Variance Reduction Techniques: SARAH and ADAM

Alex Salce
Department of Mathematics,
The University of Arizona,
asalce@arizona.edu

SIE 596, Spring 2024

Abstract

We examine two stochastic optimization methods, ADAM and SARAH, that each utilize variance reduction techniques to improve theoretical convergence over similar methods. The variance reduction techiques have a bias tradeoff for each algortihm. ADAM combines RMSProp and momentum, which are biased methods, but implements a bias correction within the algorithm. SARAH is a modification of SVRG, but has biased inner loops unlike the unbiased inner loops of SVRG. Numerical experiments demonstrate variance reduction in both methods, and the effects of SARAH's inner loop bias can be seen in an application with a real dataset.

1 Introduction

Stochastic optimizers are a class of algorithms that have become prevalent in recent years in machine learning applications to process increasingly large datasets. First-order gradient descent algorithms utilizing stochastic gradients (individual gradient selected uniformly at random), or minibatch gradients (subsets of a full gradient whose elements selected uniformly at random), have consequently become core staples in marchine learning frameworks due to their advantageous computational efficiency for applications to datasets of both large sample size and dimensionality. Generally, these approaches optimize model parameters $x \in \mathbb{R}^d$ by solving the finite-sum problem.

$$min_{x \in \mathbb{R}^d} \left\{ P(x) \ = \ \frac{1}{b} \sum_{i=1}^{b} f_i(x) \right\}, \ I_k \subset 1, ..., n, \ |I_k| = b \tag{1}$$

1: $b = 1$ stochastic, $n > b > 1$ minibatch

A drawback of a stochastic or minibatch approach is the inherent variance in gradient updates due to the random gradient selection in their computations, and subsequent parameter updates over each timestep. For algorithms utilizing these approaches, controlling variance can be critical to guarantee convergence, which has motivated variance reduction techniques in the development of new stochastic optimization algorithms.

The subjects of this paper, ADAM and SARAH, are stochastic optimizers that utilize stochastic/minibatch gradients with variance reduction techniques. Although the two algorithms are fundamentally different in their approaches, they both utilize recursive accumulated gradient information for variance reduction.

1.1 Applications

The ADAM optimizer is one of the most popular optimizers in a variety of ML frameworks including deep learning, neural networks, natural language processing, and generative adversarial networks, to name a few. These applications benefit from ADAM's strengths against other optimizers; it is fast, requires little hyperparameter tuning, performs well with sparse gradients, and can be applied to stochastic objective functions. Further, it has proven convergence for convex objective functions, as well as demonstrated convergence for nonconvex objectives (He et al., 2023) [5]. The success of ADAM in practice has motivated variants like AdaMax, which has also become popular in sparse dataset applications. ADAM, AdaMax, and other ADAM methods are built-in to the TensorFlow Keras package [1]

SARAH is a direct modification of SVRG, however does not seem to be used as widely in practice as SVRG. While speculation as to why is not worthwhile, it is noteworthy that SARAH's variance reduction technique utilizes a biased methodology, where SVRG is unbiased. Still, there is a fair amount of research available for futher modifications of SARAH. For applications, SARAH can be a direct substitute for SVRG with only some updated parameter tuning considerations, so it can be utilized for a very large variety of applications as a SVRG replacement.

1.2 Literature review

ADAM, "ADAptive Moment estimation", was initially proposed in the paper *Adam: A method for stochastic optimization* (Kingma and Ba, 2017) [6]. This paper was scrutinized for its proofs of convergence for ADAM and whether ADAM was really advantageous over stochastic gradient descent (SGD). Further, some examples of ADAM diverging under its stated assumptions have been identified [8]. In 2022, *Provable Adaptivity In ADAM* by (Wang et al., 2022) [9] proposed a "relaxed" smoothness condition that addressed some shortcomings of the original theoreticall anlaysis. Another paper, *An improvement of the convergence proof of the ADAM-Optimizer* (Bock et al., 2018) [3] corrected some errors in the original proof. Still, ADAM remains very popular despite some of the original paper's theoretical shortcomings.

SARAH, "StochAstic Recursive grAdient algoritHm", was initially proposed in the paper *Sarah: A novel method for machine learning problems using stochastic recursive gradient* by (Nguyen et al., 2017) [7]. A follow on paper including one of the original authors, *Random-reshuffled sarah does not need full gradient computations* (Beznosikov and M. Takáč, 2022) [2] demonstrates that, utilizing a random reshuffling technique, SARAH can be modified to eliminate full gradient computations entirely. *SARAH-M*, (Yang, 2024) [10], explores the incorporation of momentum with the SARAH algorithm, similar to ADAM. Examples like this, among others, indicate an interest in the algorithm despite not being a clearly preferred optimizer over SVRG or similar methods.

The subsequent coverage of each algorithm will draw primarily from their respective original publications. While we will not go into detail about variants of either algorithm, these are references suggested for further research.

2 Methodology/Algorithm description

2.1 ADAM (ADAptive Moment estimation), see Algorithm 1 in Appendix

Methodology in this section is drawn from Kingma and Ba's *ADAM: A Method for Stochastic Optimization* (Kingma and Ba, 2017) [6].

The ADAM algorithm combines the concepts of momentum and RMSProp ("Root Mean Squared Propogation") using minibatch or stochastic gradient information to iteratively udpate parameters. The key feature of ADAM is an *adaptive* learning rate that is modulated by recursive accumulated gradient information.

In basic terms, the steps that are similar to momentum utilize an exponential weighted average of previous gradient information to calculate the gradient term at each descent step from previous gradient averages, having the effect of reducing variance in steps and moving more accurately in the direction of the optimum. The steps similar to RMSProp have the effect of adaptively scaling the learning rate based on an exponential weighted average of the magnitude of recent gradients. Both of these terms are biased toward the initialized weights, and ADAM employs bias-correction udpates to both the momentum and RMSProp terms.

2.1.1 Bias

The bias-correction differentiates ADAM from RMSProp and can be attributed to it's advantageous performance with sparse data. The bias term for the second moment estimate v_t by selecting $v_0 = \mathbf{0}$ can be derived by taking the expectation of the term at time t (omitting calculation steps).

$$\mathbb{E}[v_t] = \mathbb{E}[g_t^2] \cdot (1 - \beta_2^t) + \zeta$$

Where ζ is assumed to be small since β_2 should be selected to assign very small weights to gradients far enough in the past. The proof given in the paper also applies to the first moment estimate. Hence, the first and second moment bias correction terms are incorporated by dividing the respective estimates by the $(1 - \beta_{1,2}^t)$ term in Algorithm 1, and giving $\mathbb{E}[\hat{m}_t] = \mathbb{E}[g_t]$ and $\mathbb{E}[\hat{v}_t] = \mathbb{E}[g_t^2]$.

2.1.2 Convergence

The convergence proof given by Kingma and Ba imposes the following requirements for f. First, f is convex, defined by the following.

Definition 1. *A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if for all $x, y \in \mathbb{R}^d$, for all $\lambda \in (0, 1)$*

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

Additionally, we note that the following property holds for convex functions.

Lemma 1. *If a function $f : \mathbb{R}^d \to \mathbb{R}$ is convex, then for all $x, y \in \mathbb{R}^d$*

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

Omitting details of the proof, Kingma and Ba use Lemma 1 to establish an upper bound on the regret function $R(T) = \sum_{t=1}^{T} f_t(x_t) - f_t(x^*)$, where x^* is the optimal parameters for minimizing f.

Theorem 1. *Assume that the function f_t has bounded gradients, $||\nabla f_t(x)||_2 \leq G$, $||\nabla f_t(x)||_\infty \leq G_\infty$ for all $x \in \mathbb{R}^d$ and distance between any x_t generated by ADAM is bounded, $||x_n - x_m||_2 \leq D$, $||x_n - x_m||_\infty \leq D_\infty$ for any $m, n \in \{1, ..., T\}$, and $\beta_1, \beta_2 \in [0, 1)$ satisfy $\frac{\beta_1^2}{\sqrt{\beta_2}} < 1$. Let $\alpha_t = \frac{\alpha}{\sqrt{t}}$ and $\beta_{1,t} = \beta_1 \lambda^{t-1}$, $\lambda \in (0, 1)$. ADAM achieves the following guarantee, for all $T \geq 1$.*

$$R(T) \leq \frac{D^2}{2\eta(1-\beta_1)} \sum_{i=1}^{d} \sqrt{T\hat{v}_{T,i}} + \frac{\eta(1+\beta_1)G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\frac{\beta_1^2}{\sqrt{\beta_2}})^2} \sum_{i=1}^{d} ||g_{1:T,i}||_2 + \sum_{i=1}^{d} \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\eta(1-\beta_1)(1-\lambda)^2}$$

In simple terms, the proximity of $f_t(x_t$ to the optimum is bounded by tunable parameters and constant bounds based on the data. Under the assumptions of f_t from Theorem 1, the following convergence follows.

$$\frac{R(T)}{T} = \mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$$

$\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$ is the best theoretical convergence of regular SGD, however there has been some criticism of the theory regarding the advantages of ADAM over SGD despite observed advantages in emperical performance. In 2022, *Provable Adaptivity in ADAM* (Wang et al., 2022) [9] proposed the (L_0, L_1) smoothness assumption, which slightly relaxes the L-smooth condition for bounding gradients, and in conjunction with a growth condition bounding sum of square gradients, can prove the $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$ convergence of ADAM (details omitted here, see publication for proof).

2.2 SARAH (StochAstic Recursive grAdient algoritHm), see Algorithm 2 in Appendix

Methodology in this section is drawn from Nguyen, Liu, Scheinberg, and Takáč *SARAH: A Novel Method for Machine Learning Problems Using Stochastic Recursive Gradient* (Nguyen, Liu, Scheinberg, and Takáč, 2017) [7].

The SARAH algorithm is a stochastic optimizer that converges for convex L-smooth objectives f_i with some theoretical advantages over similar methods like SAG, SAGA, and SVRG. It is most similar to SVRG, using an identical outer loop step and a modified inner loop that uses recursive gradient information rather than using only the outer loop gradient at each inner loop iterate.

2.2.1 Bias

Reference Algorithm 2. SARAH and SVRG outer loops are the same, so we start by examining the inner loop update for SVRG.

$$v_t = \nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_0) + v_0$$

w_0 is computed in the outer loop step and is only updated after each full inner loop. The SARAH update replaces $\nabla f_{it}(w_0) + v_0$ with a recursive gradient update framework $\nabla f_{it}(w_{t-1}) + v_{t-1}$.

$$v_t = \nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_{t-1}) + v_{t-1}$$

As it holds in statistical theory, there is a bias-variance tradeoff with the SARAH update. The use of recursive gradient information in SARAH actually reduces the variance of the inner loop

steps to zero as inner loops iterates increase, whereas this is not the case for SVRG. However, while the inner loops of SVRG are unbaised, individual inner loop iterates for SARAH are biased. The expectation of the inner loop after t iterations is as follows (we will use P as in Equation 1).

$$\mathbb{E}[v_t|F_t] = \nabla P(w_t) - \nabla P(w_{t-1}) + v_{t-1} \neq \nabla P(w_t)$$

where $F_t = \sigma(w_0, i_1, i_2, ..., i_{t-1})$ is the sigma algebra generated by the sequence of updates from w_0. However, the total expectation holds $\mathbb{E}[v_t] = \mathbb{E}[\nabla P(w_t)]$, which distinguishes SARAH from SAG/SAGA.

2.2.2 Convergence

In general, we aim to bound the expected norm of the gradient for stochastic algorithms as follows.

$$\mathbb{E}[||\nabla P(w_\tau)||^2] \leq \epsilon \tag{2}$$

The general convergence result imposes the same convexity assumption for f_is as Definition 1. Additionally, we require f_is to be $L-$smooth (Lipschitz continuous gradient) as follows.

Definition 2. *For $f_i : \mathbb{R}^d \to \mathbb{R}$, $i \in [n]$, is L-smooth if and only if there exists a constant $L > 0$ such that*

$$||\nabla f_i(w) - \nabla f_i(w')|| \leq L||w - w'||, \forall w, w' \in \mathbb{R}^d$$

Under these assumptions, SARAH shows a sublinear convergence rate.

Corollary 1. *Suppose that each f_i conforms with Definitions 1 and 2. For Algorithm 2 within a single outer iteration with the learning rate $\eta = \sqrt{\frac{2}{L(m+1)}}$ where $m \geq 2L - 1$ is the total number of iterations, then $||\nabla P(w_t)||^2$ converges sublinearly in expectation with a rate of $\sqrt{\frac{2L}{m+1}}$, and therefore, the total complexity to achieve an ϵ-accurate solution defined by Equation 2 is $\mathcal{O}(n + 1/\epsilon^2)$.*

For the general convex case, it can be shown that with appropriate choice of learning and rate and inner loops based on the data, the total complexity to achieve an ϵ-accuracy solution for Equation 2 is $\mathcal{O}((n + (1/\epsilon)) \log 1/\epsilon)$.

A further assumption of strong convexity imposed on f_is can improve convergence of SARAH.

Definition 3. *The function $P : \mathbb{R}^d \to \mathbb{R}$, is μ-strongly convex if and only if there exists a constant $\mu > 0$ such that $\forall w, w' \in \mathbb{R}^d$,*

$$P(w) \geq P(w') + \nabla P(w')^T(w - w') + \frac{\mu}{2}||w - w'||^2$$

Assuming f_is are strongly convex in addition to regular convexity assumption and L-smooth, we have the following.

Theorem 2. *For f_is convex, L-smooth, and μ-strongly convex, choosing η and μ such that*

$$\sigma_m \stackrel{def}{=} \frac{1}{\mu\eta(m+1)} + \frac{\eta L}{2 - \eta L} < 1 \tag{3}$$

we have

$$\mathbb{E}[||\nabla P(\tilde{w}_s||^2|||] \leq (\sigma_m)^s ||\nabla P(\tilde{w}_0)||^2$$

This result implies that the variance bound σ_m^s on the expectation of the norm squared gradient after s outer loops is smaller than the equivalent variance bound after s loops α_m for SVRG.

$$\sigma_m \stackrel{def}{=} \frac{1}{\mu\eta(m+1)} + \frac{\eta L}{2-\eta L} < \frac{1}{\mu\eta(1-2L\eta)m} + \frac{1}{\frac{1}{2\eta L}-1} = \alpha_m$$

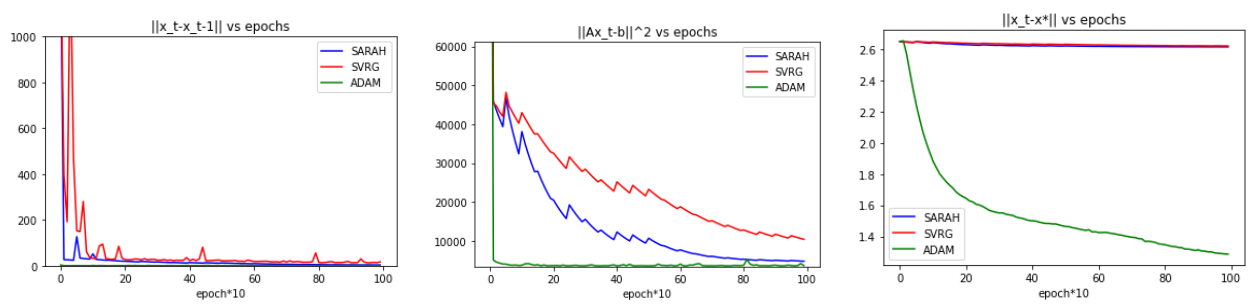
Both methods converge with rate $\mathcal{O}((n + L/\mu)\log(1/\epsilon))$, however SARAH's best theoretical convergence rate can use a higher learning rate than SVRG.

3 Numerical Experiments

The purpose of the numerical experiment will be to evaluate general performance of ADAM and SARAH on real data and to illustrate a bias-variance tradeoff between SVRG and SARAH. We use the "Wine Quality" [4] dataset from the UC Irvine Machine Learning Repository. The data are measurements of wine characteristics (features) used to predict wine quality scores. There are $n = 6497$ instances and $m = 11$ features. The experiements will use a simple least squares linear regression objective, assuming a system with data A and response b is of the form $Ax = b$.

$$\min_x ||Ax - b||^2$$

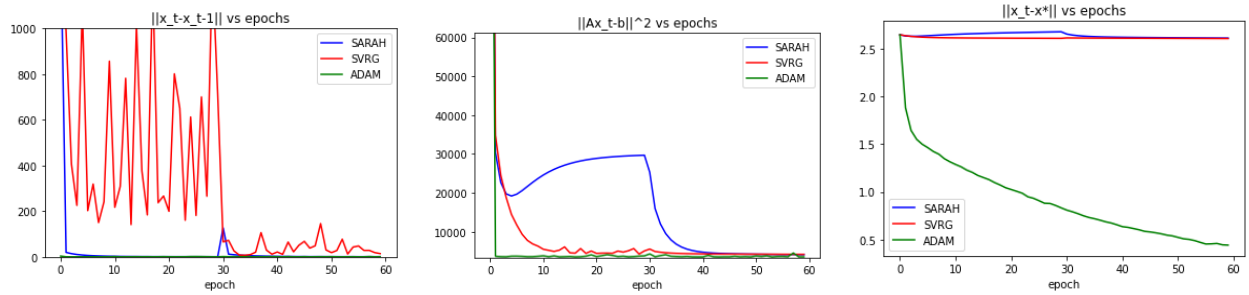
The least squares objective was chosen because it is strongly convex. x^* was computed separately using `scipy.optimize.lsq_linear` with tolerance set to $1e-10$.



ADAM		SARAH		SVRG	
Parameters	time	Parameters	time	Parameters	time
$\beta_1 = 0.9, \beta_2 = 0.999, \alpha = 0.001, batch = 10$	6.37s	$\eta = \frac{1}{2.1 \cdot L}, m = 3248, s = 20$	6.51s	$\eta = \frac{1}{5 \cdot L}, m = 3248, s = 20$	6.49s

Figure 1: Exp. A — 64970 total iterations

See Figures 1 and 2 for settings and computation time. The two examples chosen illustrate some high level takeaways. In terms of computational performance, both ADAM and SARAH are virtually identical to SVRG, though ADAM can become slightly slower for increased batch sizes. SARAH and SVRG are limited in flexibility for learning rate with this data since L is large and learning rate is $\mathcal{O}(\frac{1}{L})$. Both are descending, but are moving extremely slowly toward x^*, and the larger learning rate for SARAH hardly presents an advantage. We can see benefit of the adaptive



ADAM		SARAH		SVRG	
Parameters	time	Parameters	time	Parameters	time
$\beta_1 = 0.9, \beta_2 = 0.999, \alpha = 0.001, batch = 10$	38.6s	$\eta = \frac{1}{2.1 \cdot L}, m = 194910, s = 2$	38.8s	$\eta = \frac{1}{5 \cdot L}, m = 194910, s = 2$	38.5s

Figure 2: Exp. B — 389820 total iterations

learning rate of ADAM for both Exp A and B; it is moving toward it much faster and it continues to descend as iterations increase ($||x_t - x^*||$ vs. epochs).

Comparing SARAH and SVRG, we run the inner loop for half an epoch in Exp. A and for 30 epochs in Exp. B. The parameter choice in Exp. B adheres with Equation 3. Note that for this data, the iterations required for an ϵ accurate convergence with SARAH, for a reasonable choice of ϵ, was prohibitive. In both Exp. A and B, $||x_t - x_{t-1}||$ is clearly smoother for SARAH than SVRG, however we observe that the number of effective passes over the data in the inner loop can matter. In both the $||Ax_t - b||$ and $||x_t - x^*||$ plots, SARAH will stray away from the optimal solution due to bias after around 4 epochs. It is ultimately corrected by the second outer loop step, but we can see how the biased steps of the inner loop can cause SARAH drift, whereas SVRG continues to descend. As is the reality of statistics, we are trading variance for bias.

4 Conclusion and Future Direction

The findings in the numerical experiment highlight the advantages of the adaptive learning rate of ADAM, and some possible shortcomings of SARAH when compared against SVRG. ADAM is easy, straightforward, and doesn't require detailed parameter tuning. SARAH is more sensitive to its parameters, and where it has theoretical advantages in variance reduction, it has demonstrable tradeoffs with bias. Both methods clearly reduce the variance in steps for their implementations of stochastic descent, each in different ways. Future efforts should test these algorithms with different data of varying characteristics in size and sparsity, as well as with different objective functions. Additionally, variants of these algorithms like SARAH+, SARAH-M, and AdaMax could be compared against these methods, as well as others in the same class of stochastic first-order optimization methods, see literature review section and references for suggestions.

Appendix A Algorithms

Algorithm 1 The ADAM algorithm computes a batch stochastic gradient to compute momentum and RMSProp vectors at each timestep, with a bias correction step accounting for first and second moment estimates. Model parameters are updated using this recursive gradient information.

Require: parameters w, stochastic objective function $f_i(w)$
Require: learning rate η, exponential decay rates $\beta_1, \beta_2 \in [0, 1)$, tolerance ϵ
Initialize: initial parameter vector w_0, initial 1st moment vector $m_0 \leftarrow 0$, initial 2nd moment vector $v_0 \leftarrow 0$, initial timestep $t \leftarrow 0$

while w_t is not converged **do**

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_w f_t(w_{t-1})$ (*batch gradient at iteration t*)

$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (*[Monentum] udpate baised first moment estimate*)

$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (*[RMSProp] udpate baised second raw moment estimate*)

$\hat{m}_t \leftarrow m_t/(1 - \beta_1^t)$ (*[Monentum] bias-corrected first moment estimate*)

$\hat{v}_t \leftarrow v_t/(1 - \beta_2^t)$ (*[RMSProp] bias-corrected second raw moment estimate*)

$w_t \leftarrow w_{t-1} - \eta \cdot \hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)$ (*[Momentum + RMSProp] update parameters*)

end while

return w_t

Algorithm 2 The SARAH algorithm is identical to SVRG except for the *SARAH update*, which modifies the stochastic gradient estimate to use recursive gradient estimate information rather than the initialized gradient to update the gradient estimate in the inner loop.

Require: objective function $f(w)$
Require: learning rate $\eta > 0$, inner loop size m, outer loop size s
Initialize: initial parameter vector $\tilde{w}_0$
Iterate:

for $s = 1, 2, ...$ **do**

$w_0 = \tilde{w}_{s-1}$

$v_0 = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w_0)$ (*outer loop full gradient computation*)

$w_1 = w_0 - \eta v_0$ (*outer loop parameter update*)

for $t = 1, ..., m - 1$ **do**

Sample i_t uniformly at random from $[n]$

$v_t = \nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_{t-1}) + v_{t-1}$ (*SARAH update*)

$w_{t+1} = w_t - \eta v_t$ (*inner loop parameter update*)

end for

Set $\tilde{w}_s = w_t$ with t chosen uniformly at random from $\{0, 1, ..., m\}$

end for

References

[1] *TensorFlow Keras Optimizers Method tf.keras.optimizers.* `https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam`. Accessed: 2024-04-20.

[2] A. Beznosikov and M. Takáč, *Random-reshuffled sarah does not need full gradient computations*, Optimization Letters, 18 (2023), p. 727–749.

[3] S. Bock, J. Goppold, and M. Weiss, *An improvement of the convergence proof of the adam-optimizer*, 04 2018.

[4] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, *Wine Quality.* UCI Machine Learning Repository, 2009. DOI: https://doi.org/10.24432/C56S3T.

[5] M. He, Y. Liang, J. Liu, and D. Xu, *Convergence of adam for non-convex objectives: Relaxed hyperparameters and non-ergodic case*, 2023.

[6] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017.

[7] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, *Sarah: A novel method for machine learning problems using stochastic recursive gradient*, 2017.

[8] S. J. Reddi, S. Kale, and S. Kumar, *On the convergence of adam and beyond*, 2019.

[9] B. Wang, Y. Zhang, H. Zhang, Q. Meng, Z.-M. Ma, T.-Y. Liu, and W. Chen, *Provable adaptivity in adam*, 2022.

[10] Z. Yang, *Sarah-m: A fast stochastic recursive gradient descent algorithm via momentum*, Expert Systems with Applications, 238 (2024), p. 122295.