learning rate controls the speed of adjusting the weights of the neural networks with respect to the loss gradient. When the learning rate is too small, it could take a long time to converge. When the learning rate is too large, the local minima could be missed resulting in divergence problem. Therefore, optimizing learning rate is crucial for improving the performance of the neural networks. Typically, the learning rate is configured randomly or set by the user according to intuition or past experiences. However, this method is not only time-consuming but also hard for getting the proper learning rate. Consequently, researchers have been investigating how to address this important problem. Some researchers proposed learning rate scheduler to change the learning rate during the training process. Learning rate schedulers change the learning rate according to a pre-defined scheduler or a pre-defined threshold. If the objective between epochs falls below the threshold, the learning rate will be reduced to a default value. However, the learning rate schedulers has to be set before training the data, therefore, it still cannot be adjust during the training process to adapt to dataset characteristics. Other researchers proposed another method called gradient descent optimization method (gradient decent optimizer), which could adaptively change the learning rate during the training process and improve the accuracy of the model.

In this project, we first conduct a literature survey of some of the methods that have been proposed for gradient descent optimization method. Next, we design some computational experiments to explore the performance of some of these algorithms and evaluate their performance using accuracy and speed as metrics. Finally we propose a strategy for optimizing the gradient descent. The rest of this report is organized as follows. Section II presents the the survey of learning rate algorithms. Section III describes our methodology for proposing a strategy for updating learning rate. Section IV describes the datasets, experiments and results of using existing and proposed algorithms. Section V discuses the conclusion of this project and possible future works.

II. Survey of Gradient Descent Optimization Algorithms

The Stochastic gradient descent [2] is simple, performs well across a variety of applications [3] - [5], and possess strong theoretical foundations including guarantee of saddle point avoidance, improved generalization and Bayesian interpretability [1] - [9]. Like many machine learning methods, the goal of the SGD is to solve the following non-convex optimization problem

$$\min_{\mathbf{w}} f(\mathbf{w})$$

where f is the loss function. The SGD updates the current weight using gradient multiplied by the learning rate, α. The update for SGD is :

$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w_t} \tag{1}$$

Although SGD is effective, it suffers some drawbacks. One disadvantage of the SGD is that it scales the gradient uniformly in all directions. This can be detrimental to ill-scaled problems. Additionally it makes the process of tuning the learning rate laborious [1]. As a result, researchers have proposed methods to improve the performance of the SGD. Broadly, the improvements modify the gradient or/and learning rate to achieve a better performance. Consequently, We divide these methods into 4 categories as follow:

- Momentum-based acceleration algorithms
- Adaptive learning rate algorithms
- Hybrid algorithms
- Switching algorithms

A. Momentum-based acceleration algorithms

These algorithms update the weights by using exponential decaying averages to combine the past gradient values with the current gradient values. The past gradient values are given higher weight than the current weight to ensure that the update is not overly sensitive to the current gradient value. Two popular algorithms in this category are the Momentum and Nesterov algorithm. The **Momentum** algorithm [10] helps accelerate SGD in the relevant direction and dampens oscillations by adding a fraction of the update vector of the past time step to the current update vector. The update for Momentum is

$$w_{t+1} = w_t - \alpha V_t, \tag{2}$$

where $V_t = \beta V_{t-1} + (1-\beta)\frac{\partial L}{\partial w_t}$, β is the momentum coefficient and V is initialized to 0.

Another momentum-based accelerating algorithm is the **Nesterov Accelerated Gradient** (NAG) which was proposed in 2013 [11] as an improvement on the classical momentum algorithm. In [11] the authors showed that when stochastic gradient descent with momentum uses a well-designed random initialization and a particular type of slowly increasing schedule for the momentum parameter, it can train both DNNs and RNNs (on datasets with long-term dependencies) to levels of performance that were previously achievable only with Hessian-Free optimization.

B. Adaptive learning rate algorithms

These are learning rate modifying algorithms. The algorithms in this category adapt the learning rate to the gradient by diagonally scaling the gradient via estimates of the function's curvature. Essentially, the algorithms ensure that the update is small when the gradient is large to prevent a huge value from being subtracted from the current weight. To achieve this, the learning rate is scaled by the current gradient. Examples of the adaptive algorithm include AdaGrad, RMSprop, and Adadelta. In **AdaGrad** (or Adaptive gradient) [12], the learning rate is divided by the square root of cumulative sum of current and past squared gradients while keeping the gradient component unchanged. The update for Adagrad is

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{S_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t} \tag{3}$$

where

$$S_t = S_{t-1} + \left(\frac{\partial L}{\partial w_t}\right)^2 \tag{4}$$

S is initialized to 0 and ϵ is a small floating point factor that ensures that the denominator term is positive. The **RMSprop** (or Root mean square prop) [13] is an improvement of the AdaGrad where the S_t in (3) is computed as:

$$S_t = \beta S_{t-1} + (1-\beta)\left(\frac{\partial L}{\partial w_t}\right)^2 \tag{5}$$

Note that unlike the AdaGrad that takes the cumulative sum of squared gradients, the RMSprop takes the exponential moving average of the gradients. The **Adadelta** [14] is another improvement on AdaGrad that completely removes the use of learning rate parameter and has an update given as:

$$w_{t+1} = w_t - \frac{\sqrt{D_{t-1} + \epsilon}}{\sqrt{S_t + \epsilon}} \cdot \frac{\partial L}{\partial w_t} \tag{6}$$

where

$$D_t = \beta D_{t-1} + (1-\beta)[\Delta w_t]^2$$

$\Delta w_t = w_t - w_{t-1}$, S_t is as defined in (5), D and S are initialised to 0.

C. Hybrid algorithms

These are learning rate and gradient modifying algorithms. Examples are Adam, AdaMax, Nadam and AMSGrad. The **Adam** (or Adaptive moment estimation) is a combination of momentum and RMSprop [15]. The update equation can be represented as:

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{S_t^* + \epsilon}} \cdot V_t^* \tag{7}$$

where

$$V_t^* = \frac{V_t}{1-\beta_1^t}, \qquad S_t^* = \frac{S_t}{1-\beta_2^t}$$

$$V_t = \beta_1 V_{t-1} + (1-\beta_1)\tfrac{\partial L}{\partial w_t},$$
$$S_t = \beta_2 S_{t-1} + (1-\beta_2)\left(\tfrac{\partial L}{\partial w_t}\right)^2$$

The Adam uses the bias correction to improve the RMSprop. The authors of [15] further proposed an adaptation of the Adam optimiser called AdaMax whose update is given as:

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{S_t^*}} \cdot V_t^* \tag{8}$$

where V_t^* and S_t are as earlier defined and

$$S_t = \max\left(\beta_2 S_t, \left|\frac{\partial L}{\partial w_t}\right|\right)$$

The **Nadam** (or Nesterov-accelerated Adaptive Moment Estimation) optimiser provides a more efficient modification of the Nesterov implementation that updates the gradient one step ahead [16]. The **AMSGrad** is a variant of Adam that was proposed in 2018 [17]. The AMSGrad changes the adaptive learning rate component in Adam by monotonically reducing the stepsizes. Thus, it possesses theoretical convergence guarantees. It has been reported that despite these guarantees the AMSGrad empirically has a generalization performance that is similar to that of Adam on problem where ofAMSGrad to be similar to that of Adam on problems where a generalization gap exists between Adam and SGD. In all of the algorithms, V and S are initialized to 0.

D. Switching algorithms

Despite superior training outcomes, adaptive optimization methods such as Adam, Adegrad and RMSprop have been found to generalize poorly compared to SGD [1]. As a result, a number of researchers have proposed a hybrid strategy that begins training with one method and switches to another. For example, in machine translation, the authors in [18] used a strategy that starts with Adam then switches to SGD to implement the Google's Neural Machine Translation (GNMT) system, including all the techniques that are critical to its accuracy, speed, and robustness. In [19], a convex combination of RMSprop and SGD was used to maintain accuracy with a large minibatch size in an image classification application. The authors in [1] investigated a strategy named SWATS that begins training with an adaptive method and switches to SGD when appropriate. Specifically, SWATS algorithm switches from Adam to SGD when a triggering condition is satisfied. The authors reported closing the generalization gap between SGD and Adam on a majority of the tasks.

III. RADAMS Algorithm

A. Motivation

We propose a switching strategy that builds on the procedure proposed by [19] and [1]. We call this strategy RADAMS as it executes the **R**MSprop, **Ada**m and **s**tochastic gradient descent on schedule in that order. Our method exploits the strength of each of the algorithms. We start with RMSprop because of its reported effectiveness at the beginning of image experimental runs. We switched to Adam to take advantage of the improvement that Adam provides over RMSprop. Finally, we switch from Adam to SGD to utilize the fact that SGD generalizes better than Adam. We elaborate each of these points in the following discussion.

B. Idea 1: Starting with RMSprop

To deal with challenge of the optimization difficulty at the start of training, we exploit the observation made by the authors of [19] that starting with the RMSprop instead of the Adam performed better for the image classification task. The update rule used in [19] was a simple combination of momentum SGD and RMSprop defined as follows:

$$w_t = w_{t-1} + \eta\Delta_t$$

$$\Delta_t = \mu_1\Delta_{t-1} - \left(\alpha_{SGD} + \frac{\alpha_{RMSprop}}{\sqrt{m_t + \epsilon}}\right),$$

$$m_t = \mu_2 m_{t-1} + (1-\mu_2)g_t^2$$

The algorithm starts with RMSprop (i.e, $\alpha_{SGD} \approx 0$) and then smoothly switch to SGD using the transition schedule below:

$$\alpha_{SGD} = \begin{cases} \frac{1}{2}\exp\left(2\left(epoch - \beta_{center}\right)/\beta_{period}\right), \\ \qquad\qquad \text{if } epoch < \beta_{center} \\ \frac{1}{2} + 2\left(epoch - \beta_{center}\right)/\beta_{period} \\ \qquad\qquad \text{if } epoch < \beta_{center} + \frac{1}{2}\beta_{period} \\ 1, \qquad\qquad \text{otherwise} \end{cases}$$

Inspired by this idea, we also start with RMSprop to take advantage of the good starting behavior observed for image data. However, our approach is different in two ways. First, we didn't switch to SGD after the RMSprop schedule. Additionally, our switching schedule does not employ the exponential linear unit (ELU) activation used in [19].

C. Idea 2: RMSprop to Adam

Rather than perform all the earlier training using RMSprop like it was done in [19], we switched to Adam to enable us take advantage of the fact that the Adam is particularly developed to improve the performance of the RMSprop. Thus our motivation for this switch is to exploit the potential superior performance that Adam can give as it uses an exponential moving average step in lieu of gradient. Specifically, Adam is able to rectify the biased scaling update that characterizes the RMSprop. Also rather than use a function as done in [19], our approach uses a simple switching criteria whereby we switch to Adam after $\theta \times epochs$ number of RMSprop runs where $epochs$ is the total number of epochs. In our experiments, we set $\theta = 0.2$.

D. Idea 3: Adam to SGD

It has been pointed out that despite the superior training outcome, the Adam generalizes poorly compared to SGD. The SGD has been reported to outperform the Adam at the later stages of training because the performance of the later stagnates. Therefore, building on [1], we implement a switch to SGD from Adam at the later stage of the algorithm. This is related to the projection of Adam steps on the gradient subspace. The pseudocode of the RADAMS algorithm is illustrated below. For completeness and ease of reference, we also incorporate the strategy proposed in [1] which we applied in RADAMS.

Algorithm 1 RADAMS Algorithm

```
Input: Objective function $f$, initial point $w_0$, learning rate $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ , $phase = RMSprop$, $\theta = 0.2$, $epochs = 250$
Output: $w_k$
    Initialisation : $t \leftarrow 0$, $m_t \leftarrow 0$, $a_t \leftarrow 0$, $\lambda_t \leftarrow 0$, $p_t \leftarrow 0$
 1: while stopping criterion not met do
 2:     $t = t + 1$
 3:     Compute stochastic gradient $g_t = \nabla f(w_{t-1})$
 4:     if $numberofepoch > \theta \times epochs$ then
 5:         $phase = Adam$
 6:         $a_t = 0$
 7:     end if
 8:     if $phase = RMSprop$ then
 9:         $a_t = 0.9 * v_{t-1} + 0.1 * g_t^2$
10:         $p_t = p_{t-1} - \frac{\alpha * g_t}{\sqrt{a_t^2 + \epsilon}}$
11:         continue
12:     end if
13:     if $phase = SGD$ then
14:         $v_t = \beta_1 v_{t-1} + g_t$
15:         $w_t = w_{t-1} - (1 - \beta_1)\Lambda v_t$
16:         continue
17:     end if
18:     $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$
19:     $a_t = \beta_2 a_{t-1} + (1 - \beta_2) g_t^2$
20:     $p_t = -\alpha_t \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t} \frac{m_t}{\sqrt{\alpha_t + \epsilon}}$
21:     $w_t = w_t + p_t$
22:     if $p_t^T g_t \neq 0$ then
23:         $\gamma_t = \frac{p_t^T p_t}{-p_t^T g_t}$
24:         $\lambda_t = \beta_2 \lambda_{t-1} + (1 - \beta_2)\gamma_t$
25:         if $k > 1$ and $|\frac{\lambda_t}{1-\beta_2^t} - \gamma_t| < \epsilon$ then
26:             $phase = SGD$
27:             $v_t = 0$
28:             $\Lambda_t = \frac{\lambda_t}{1-\beta_2^t}$
29:         end if
30:     else
31:         $\lambda_t = \lambda_{t-1}$
32:     end if
33: end while
34: return $w_t$
```

Parameters:

- α: the default learning rate.
- θ:the percentage cutoff for RMSprop and Adam.
- β_1:The exponential decay rate for the first moment estimates.
- β_2:The exponential decay rate for the second-moment estimates.

Variables:

- w_t: the updated point at time step t.
- m_t: 1^{st} moment vector at time step t.
- a_t: 2^{nd} moment vector at time step t.
- p_t intermediate parameters at time step t.
- γ_t: bias-corrected exponential averaged value at time step t.
- γ_t: current exponential averaged value at time step t.
- Λ_t: initialization learning rate of SGD
- v_t 1^{st} moment vector at time step t for SGD.
- ϵ: a very small number to prevent any division by zero in the implementation .

In the following sections, we evaluate the performance of these algorithms on image and stock data.

IV. Experiments

We design computational experiments to explore the performance of some of the algorithms. We do this in four experiments. First, we look at the performance of the more popular algorithms: SGD, Momentum, RMSprop and Adam on an image classification task using the the CIFAR-10 dataset. The goal here is to see their relative performance using accuracy and time as metrics. Next, we explore the relative performance of each class of algorithms using time as the metric of interest. Specifically, we experiment on the SGD algorithm (vanilla), Momentum algorithm (momentum-based), RMSprop algorithm (Adaptive), Adam (Hybrid), SWATS (switching) and the proposed strategy, RADAMS (switching). For this experiment, we consider a handwritten digit classification problem using the MNIST datasets. In the third experiment, we compare the performance of the proposed algorithm (RADAMS) to SGD and Adam using CIFAR-100 dataset. For this experiment, the goal is to investigate the actual validation accuracy as the number of epochs increases. Consequently, we made sufficient runs to ensure convergence. Finally, to validate the claim that an algoirthm that starts with RMSprop (which in our case is the RADAMS) has been reported to do better in image data, we repeat the procedure of experiment three but this time on a stock data. The goal is to see how RADAMS performs relative to SGD and Adam on a stock data.

A. MultiClass Color Image Classification

We consider the CIFAR-10 data for this experiment. The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. Since this is a classification problem, we evaluate the performance of the model using accuracy defined as:
Accuracy = (TP + TN)/(TP + TN + FP + FN) where:

- TP = True positive
- TN = True negative
- FP = False positive
- FN = False negative

Table II summarizes the result obtained on a Dell Inspiron 7000 core i7 laptop. It can be seen that for this dataset, the accuracies of SGD and Momentum algorithm are comparable and lower than those of both RMSprop or Adam. The time taken by SGD and momentum are comparable and higher than the time taken by the RMSprop and Adam. Of all four algorithms, Adam has the best performance in terms of accuracy and time taken.

Figure 1 depicts the accuracy as a function of epochs. Again it can be seen that the Momentum and SGD algorithms follow similar path while the RMSprop and Adam are close in their performance. There is however more fluctuations at the earlier epochs in the case of Adam.

Algorithm	Class	Epochs	Acc.	Time(hr)
SGD	Vanilla	100	0.5847	43.9
Momentum	Momentum	100	0.5778	43.6
RMSprop	Adaptive	100	0.8415	40.8
Adam	Hybrid	100	0.8902	38.0

TABLE I: Performance of learning rate algorithm on CIFAR-10 data

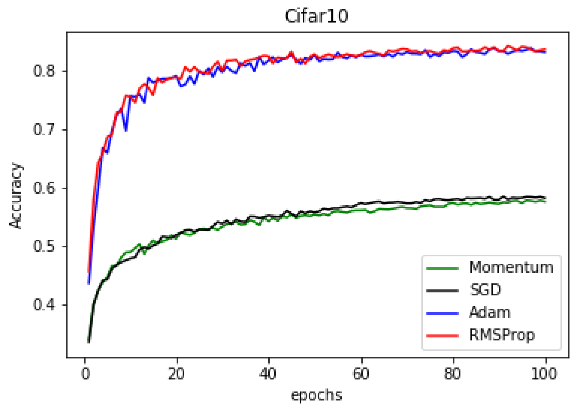


Fig. 1: Comparison of validation accuracy for CIFAR-10 data, 100 epochs

B. Handwritten Digit Classification

The second image classification task is on the MNIST dataset. The MNIST is a database of handwritten digits that has a training set of 60,000 examples, and a test set of 10,000 examples. The goal of this experiment is to measure the relative time taken by each class of algorithm and draw a conclusion on whether or not this time is the same across different data types. We run 50 epochs for at least one algorithm from each of the discussed class for comparison. Note that to access the accuracy of the algorithms, experimental runs in excess of the 50 epochs will be required. Notwithstanding, we report the maximum accuracy within the 50 epochs for completeness. For this particular data, it can be observed that the computational time of the SGD and Momentum algorithm are comparable. This is similar to our observation in the first experiment. The computational time of the RMSprop and Adam are comparable unlike our experience in the first experiment. Also, unlike our observation in the multi-class color image, the Adaptive algorithms take longer time than the momentum-based algorithm. It can therefore be inferred from this that time taken is data dependent. However, the SGD and Momentum seem to take approximately the same amount of time to run. This is not the case for the RMSprop and Adam. A quick look at the algorithms reveals why this is the case. Furthermore, for this data, the switching algorithms (SWAT and RADAMS) take a slightly higher computational time when compared with adaptive algorithm. It can however be noted that the the proposed RADAMS algorithm takes a slightly less computational time than the other switching algorithm (SWATS). The accuracy of the adaptive algorithms and the switching algorithms are comparable for this data and are much higher than the SGD or Momentum algorithm.

Algorithm	Class	Epochs	Acc.	Time (hr)
SGD	Vanilla	50	0.9709	18.0
Momentum	Momentum	50	0.9677	18.0
RMSProp	Adaptive	50	0.9941	19.3
Adam	Hybrid	50	0.9940	18.9
SWATS	Switching	50	0.9934	20.6
RADAMS	Switching	50	0.9933	20.1

TABLE II: Performance of learning rate algorithm on MNIST data

Figure 2 presents a graphical display of the summary of Table II with a focus on the validation accuracy. Here, it can be clearly observed here that the SGD and Momentum under-perform the adaptive and switching algorithms. Also note that the switching algorithm still fluctuates a lot within this range of epochs indicating that they need more runs to converge.

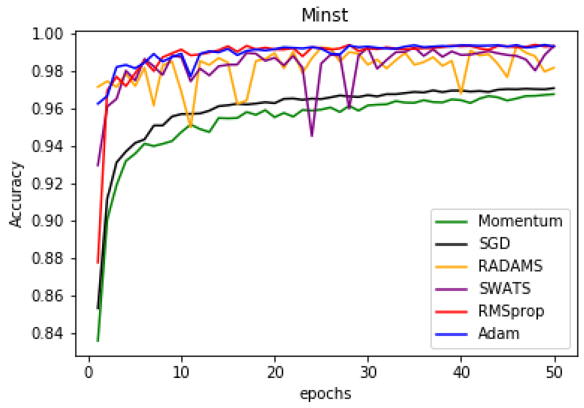


Fig. 2: Comparison of validation accuracy for MNIST data, 50 epochs

C. MultiClass Color Image Classification (II)

This dataset is similar to the CIFAR-10, except it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 superclasses. Each image comes with a ”fine” label (the class to which it belongs) and a ”coarse” label (the superclass to which it belongs). Our goal here is to investigate the accuracy of the proposed RADAMS algorithm and comapre it to SGD and

Adam. Consequently, we ran the algorithm long enough (250 epochs) to give room for the algorithms to converge. Two reasons explains why we decided to compare RADAMS to SGD and Adam. SGD is like the benchmark algorithm for neural network hence it's necssary to compare any proposed algorithm to SGD. Adam on the other hand is one of the most effective algorithm for neural network. This can be further validated by looking at the experiments in Sections IV.A and IV.B In both previous experiments, it can be seen that Adam is indeed effective. Consequently, it's reasonable to compare the proposed algorithm to the industry standard. Figure 3 gives the result obtained when SGD, Adam and RADAMS were applied on the CIFAR 100. The figures shows the proposed algorithm, RADAMs and the industry standard Adam outperform the SGD. This shouldn't come as a surprise since the algorithms were developed to improve the performance of the SGD. An interesting result on this data is that RADAMS does even better than the Adam algorithm. This shows that the proposed idea does indeed work. The outperformance started as early as after about 50 epochs. The fact that this was the observation with the MNIST data may imply that the result may vary from data to data.

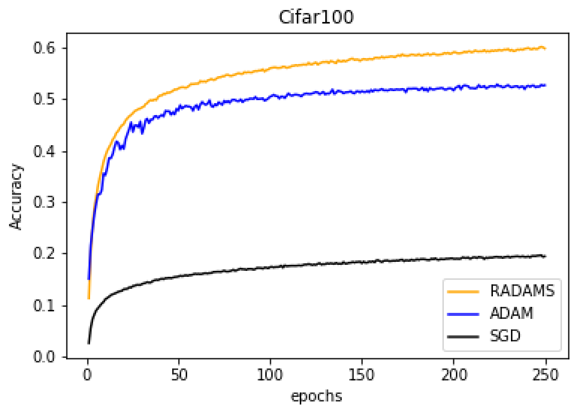


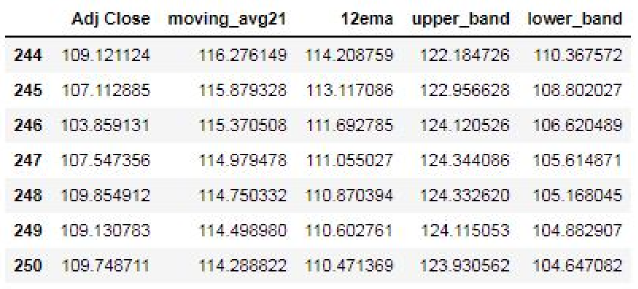
Fig. 3: Comparison of validation accuracy for Cifar100 data, 250 epochs

D. Stock Data

We analyzed stock data gleaned from yahoo finance across several years. The raw data contains the stock's open price, close price, high, low, volume and adjusted close. Since we will like to eventually use an algorithm to predict stock trend, we posed and answered a question about stock returns. Since the original datasets only has few features, to incorporate more useful information, we added some popular technical indicators used by investors and performed feature engineering to potentially improve the accuracy of the model. The features we added include 7 and 21 days Moving Average, Exponential Moving Average, Bollinger Bands and MACD. We also scaled the features using data normalization. We briefly describe each engineered feature:

- **Moving Average:** The moving average is the average of the stock prices in a certain time period.
- **Exponential Moving Average:** Exponential Moving Average is a weighted moving average that gives more weighting to recent price data.
- **Bollinger Band:** Bollinger Band is a technical analysis tool defined by the lines plotting positive and negative value of the standard deviations away from the moving average.
- **MACD:** The MACD is the difference between 26-period Exponential Moving Average and the 12-period Exponential Moving Average.

Fig. 6 below shows part of these features.



	Adj Close	moving_avg21	12ema	upper_band	lower_band
244	109.121124	116.276149	114.208759	122.184726	110.367572
245	107.112885	115.879328	113.117086	122.956628	108.802027
246	103.859131	115.370508	111.692785	124.120526	106.620489
247	107.547356	114.979478	111.055027	124.344086	105.614871
248	109.854912	114.750332	110.870394	124.332620	105.168045
249	109.130783	114.498980	110.602761	124.115053	104.882907
250	109.748711	114.288822	110.471369	123.930562	104.647082

Fig. 4: Generated Features from Stock Data

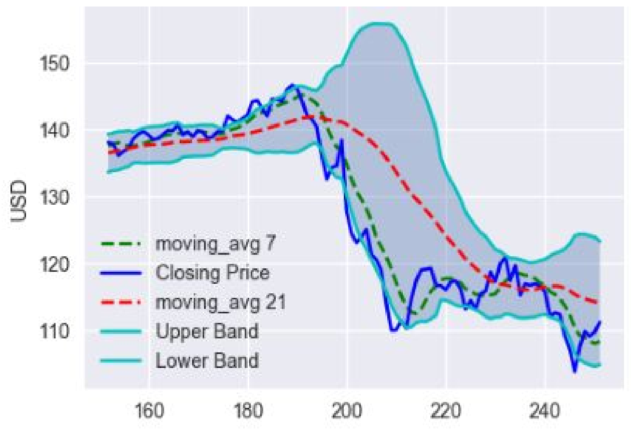


Fig. 5: Plot of Important Features

We evaluated the performance of the algorithms using MSE and computation time. As it can be seen from Table III, the Adam and RMSprop algorithms outperform the SGD which again does not come as a surprise. However, the interesting observation is the fact that the performance of Adam is comparable to the RADAMS in the stock data unlike the image data where RADAMS outperforms Adam, This is an important observation that validates the claim that RADAMS superiority may be in the space of image data. Notwithstanding, it's remarkable to find that even in a different data like the stock data, the performance of the proposed algorithm is at par with Adam which is considered the "industry" standard. Another observation is that for the stock data, all three algorithms converged to an accuracy and

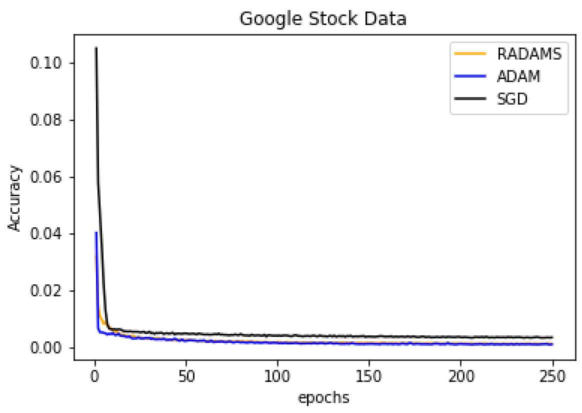


Fig. 6: Comparison of validation loss for Stock data, 250 epochs

maintained that value regardless of how long the algorithms continue to run.

Algorithm	Nos. of epochs	MSE
SGD	250	0.0034
Adam	250	0.0011
RADAMS	250	0.0011

TABLE III: Performance of learning rate algorithm on stock data

V. Discussion

The SGD is a simple yet effective algorithm that performs well across a variety of applications. However it suffers some drawbacks which adversely impact it's usefulness in certain applications. To address this, a variety of algorithms have been proposed. In this work, we grouped those algorithms into four classes namely the Momentum-based algorithm class (e.g Momentum), the adaptive learning rate algorithm class (e.g RMSprop, AdaGrad), the hybrid class (Adam, Nadam) and the switching algorithm class (e.g SWATS). Note that the although the hybrid class are learning-rate and gradient modifying algorithms, they are also often categorized alongside the algorithms in the adaptive class. The momentum class sometimes improves on the SGD while falling short in others. The adaptive class and the hybrid class provide superior performance and may be termed the ”industry standard”. However, the insufficiency of the adaptive and Hybrid class have been pointed out in [8] . Precisely, it has been observed that despite their effectiveness, they suffer from generalization gap due to the scaling of the per-variable learning rates. It turns out that the remedy to this actually lies with where it all started from which is the SGD. SGD have been reported to provide a level of generalization that is not attained by the adaptive and Hybrid algorithms. Consequently, researchers are beginning to look at the strategy of switching from one algorithm to the other within an experiment. In [19], the switch was from RMSprop to SGD. It was noted that the RMSprop has nice initial starting behaviour especially for image data. The authors used this idea to develop a robust algorithm that does well in image data. Motivated by the observation that adaptive algorithms retain the adaptivity and non-uniform gradient scaling and are expected to suffer from generalization issues, authors of [1] proposed a switching algorithm that starts with Adam and later transition to SGD. Inspired by the results obtained by these authors, we used a hybrid strategy that exploits the strength of each of these algorithms to propose a switching algorithm that has yieled promising results on image data as well as stock data. Specifically, knowing that RMSprop does well at the start of the training with image data, we design our strategy to begin with RMSprop. However, we also know that Adam was developed as an improvement for RMSprop. Rather continuing with RMSprop we switched to Adam after some iterations to take advantage of the potential superior performance that it gives. Finally, we switched from Adam to SGD to take utilize SGD superior generalization capability. The ideas though simple gives results that are at least as good as Adam in all the experiments we performed. We validated this algorithm using the CIFAR-10, MNIST, CIFAR-100 and stock data. in CIFAR-100 in particular, the proposed algorithm clearly outperforms Adam. While the focus has been on Adam, we will like to report that we also observed a slight edge of our algorithm over the SWATS algorithm which only does a switching from Adam to SGD. We could not explore more data types because we had limited computing capacity and time as some of the algorithms took about a week to run on our computer. However, the result so far has been very promising.

VI. Conclusion/Future Work

In this project, we investigated learning rate algorithms used in neural networks. We surveyed current state of the art algorithms and divided them into four classes. We designed some computational experiments to explore the performance of some of one or two algorithms from each class of algorithms and evaluated their performance using accuracy and speed as metrics. We proposed a new strategy which we call RADAMS that builds on previously documented performance of RMSprop, Adam and SGD. We validated our proposed strategy on four dataset consisting of three image data and a stock data. Our results shows that the proposed algorithm outperforms the Adam and SGD in image data it was trained on. This validates the underpinning assumption made in developing the algorithm which is the fact that RMSprop seems to do well for image data at the beginning of the training and the initial gain can be carried over to the rest of the experimental run by switching to Adam and later SGD. We also compared the proposed algorithm to Adam and SGD on stock data and found their performance to be comparable.

In the future, it will be necessary to perform more computational experiments across other data types to see how the proposed algorithm performs relative in those applications. In developing our algorithm, we suddenly we switched from RMSprop to Adam after θ epochs. Rather than such sudden switch, a viable future work would be to investigate a smoother transition between RMSprop and Adam as well as from Adam to SGD. [1] noted that the later case can be achieved by using a convex combination of the SGD and [19] as well as a gradual increase of the SGD contribution by a criterion. Finally, we note that in both our strategy and earlier works, a robust theoretical justification for why RMSprop does well (which translates to the robust performance of the proposed algorithm) for image data at the beginning of the experiment wasn't provided. A future direction could be to provide such mathematical/theoretical justification and proof.

References

[1] N. S. Keskar, R. Socher, *"Improving Generalization Performance by Switching from Adam to SGD"*, arXiv:1712.07628, 2017.

[2] H. Robbins, S. Monro, *"A stochastic approximation method"*, The annals of mathematical statistics, pp. 400–407, 1951.

[3] S. Merity, C. Xiong, J. Bradbury, and R. Socher *"Pointer sentinel mixture models"*, arXiv:1609.07843, 2016.

[4] K. He, X. Zhang, S. Ren, and J. Sun *"Deep residual learning for image recognition"*, arXiv:1512.03385, 2015.

[5] I. Loshchilov, and F. Hutter *"SGDR: Stochastic gradient descent with warm restarts"*, 2016.

[6] J. Lee, M. Simchowitz, M. I. Jordan, and B. Recht, *"Gradient descent converges to minimizers"*, University of California, Berkeley, 1050:16, 2016.

[7] M. Hardt, B. Recht, and Y. Singer, *"Train faster,generalize better: Stability of stochastic gradient descent"*, arXiv:1509.01240, 2015.

[8] A. C. Wilson, R. Roelofs, R. Stern, M. Srebro, and B. Recht, *"The Marginal Value of Adaptive Gradient Methods in Machine Learning"*, ArXiv e-prints, May 2017.

[9] S. Mandt, M. D. Hoffman, and D.M. Blei, *"Stochastic Gradient Descent as Approximate Bayesian Inference"*, ArXiv e-prints, April 2017.

[10] B.T.Polyak, *" Some methods of speeding up the convergence of iteration methods"*, AUSSR Computational Mathematics and Mathematical Physics, 4(5):1–17, 1964.

[11] I. Sutskever, J. Martens, G. Dahl and G. Hinton, *"On the importance of initialization and momentum in deep learning"*, Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013.

[12] J. Duchi, E. Hazan, and Y. Singer, *"Adaptive subgradient methods for online learning and stochastic optimization"*, The Journal of Machine Learning Research, 12:2121– 2159, 2011.

[13] T. Tieleman, and G. Hinton, *"RMSprop: Divide the gradient by a running average of its recent magnitude"*, COURSERA:NeuralNetworksforMachineLearning, 4, 2012.

[14] M. D. Zeiler, *" ADADELTA: An Adaptive Learning Rate Method"*, arXiv:1212.5701v1 , 2012

[15] D. P. Kingma, and J. L. Ba, *" Adam: a Method for Stochastic Optimization"*, International Conference on Learning Representations, 1–13, San Diego, 2015

[16] T. Dozat, *" Incorporating Nesterov Momentum into Adam"*, International Conference on Learning Representations, 2016

[17] J.S. Reddi, S.Kaly and S. Kumar, *"On the Convergence of Adam and Beyond"*, International Conference on Learning Representations, 2018

[18] Y. Wu, M. Schuster, Z. Chen, Q. Le, M. Norouzi, W. Macherey, M. Krikun *et al* *"Google's neural machine translation system: Bridging the gap between human and machine translation"*, arXiv:1609.08144, 2016.

[19] T. Akiba, S. Suzuki, and K. Fukuda, *"Extremely large minibatch SGD: Training resnet-50 on ImageNet in 15 minutes"*, arXiv:1711.04325, 2017.