# FASFA: A Novel Next-Generation Backpropagation Optimizer

Philip Naveen

June 26, 2022

## 1 Abstract

This paper introduces the fast adaptive stochastic function accelerator (FASFA) for gradient-based optimization of stochastic objective functions. It works based on Nesterov-enhanced first and second momentum estimates. The method is simple and effective during implementation because it has intuitive/familiar hyperparameterization. The training dynamics can be progressive or conservative depending on the decay rate sum. It works well with a low learning rate and mini batch size. Experiments and statistics showed convincing evidence that FASFA could be an ideal candidate for optimizing stochastic objective functions, particularly those generated by multilayer perceptrons with convolution and dropout layers. In addition, the convergence properties and regret bound provide results aligning with the online convex optimization framework. In a first of its kind, FASFA addresses the growing need for diverse optimizers by providing next-generation training dynamics for artificial intelligence algorithms. Future experiments could modify FASFA based on the infinity norm.

## 2 Introduction

We live in a society that continues use artificial intelligence in devices across the globe. The artificial intelligence among us that drives the functionality of various commercial, open-sourced, and private applications for computers and handheld devices uses real-world data in order to make inferences. Examples of how such algorithms are used today range from facial recognition software, vaccine and drug discovery [17], cancer classification [14], and traffic management systems [13]. The algorithms that help secure our data and create new medicines could be improved however, and they are mostly based on online convex optimization [7].

Suppose a given mathematical object can be mapped onto a plane using two or more dimensions. Such an object has an existing derivative for $\mathbf{R}$ across at least one of those axes, implying a minimum/maximum value of $\min_{\theta \in \mathbf{R}} f(\theta)$ or

$\max_{\theta \in \mathbf{R}} f(\theta)$ exist in at least one of those dimensions with some kind of multidimensional concavity. Locating these extreme values is one of the key drivers for search algorithms that optimize the parameters of designs for engineering. Problems encountered in the real world can often be interpreted as an optimization of an objective function in order to estimate a minimum/maximum in respect to its parameters [7].

Stochastic gradient descent (SGD) optimization is an useful and effective technique finding the extreme values. Many of these objective functions comprise of sub-functions that exhibit different patterns depending on the particular data sub-sampling, giving them some stochasticity [2]. However, SGD on is not always the best method for minimization/maximization on noisy situations created by factors such as dropout layers in a multilayer perceptron [9]. The purpose of this project was to introduce a new method of stochastic optimization for problems high in variables and parameters.

This paper introduces the fast adaptive stochastic function accelerator (FASFA) for estimations. It requires only the first order gradients. The method uses two exponential decay rates that adapt based on the first and second moment estimates. The momentum-based optimizer was designed to improve the performance of Adam [1], the most commonly used optimizer in multilayer perceptrons.

[1]Adam stands alone as an optimizer, but it has many subtypes including AdaMax, and NAdam. FASFA's performance will be compared only to the standard Adam method.

**Algorithm 1** Fast adaptive stochastic function accelerator (FASFA) for stochastic optimization of objective functions. The $g_t^2$ denotes for the element-wise $g_t \otimes g_t$ product from squaring. For hyperparameterization, $\alpha = 0.001$, $\mu = 0.8$, and $\nu = 0.999$ are some good starting variables. The subscripts and superscripts denote the iteration to convergence and power. Hats denote the updated moment estimates. Note that $\xi_{decay} = |\mu + \nu|$.

```
Require: $\mu$: First order momentum decay estimate
Require: $\nu$: Second order momentum decay estimate
Require: $\alpha$: Learning rate
Require: $f(\theta)$: Stochastic and objective function for loss minimization
Require: $\theta$: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize first moment vector)
  $n_0 \leftarrow 0$ (Initialize second moment vector)
  $t \leftarrow 0$ (Initialize timestep/iteration)
  while Unconverged do
    $t \leftarrow t + 1$ (Iterates by $t$ until convergence step $T$)
    $g_t \leftarrow \nabla_\theta f(\theta_{t-1})$ (Getting gradients for stochastic objective at timestep $t$)
    $m_t \leftarrow \mu m_{t-1} + (1-\mu)g_t$ (Update biased first moment estimation)
    $n_t \leftarrow \nu n_{t-1} + (1-\nu)g_t^2$ (Update biased second moment estimation)
    $\hat{m} \leftarrow \frac{\mu m_t + g_t(1-\mu^t)}{1-\mu^t}$ (First raw moment estimate)
    $\hat{n} \leftarrow \frac{\nu n_t + g_t^2(1-\nu^t)}{1-\nu^t}$ (Second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \frac{\hat{m}_t \xi_{decay}}{\sqrt{\hat{n}_t} + \epsilon}$ (Implement FASFA update rule)
  end while
  return $\theta_t$ (Resulting parameter for objective function)
```

This is the workflow of the iterative training dynamics until convergence. The basic notation for the hyperparameters is shown here. So basically, the method retains the order of updates for the first and second moment estimates while augmenting the actual operations that occur. This means that the method is simple and easy to use. The average programmer working with gradient descent typically defaults to Adam, ergo retaining the structural properties in FASFA would increase its usability in practice. The optimizer is a novel invention, but structurally, it is almost akin to the current meta and should work with Adam's default parameters of $\alpha = 0.001$, $\mu = 0.8$, and $\nu = 0.999$ [9].

Throughout the experiments surrounding the FASFA optimizer, the independent variable is the method of stochastic optimization. It is hypothesized that if indeed FASFA is used to optimize an algorithm to locate $\min_{\theta \in \mathbf{R}} f(\theta)$ on objective function $f(\theta)$, then it would result in the greatest distance traversed on that aforementioned three-dimensional mathematical object.

The purpose of this project was to design and test a new method of optimizing stochastic objective functions. Section 3 describes the properties of the optimizer, section 4 describes the experiments testing it out, section 5 analyzes the data, and section 6 has some conclusions. There is also an appendix with some extra information.

## 3 FASFA

Algorithm 1 shows the steps outlining the FASFA across iterative training. The objective function with some level of noise is denoted $f(\theta)$ differentiable with reference to parameter $\theta$. The expected value of the function $\mathbf{E}[f(\theta)]$ is to be minimized across the convex surface in the parameter space, which in the case of deep-learning is often the relation between synaptic weight values. With sequence of $f_1...f_T$ where $T$ is the timestep of convergence. The gradient or the vector of partial derivatives is denoted by $g_t = \nabla_\theta f_t(\theta)$ where $t$ represents a timestep between initialization and convergence/termination.

Using the two hyperparameters $\mu, \nu \in [0, 1)$ FASFA updates the moving average of the gradient $m_t$ and squared gradient $n_t$ by controlling the decay rate of the moving averages. Those are estimated using the first and second raw moment estimates. Opposed to randomly initializing the vectors of moving averages, they are initialized to 0s such that they begin at a similar starting point. Like other algorithms [9,10], FASFA combats the bias towards 0 throughout the optimizer by implementing the bias correction step for the correct estimates $\hat{m}$ and $\hat{n}$.

Let $g_1...g_t$ be representative of the sequence of past gradients on iteration $t$. The second moment estimate $n_t$ can be written as a summation of the previous gradients

$$n_t = (1-\nu)\sum_{t}^{i=1} \nu^{t-1} \cdot g_i^2$$

and the timestep $t$ for the expected value there of $\mathbf{E}[v_t]$ is known to relate to the second true moment $\mathbf{E}[g_t^2]$. Through rearranging the terms and substituting the variables, the implemented bias correction becomes

$$\mathbf{E}[n_t] = \mathbf{E}\left[(1-\nu)\sum_{t}^{i=1} \nu^{t-1} \cdot g_i^2\right] = \mathbf{E}[g_t^2](1-\nu)\sum_{t}^{i=1} \nu^{t-1} + \zeta = \mathbf{E}[g_t^2](1-\nu^t) + \zeta$$

where $\gamma$ remains close to 0 for when $\mathbf{E}[g_t^2]$ remains stationary. This step is also conducted with $\hat{m}$, giving the following bias correction of

$$\mathbf{E}[m_t] = \mathbf{E}[g_t](1-\mu^t) + \zeta$$

with $\mu$ and a non-squared gradient for the formal update rules in Algorithm 1.This indeed replicates the effect of disassembling the bias from the vectors of 0s. This system of bias correction is used in various other Quasi-Newton methods, so it retains the same effect here [9,10].

The online convex optimization framework is where an algorithm makes a decision over some iterations. After a decision is made, a loss is given [7]. FASFA falls under this framework because its decisions are indeed bounded with a finite amount of decisions [7]. What usually occurs is that after the algorithm makes a decision, and then the convex cost function is revealed. The main metric of this system is the regret of the player [3, 7]. Regret is defined as

$$R(T) = \sum_{t=1}^{T} f_t(\theta_t) - \inf_{\theta^* \in \chi} \sum_{t=1}^{T} f_t(\theta^*)$$

and often times, it is beneficial to locate the upper bound of the function. This is because it shows the worst possible case situation, with the maximum possible regret. In this case, the regret bound is

$$R(T) \leq \frac{D^2}{2\alpha\xi_{decay}(1-\mu)} \sum_{i=1}^{d} \sqrt{T\hat{v}_{T,i}} + \frac{(1+\mu)\alpha G_\infty}{(1-\mu)\sqrt{1-\nu}(1-\gamma)^2} \sum_{i=1}^{d} \|g_{1:T,i}\|_2 + \sum_{i=1}^{d} \frac{D_\infty^2 G_\infty \sqrt{1-\nu}}{2\alpha\xi_{decay}\mu(1-\lambda)^2}$$

which is similar to the results delivered by other reliable optimization techniques such as AdaGrad, RMSProp, and Adam [4, 8, 9]. The entire proof is in the appendix. Observing the denominator of the regret bound shows that a low user controlled decay sum $\xi_{decay}$ and learning rate $\alpha$ guarantee a lower regret bound, but may result in a slower overall convergence and vice versa [7].

The algorithm is defined on both Adam and Nesterov-based momentum optimization. It integrates Adam's workflow, but implements the Nesterov-enhanced momentum. However, unlike other algorithms that implement it such as NAdam, FASFA uses it on both $\hat{m}$ and $\hat{n}$ [6, 12]. Furthermore, it still has two bias correction terms like Adam [9]. So FASFA basically hybridizes ideal properties of optimizers such as Adam that favor high mini batch size and datapoints and optimizers such as Nesterov gradient descent that work well with low learning rates and mini batch size in order to work well in both worlds. In addition, it retains the same ideal low signal to noise ratio of $\frac{\hat{m}_t}{\sqrt{\hat{m}_t}}$ of other methods such as Adam [9], Eve [10], and AMSGrad [15].

## 4 Experiments

### 4.1 Materials and Safety

One Acer Nitro 5 laptop with an Intel Core i7 processor, NVIDIA GeForce RTX 3050 graphics card, and dual M.2 solid state drive was obtained for the procedures. An Oracle VirtualBox virtual machine running bidirectionally functional sharing between the Windows 11 host and Ubuntu Linux distribution was installed. The procedures were conducted using custom C++ and Python3 computer code. The JupyterLab and Visual Studio Code text editors and information development environments were used. Excel was used to generate all of the graphs.

There were minimal safety risks, as the procedures were all conducted on a computer. To prevent eye strain, this project was only worked on for 3 hour intervals during the day, with 50% of all blue light on the display being filtered out any time working after 8 PM.

## 4.2 Convex Surface Testing

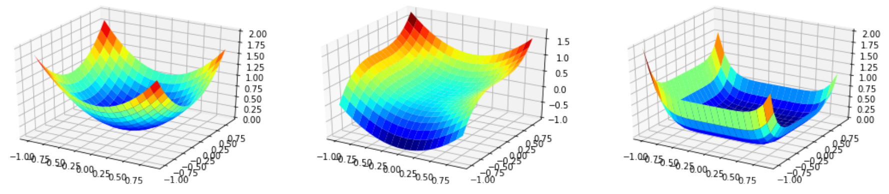

Figure 1: Convex Surfaces for Objective Functions used in Testing Search Algorithms

In this first experiment, three convex surfaces were generated for locating extrema. The surfaces represented possible objective functions of $f_t(\theta)$. The vertical dimension was the one of interest. The surfaces used were fabricated using multivariable objective functions. They took parameters $x$ and $y$, augmented them, and returned them. The first returned $x^2$ and $y^2$, the second returned $x^3$ and $y^2$, and the third one returned $x^4$ and $y^4$.For a given optimizer, Mersenne twister generator was used to select psuedorandom points on each graph.

Each optimizer used the recommended hyperparameterization from their original papers or from subsequent projects detailing experiments using them. For FASFA, the same settings recommended for Adam were used. For 100 iterations, $\alpha = 0.001$, $\beta_1 = \mu = 0.8$, and $\beta_2 = \nu = 0.999$, the optimizers were tracked on the surfaces. For Adam and FASFA, the first and second moment estimates were also tracked.

## 4.3 Multilayer Perceptron Testing

For image-based testing using backpropagation, MNIST [11] numbers, CIFAR-10 [5], and MNIST fashion [16] were used. One testing model of three blocks using convolution, max pooling, and dropout layers in addition to a multilayer perceptron using one flattening layer and three dense layers of 64, 32, and 10 layers was fabricated. It used the ReLU activation function in the hidden layers and SoftMax in the output layer. Once again, $\alpha = 0.001$, $\beta_1 = \mu = 0.8$, and $\beta_2 = \nu = 0.999$ was used. Furthermore, mini batch size was 100 and the loss function was sparse categorical crossentropy. For 10 epochs, SGD, AdaGrad, RMSProp, Adam, and FASFA were used in identical versions of the model described earlier on each of the three datasets. At the end of each epoch, a cross-validation using 80% training and 20% testing split was calculated using a confusion matrix. The values were then recorded for each optimizer.

For testing the effect of the learning rate on loss minimization for stochastic optimization, synthetic data was used. Crest shaped patterns of data were generated using Sci-Kit Learn. The Sci-Kit Learn multilayer perceptron framework was used to replicate the same model from earlier, excluding the convolution, max pooling, and dropout layers. Once again, $\beta_1 = \mu = 0.8$, and $\beta_2 = \nu = 0.999$ were the decay rate hyperparameters. The learning rate, or step size was incremented by a factor of 0.001 across 10 intervals. SGD, Momentum, Nesterov, AdaGrad, RMSProp, Adam, Adam1, Adam2, and FASFA were tested, as they all use a learning rate of some sort. The number of samples generated was 50000, with the same 80% training and 20% testing split for validation. The cross-validation was calculated using the same confusion matrix and the accuracy values were recorded for analysis.

## 5 Results

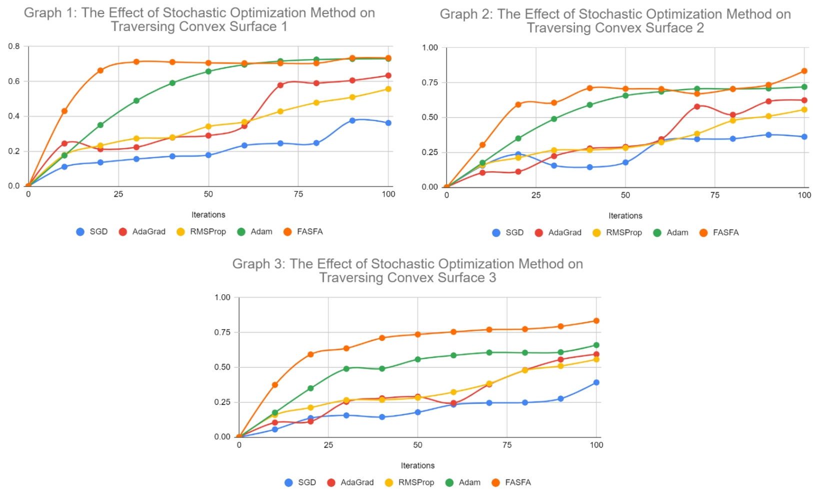

Figure 2: Vertical Traversal of Gradient Descent Methods for the Coalescing Convex Points on Three Surfaces

Figure 1 shows the relationship between optimizers and magnitude of movement on the third axis. These are the results of the first experiment. The line graphs do not act in respect to time, but based on the number of iterations. Bar graphs at each interval were constructed as well, but because of the space limitations, those were omitted.

Graphs 1, 2, and 3 show a shared trend between the techniques used. SGD carried the lowest magnitude of descent. RMSProp and AdaGrad followed, with similar performance on minimization. Adam outperformed all three of these methods by quite a significant margin. Finally, FASFA performed the best. Notably, the early stages of training showed a dramatic shift in descent for FASFA, with those values sometimes existing at almost four times the ones for SGD, AdaGrad, RMSProp, and Adam.

In terms of stability, the expected values from past research suggested that SGD would be the least stable optimization technique while Adam would be the most. SGD exceeded past research claims in terms of stability, and Adam retained its claims. However, AdaGrad seems to be rather unstable under these circumstances. Notably, despite FASFA having a high jump in movement early on, it remains relatively stable throughout the later iterations. These trends are especially apparent in graph 1.

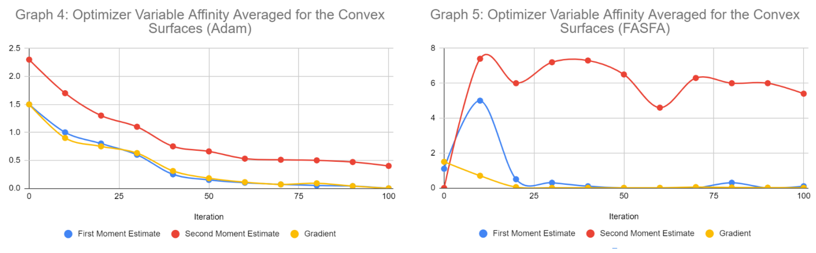

Figure 3: Comparison of Adam and FASFA With Exact Same Hyperparameters, Where $\beta_1, \beta_2$ are Functionally the $\mu, \nu$

In figure 2, the comparison between Adam and FASFA was made. The other optimizers were omitted because they do not use two exponential decay rates. Separate line graphs for each variable were also constructed, but were combined to save space. Adam is shown in graph 1, and FASFA in graph 2.

The first moment estimate showed a decay in both graphs 4 and 5. FASFA showed to have an initial spike on all three surfaces, while this trend was not apparent in Adam. However, by the first quarter of timesteps, the first moment estimate was significantly less in FASFA than in Adam. The second moment estimate showed an interesting trend in FASFA. While in Adam, it decays in a similar pattern to gradient, in FASFA it shoots upward and plateaus. The graph shows dips and spikes, but that is likely because of the inherit variability of the surfaces. Towards the end, FASFA does exhibit some decay in this term, but not to the extend that is in Adam.

Gradient in respect to the vertical distance shows a similar pattern between Adam and FASFA. Adam's gradient decays as expected, indicating that a minimum of sorts on the objective function is being approached through the same logic as the first derivative test. However, FASFA's gradient approaches 0 faster than Adam's. This offered some convincing evidence that even with Adam's default parameters, FASFA still performs at an increased rate.

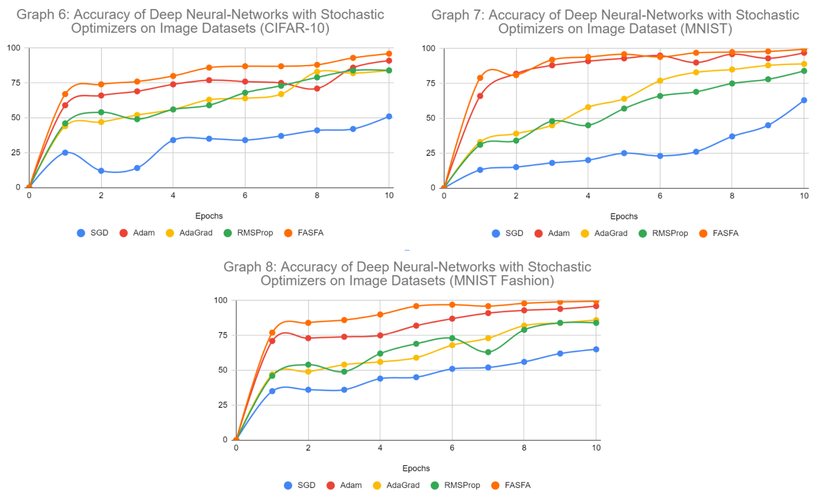

Figure 4: Progression of Accuracy in multilayer perceptrons for the Different Methods of Stochastic Optimization

Figure 3 depicts the results of the cross-validation using multilayer perceptrons. This was the result of the third experiment. Graphs 6, 7, and 8 depict the increase in accuracy as the networks trained. These graphs are not in respect to time, but into the epochs trained. The independent bar graphs were omitted because of the space limitations.

The trends seen in graphs 1, 2, and 3 remained prevalent. The graphs showed that once again, FASFA produced the best results the fastest. SGD showed to perform the worst. Furthermore, AdaGrad and RMSProp remained close to one another with Adam being the closest to matching FASFA. Though FASFA and Adam typically ended the training with accuracy values above 90%, FASFA reached that point earlier on. Furthermore, in each instance, the final accuracy value for FASFA was always higher than that of Adam.

One difference between this experiment and the first was that AdaGrad was more stable than RMSProp, which can be seen on graph 8. Across the three datasets, the performance was rather consistent, and the training stability of FASFA in particular seems to be similar to Adam, but it just converges faster, resulting in the higher accuracy.

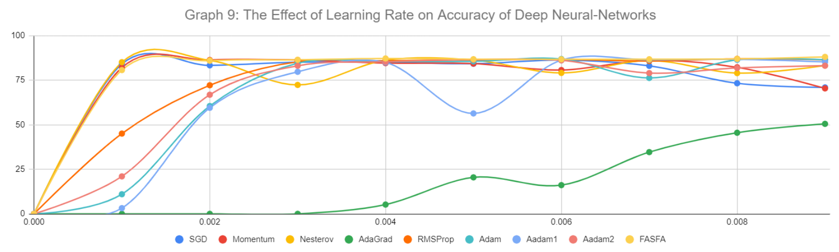

Figure 5: Trends of Accuracy Across Learning Rates from Multilayer Perceptrons

The figure above shows the accuracy trends resulting from backpropagation using various optimizers. The stability across the learning rate being incremented by a factor of 0.001 across ten intervals. The individual bar graphs for each interval were omitted because of space limitations.

In terms of accuracy, most of the networks were approximately close. FASFA remained as one of the top scoring algorithms. Adam and Adam2 also showed a similarly high performance. Adam1 and AdaGrad performed the worst. In the beginning, SGD, Momentum, and Nesterov showed they favor low learning rates. AdaGrad showed to increase in performance with a higher one. Algorithms such as Adam, Adam2, and FASFA showed to retain their capabilities with a diverse set of commonly used learning rates.

It is shown in graph 9 that SGD, Momentum, Adam, Adam2, and FASFA were relatively stable. Nesterov and RMSProp, for the most part, remained somewhat stable, apart from the extremes. AdaGrad and Adam1 were the least stable by far, with Adam1 having an almost oscillatory effect between 0.004 and 0.006 learning rate.

## 6 Discussion and Conclusions

The purpose of this project was to develop a new method of stochastic optimization that is both innovative and leverages existing methods. The method, like many others before it, is inspired by SGD. Other researchers have also attempted to introduce new mechanisms into the Adam framework to address its weaknesses. The attempts included things such as adding proportion terms and implementing Nesterov momentum [6, 10, 15].

FASFA uses Nesterov momentum but retaining the bias correction term from Adam [9, 12]. In addition, it also features a control term $\xi_{decay}$ that gives either progressive or conservative updates depending on the sum of $\mu$ and $\nu$. Furthermore, FASFA uses Nesterov momentum on both $\hat{m}$ and $\hat{n}$ for estimating diagonals from the Fischer information matrices. Despite innovating in these areas, FASFA keeps the helpful properties from the current meta of optimization, Adam. This is most present in the hyperparameters, which would make implementing the method simple and effective for a new user that is not aware of how convex optimization techniques work. In addition, the order of computations is concrete between Adam and FASFA [9].

FASFA is a novel invention that excels with low learning rates and mini batch sizes, while retaining respectable performance and stability using high ones. In all three experiments, FASFA showed promise even when using other optimizers' default settings. The learning rate experiments showed that its training stability was remarkably high even with high step sizes.

Some flaws within the experiments could lie within the limited testing algorithms. Stochastic gradient descent for artificial intelligence is used most frequently with the multilayer perceptrons [2]. Therefore, the scope of the project focused completely on these methods. Although some basic surface testing was done using the optimizer outside of the multilayer perceptrons, it was still rather limited. Based on these results and the convergence analysis, the optimizer will theoretically work and deliver strong performance on any model that wants to minimize loss, which is the vast majority of artificial intelligence algorithms [2]. However, scientific validation should be conducted to show evidence for this claim.

Future research could include a myriad of things such as testing the optimizer in a more traditional machine-learning setting. This could include pairing it with various regression or classification algorithms such as support vector machines, k-nearest-neighbors, isolation forests, and k-means clustering. FASFA's regret bound proof is based on Adam's. However, some researchers have shown arguments towards the original regret bound for Adam being incorrect and proposing their own regret bounds [1, 4]. Another future study could involve deriving FASFA's regret bound using the alternate methods and comparing them to the one shown in this project. Furthermore, there exists a derivative of Adam based on the infinity norm called AdaMax [9]. Since FASFA follows the same structure as Adam, a derivative of it following the infinity norm could also be feasible.

## References

[1] Ahmet Alacaoglu, Yura Malitsky, P. Mertikopoulos, and Volkan Cevher. A new regret analysis for adam-type algorithms. *ArXiv*, abs/2003.09729, 2020.

[2] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, 2010.

[3] Sébastien Bubeck. Theory of convex optimization for machine learning. *ArXiv*, abs/1405.4980, 2014.

[2]Only basic versions of the algorithms were constructed either from scratch or in primitive prototypes in PyTorch or Sci-Kit Learn that just have the mathematical backing of the optimizer. In the future, MATLAB or TensorFlow verions could be made for larger scale experiments.

[4] Alexandre D'efossez, Léon Bottou, Francis R. Bach, and Nicolas Usunier. On the convergence of adam and adagrad. *ArXiv*, abs/2003.02395, 2020.

[5] Raveen Doon, Tarun Kumar Rawat, and Shweta Gautam. Cifar-10 classification using deep convolutional neural network. *2018 IEEE Punecon*, pages 1–5, 2018.

[6] Timothy Dozat. Incorporating nesterov momentum into. 2015.

[7] Elad Hazan. Introduction to online convex optimization. *Found. Trends Optim.*, 2:157–325, 2016.

[8] Geoff Hinton. Visualizing optimization algos. *Coursera*.

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[10] Jayanth Koushik and Hiroaki Hayashi. Improving stochastic gradient descent with feedback. *ArXiv*, abs/1611.01505, 2016.

[11] Yann LeCun and Corinna Cortes. The mnist database of handwritten digits. 2005.

[12] Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E. Hopcroft. Nesterov accelerated gradient and scale invariance for adversarial attacks. *arXiv: Learning*, 2020.

[13] Fareed Sheriff. Elmopp: An application of graph theory and machine learning to traffic light coordination. *ArXiv*, abs/2106.10104, 2021.

[14] Dr.Rajasekaran Subramanian, Dr.Devika Rubi, R. Lakshmi, Priya Jain, and Sai Rohith Kanneganti. Breast cancer lesion detection and classification in radiology images using deep learning. 2021.

[15] Phuong T. Tran and Le Trieu Phong. On the convergence proof of amsgrad and a new version. *IEEE Access*, 7:61706–61716, 2019.

[16] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *ArXiv*, abs/1708.07747, 2017.

[17] Dylan Zhuang and Ali K. Ibrahim. Deep learning for drug discovery: A study of identifying high efficacy drug compounds using a cascade transfer learning approach. *Applied Sciences*, 2021.

## 7 Appendix

### 7.1 Regret Bound

In this section, an attempt at determining the regret bound is shown. Adam and FASFA follow many of the same rules, so the proof is similar as well. To begin, the mathematical object that is being traversed has some basic properties for this convergence to work. The main one is that it cannot have any irregular non-convexity. First it is defined that $f : R^d \to R$ is convex for $x, y \in R^d$. When $\lambda \in [0, 1]$,

$$\lambda f(x) + (1 - \lambda) f(y) \geq f(\lambda x + (1 - \gamma) y)$$

holds. This surface can be lower bound by a tangential hyperplane. Assuming the surface is convex for $x, y \in R^d$, then

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

holds [9]. These definitions can be used as an advantage when upper bounding possible regret by replacing the tangential hyperplane with one generated by the update rules from algorithm 1.

The next two proofs FASFA shares in common with Adam because they share the bias correction step. To make notation easier to understand, some simplifications are going to be made. Let $g_t \triangleq f_t(\theta_t)$, and $g_{t,i}$ be the gradient at element $i$. Logically, $g_{1:t,i} \in \mathbf{R}^t$ represent the sequence of gradients over the iterations until $t$. To illustrate, $g_{1:t,i} = [g_{1,i}, g_{2,i} ... g_{t,i}]$ [9].

Suppose $g_{1:t,i}$ exists and is bound within the gradient intervals $\|g_t\|_2 \leq G$ and $\|g_t\|_\infty \leq G_\infty$. All things considered, inequality

$$\sum_{t=1}^{T} \sqrt{\frac{g_{t,i}^2}{t}} \leq 2G \|g_{1:t,i}\|_2$$

can be proved by inducting over iterations in the series until $T$. For $T = 1$ as the first instance of a natural number in the iterations, $\sqrt{g_{1,i}^2} \leq 2G \|g_{1:t,i}\|_2$ becomes the base case. The cumulative inductive step becomes

$$\sum_{t=1}^{T} \sqrt{\frac{g_{t,i}^2}{t}} = \sum_{t=1}^{T-1} \sqrt{\frac{g_{t,i}^2}{t}} + \sqrt{\frac{g_{T,i}^2}{T}}$$

and then

$$\sum_{t=1}^{T} \sqrt{\frac{g_{t,i}^2}{t}} \leq 2G_\infty \sqrt{\|g_{1:T,i}\|_2^2 - g_T^2} + \sqrt{\frac{g_{T,i}^2}{T}}$$

after some more basic replacements. Because it is known that $\|g_{1:T,i}\|_2^2 \leq \|g_{1:T,i}\|_2^2 + \left(g_{T,i}^4 / 4 \|g_{1:T,i}\|_2^2\right)$, the subsequent step of the derivation is achieved by taking the square root of both sides of the inequality. This brings it to

$$\sqrt{\|g_{1:T,i}\|_2^2 - g_T^2} \leq \|g_{1:T,i}\|_2 - \frac{g_{T,i}^2}{2 \|g_{1:T,i}\|_2}$$

and then

$$\sqrt{\|g_{1:T,i}\|_2^2 - g_T^2} \leq \|g_{1:T,i}\|_2 - \frac{g_{T,i}^2}{2\sqrt{TG_\infty^2}}$$

which readies the inequality for the final step. So

$$G_\infty \sqrt{\|g_{1:T,i}\|_2^2 - g_T^2} + \sqrt{\frac{g_{T,i}^2}{T}} \leq 2G_\infty \|g_{1:T,i}\|_2$$

is arrived at after substituting $\sqrt{\|g_{1:T,i}\|_2^2 - g_T^2}$ and rearranging the terms. Looking backwards, it can be seen that this indeed works under the two introductory definitions about the concavity of the surfaces, which was at the beginning of this section [9].

Beginning with the summation

$$\sum_{t=1}^{T} \frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{n}_{t,i}}} = \sum_{t=1}^{T-1} \frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{n}_{t,i}}} + \frac{\hat{m}_{T,i}^2}{\sqrt{T\hat{n}_{T,i}}}$$

from the inductive inequality, the last term can be expanded to

$$\sum_{t=1}^{T} \frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{n}_{t,i}}} \leq \sum_{t=1}^{T-1} \frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{n}_{t,i}}} + \frac{\sqrt{1-\nu^T}}{(1-\mu^T)^2} \frac{(\sum_{k=1}^{T}(1-\mu)\mu^{T-k}g_{k,i})^2}{\sqrt{T\sum_{j=1}^{T}(1-\nu)\nu^{T-j}g_{j,i}^2}}$$

using the optimizer update rules. Combining and evaluating the summations gives

$$\sum_{t=1}^{T} \frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{n}_{t,i}}} \leq \sum_{t=1}^{T-1} \frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{n}_{t,i}}} + \frac{\sqrt{1-\nu^T}}{(1-\mu^T)^2} \frac{(1-\mu)^2}{\sqrt{T(1-\nu)}} \sum_{k=1}^{T} T \left(\frac{\mu^2}{\sqrt{\nu}}\right)^{T-k} \|g_{k,i}\|_2$$

where a substitution can take place. In order to prevent this from getting ugly, some new notation is introduced just to simplify this section and the ending regret bound. Using the formal definition $\gamma \triangleq \frac{\mu^2}{\sqrt{\nu}}$ and substituting it into the last equation gives

$$\sum_{t=1}^{T} \frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{n}_{t,i}}} = \sum_{t=1}^{T-1} \frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{n}_{t,i}}} + \frac{T}{\sqrt{T(1-\nu)}} \sum_{k=1}^{T} \gamma^{T-k} \|g_{k,i}\|_2$$

after rearranging the variables. After,

$$\sum_{t=1}^{T} \frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{n}_{t,i}}} \leq \sum_{t=1}^{T} \frac{\|g_{t,i}\|_2}{t(1-\nu)} \sum_{j=0}^{T-t} t\gamma^t \leq \sum_{t=1}^{T} \frac{\|g_{t,i}\|_2}{t(1-\nu)} \sum_{j=0}^{T} t\gamma^t$$

is achieved after placing an upper bound on the left hand side of the inequality from the inductive step. Taking this a step further gives

$$\sum_{t=1}^{T} \frac{\|g_{t,i}\|_2}{t(1-\nu)} \sum_{j=0}^{T} t\gamma^t \leq \frac{1}{(1-\gamma)^2\sqrt{1-\nu}} \sum_{t=1}^{T} \frac{\|g_{t,i}\|_2}{t}$$

by upper bounding the previous inequality. This is the second big component that will be used in the main derivation of the regret bound because it is useful to know the convergence properties of the optimizer across every possible real iteration for the searching of $\min_{\theta \in \mathbf{R}} f(\theta)$ or $\max_{\theta \in \mathbf{R}} f(\theta)$ [9].

Regret is

$$R(T) = \sum_{t=1}^{T} f_t(\theta_t) - \inf_{\theta^* \in \chi} \sum_{t=1}^{T} f_t(\theta^*)$$

with reference to parameter $T$. So basically, extending the second definition for the convex surfaces gives

$$f_t(\theta_t) - f_t(\theta^*) \leq g_t^T(\theta_t - \theta^*) = \sum_{i=1}^{d} g_{t,i}(\theta_{t,i} - \theta_i^*)$$

for regret. So

$$R(T) \leq \sum_{t=1}^{T} \sum_{i=1}^{d} g_{t,i}(\theta_{t,i} - \theta_i^*)$$

is created when the upper bound is placed on the regret after summing on $t \in T$ [3, 7].

Beginning with the FASFA update rule of

$$\theta_{t+1} = \theta_t - \alpha_t \frac{\hat{m}\xi_{decay}}{\sqrt{\hat{n}}} = \theta_t - \alpha_t \xi_{decay} \frac{\mu_{1,t} m_{t-1} - (1-\mu_{1,t}) g_t}{\sqrt{\hat{n}_t}}$$

it can be applied into the context of regret by focusing on dimension $i$. Subtracting $\theta_i^*$ and squaring the sides gives

$$\begin{aligned}(\theta_{t+1,i} - \theta_i^*)^2 = (\theta_{t,i} - \theta_i^*)^2 \\ &- \frac{2\alpha\xi_{decay}}{1-\mu^t} \left( \frac{\mu_t m_{t-1,i}}{\sqrt{\hat{n}_{t,i}}} - \frac{-(1-\mu_t) g_{t,i}}{\sqrt{\hat{n}_{t,i}}} \right) (\theta_{t,i} - \theta_i^*) \\ &+ (\alpha_t \xi_{decay})^2 \left( \frac{\hat{m}_{t,i}}{\sqrt{\hat{n}_{t,i}}} \right)^2\end{aligned}$$

so that rearranging the terms gives

$$\begin{aligned}\frac{2\alpha_t \xi_{decay}(1-\mu_t)}{(1-\mu^t)\sqrt{\hat{n}_{t,i}}} g_t(\theta_{t,i}-\theta_i^*) = (\theta_{t,i}-\theta_i^*)^2 - (\theta_{t+1,i}-\theta_i^*)^2 \\ - \frac{2\alpha_t \xi_{decay}\mu_t}{(1-\mu^t)\sqrt{\hat{n}_{t,i}}} m_{t-1,i}(\theta_{t,i}-\theta_i^*) \\ + (\alpha_t \xi_{decay})^2 \left(\frac{\hat{m}_{t,i}}{\sqrt{\hat{n}_{t,i}}}\right)^2\end{aligned}$$

for the next step. For upper bounding with an arithmetic geometric series, it helps to simplify one side of the equation. Here it is pretty evident that the left side would be much easier to simplify than the right one. It can be cleared into just $g_t(\theta_{t,i}-\theta_i^*)$ by taking $\frac{(1-\mu^t)\sqrt{\hat{n}_{t,i}}}{2\alpha_t\xi_{decay}(1-\mu_t)}$ and multiplying it onto every term into the equation. Doing this gives

$$\begin{aligned}g_{t,i}(\theta_{t,i}-\theta_i^*) = \frac{(1-\mu^t)\sqrt{\hat{n}_{t,i}}}{2\alpha_t \xi_{decay}(1-\mu_t)}((\theta_{t,i}-\theta_i^*)^2 - (\theta_{t+1,i}-\theta_i^*)^2) \\ - \frac{\mu_t}{(1-\mu_t)} m_{t-1}(\theta_{t,i}-\theta_i^*) \\ + \alpha_t \frac{\xi_{decay}(1-\mu^2)}{2(1-\mu_t)}(\hat{m}_{t,i})^2\end{aligned}$$

which leaves only one more major adjustment left before upper bounding for the regret.

On the right side of this equation, the $(\theta_{t,i}-\theta_i^*)$ is going to be problematic. Algebraically, it is difficult to deal with here. This could be fixed by splitting the term up into ones that are more manageable using Young's inequality of $ab \le a^p/p + b^q/q$ as a pseudo substitution. Let

$$a = \frac{\sqrt[4]{\hat{n}_{t-1,i}}}{\sqrt{a_{t-1}}}(\theta_i^* - \theta_{t,i})$$

and

$$b = \frac{\sqrt{a_{t-1}}}{\sqrt[4]{\hat{n}_{t-1,i}}} m_{t-1,i}$$

exist. This implies inequality

$$\frac{\mu_t}{(1-\mu_t)} m_{t-1,i}(\theta_i^* - \theta_{t,i}) \le \frac{\mu_t}{(1-\mu_t)}\left(\frac{\sqrt{\hat{n}_{t-1,i}}}{2a_{t-1}}(\theta_i^* - \theta_{t,i})^2 + \frac{a_{t-1}}{2\sqrt{\hat{n}_{t-1,i}}} m_{t-1,i}^2\right)$$

holds. So,

$$\begin{aligned} g_{t,i}(\theta_{t,i}-\theta_i^*) &= \frac{(1-\mu^t)\sqrt{\hat{n}_{t,i}}}{2\alpha_t\xi_{decay}(1-\mu_t)}((\theta_{t,i}-\theta_i^*)^2-(\theta_{t+1,i}-\theta_i^*)^2) \\ &+ \frac{\mu_t}{(1-\mu_t)}\left(\frac{\sqrt{\hat{n}_{t-1,i}}}{2a_{t-1}}(\theta_i^*-\theta_{t,i})^2+\frac{a_{t-1}}{2\sqrt{\hat{n}_{t-1,i}}}m_{t-1,i}^2\right) \\ &+ \alpha_t\frac{\xi_{decay}(1-\mu^2)}{2(1-\mu_t)}(\hat{m}_{t,i})^2 \end{aligned}$$

is given by implementing the identity.

Next, the inequality from the expansion from earlier is applied to to $g_{t,i}(\theta_{t,i}-\theta_i^*)$ to get part of the regret bound by summations on $i \in d$ for the $f_t(\theta_t)-f_t(\theta^*)$ and subsequently on $t \in T$ for the points on convex functions. Both summations started at $i=1$, and $t=1$. So in turn,

$$\begin{aligned} R(T) \leq & \sum_{d}^{i=1}\frac{(\theta_{1,i}-\theta_i^*)^2\sqrt{\hat{n}_{1,i}}}{2\alpha_1\xi_{decay}(1-\mu)}+\sum_{i=1}^{d}\sum_{t=1}^{T}\frac{(\theta_{t,i}-\theta_i^*)^2}{2\xi_{decay}(1-\mu)}\left(\frac{\sqrt{\hat{v}_{t,i}}}{a_t}-\frac{\sqrt{\hat{v}_{t-1,i}}}{a_{t-1}}\right) \\ &+\frac{\mu\alpha G_\infty}{(1-\mu)\sqrt{1-\nu}(1-\gamma)^2}\sum_{i=1}^{d}\|g_{1:T,i}\|_2+\frac{\alpha G_\infty}{(1-\mu)\sqrt{1-\nu}(1-\gamma)^2}\sum_{i=1}^{d}\|g_{1:T,i}\|_2 \\ &+\sum_{i=1}^{d}\sum_{t=1}^{T}\frac{\mu_t(\theta_i^*-\theta_{t,i})^2\sqrt{\hat{n}_{t,i}}}{2\alpha_t\xi_{decay}(1-\mu_t)} \end{aligned}$$

becomes

$$\begin{aligned} R(T) \leq \frac{D^2}{2\alpha\xi_{decay}(1-\mu)}\sum_{i=1}^{d}\sqrt{T\hat{v}_{T,i}}+\frac{(1+\mu)\alpha G_\infty}{(1-\mu)\sqrt{1-\nu}(1-\gamma)^2}\sum_{i=1}^{d}\|g_{1:T,i}\|_2 \\ +\frac{D_\infty^2 G_\infty\sqrt{1-\nu}}{2\alpha\xi_{decay}}\sum_{i=1}^{d}\sum_{t=1}^{T}\frac{\mu_t\sqrt{t}}{(1-\mu_t)} \end{aligned}$$

after factoring the third and fourth terms and considering that $\|\theta_t-\theta^*\|_2 \leq D$ paired with $\|\theta_m-\theta_n\|_\infty \leq D_\infty$. Also, $\alpha_t=\alpha/\sqrt{t}$ was substituted into the inequality [9].

Using an upper bound on the last term in the inequality by

$$\sum_{t=1}^{T}\frac{\mu_t\sqrt{t}}{(1-\mu_t)} \leq \sum_{t=1}^{T}\frac{\lambda^{t-1}\sqrt{t}}{(1-\mu)} \leq \frac{1}{(1-\mu)(1-\lambda)^2}$$

in which

$$\begin{aligned} R(T) \leq \frac{D^2}{2\alpha\xi_{decay}(1-\mu)}\sum_{i=1}^{d}\sqrt{T\hat{v}_{T,i}}+\frac{(1+\mu)\alpha G_\infty}{(1-\mu)\sqrt{1-\nu}(1-\gamma)^2}\sum_{i=1}^{d}\|g_{1:T,i}\|_2 \\ +\sum_{i=1}^{d}\frac{D_\infty^2 G_\infty\sqrt{1-\nu}}{2\alpha\xi_{decay}\mu(1-\lambda)^2} \end{aligned}$$

becomes the regret bound. But this is assuming the user does not enter in negative decay rates for $\mu, \nu$, in which this regret bound becomes invalidated apart from the control term $\xi_{decay}$ in the settings because of its outer absolute value function [9].