A Theoretical and Empirical Study on the Convergence of Adam with an *"Exact"* Constant Step Size in Non-Convex Settings

Anonymous authors
Paper under double-blind review

Abstract

In neural network training, RMSProp and Adam remain widely favoured optimisation algorithms. One of the keys to their performance lies in selecting the correct step size, which can significantly influence their effectiveness. Additionally, questions about their theoretical convergence properties continue to be a subject of interest. In this article, we theoretically analyse a constant step size version of Adam in the non-convex setting and discuss why it is important for the convergence of Adam to use a fixed step size. This article demonstrates the derivation and effective implementation of a constant step size for Adam, offering insights into its performance and efficiency in non-convex optimisation scenarios. **(i)** First, we show that the deterministic Adam can be affected by rapidly decaying learning rates (linear decay, exponential decay and *etc.*), mostly used to provide tight convergence bounds for Adam. This indicates that these rapidly decay rates play a dominant role in driving convergence. Next, (ii) Based on this observation, we derive a constant step that depends on the dynamics of the network as well as data to ensure that Adam can reach criticality for smooth and non-convex objectives, and we give bounds on the running time. Both deterministic and stochastic versions of Adam are analysed in this article. We show sufficient conditions for the derived constant step size to achieve asymptotic convergence of the gradients to zero with minimal assumptions. **(iii)** We design experiments to empirically study Adam's convergence with our proposed constant step size against state-of-the-art step size schedulers on classification tasks. Lastly, **(iii)** we also demonstrate that our derived constant step size has better abilities in reducing the gradient norms than the various state-of-the-art learning rate schedulers and a wide array of constant step sizes. Empirically, we also show that despite accumulating a few past gradients, the key driver for convergence in Adam is the non-increasing step sizes.

1 Introduction

Optimisation problems in machine learning are often structured as minimizing a finite sum $\min_{x\in\mathbb{R}^n} f(x)$, where $f : \mathbb{R}^n \mapsto \mathbb{R}$ and $f(x) = \frac{1}{k}\sum_{i=1}^{k} f_i(x)$. Each $f_i : \mathbb{R}^n \mapsto \mathbb{R}$ typically exhibits non-convex behavior, particularly in neural network domains. A prominent method for tackling such problems is *Stochastic Gradient Descent* (SGD), where updates occur iteratively as $x_{t+1} := x_t - \alpha\nabla\tilde{f}_{i_t}(x_t)$, with α denoting the step size and $\tilde{f}_{i_t}$ being a randomly chosen function from $\{f_1, f_2, ..., f_k\}$ at each iteration t. SGD is favoured for training deep neural networks for its computational efficiency, especially with mini-batch training, even with large datasets Bottou (2012).

Adaptive variants of SGD, which incorporate past gradients through averaging, have gained traction due to their ability to track gradient scaling on an individual parameter basis, as highlighted by *Bottou et al.* Bottou (2009). These methods are favoured for their perceived ease of tuning compared to traditional SGD. Adaptive gradient methods typically update using a vector obtained by applying a linear transformation, often referred to as "diagonal pre-conditioner" to the linear combination of all previously observed gradients.

This pre-conditioning is believed to enhance algorithm robustness to hyperparameter choices, making them less sensitive to initial settings.

Adagrad, introduced by *Duchi et al.* in *et al.* (2011); McMahan & Streeter (2010), showed superior performance, especially in scenarios featuring sparse or small gradients. However, its effectiveness diminishes in situations with non-convex loss functions and dense gradients due to rapid learning rate decay. To address this, variants like RMSProp *et al.* (2012), Adam Kingma & Ba (2015) (**Algorithm** 1), Adadelta Zeiler (2012), and Nadam Dozat (2016) have been proposed. These methods mitigate learning rate decay by employing exponential moving averages of squared past gradients, thereby limiting reliance on all past gradients during updates. While these algorithms have found success in various applications *et al.* (2019b), they have also been observed to exhibit non-convergence in many settings Sashank et al. (2018), especially in deterministic environments where noise levels are controlled during optimisation. This regulation is accomplished either by employing larger batches Martens & Grosse (2015); De et al. (2017); *et al.* (2015) or by integrating variance-reducing techniques *et al.* (2013; 2014).

Despite their widespread adoption in solving neural network problems, adaptive gradient methods like RMSProp and Adam often lack theoretical justifications in non-convex scenarios, even when dealing with exact or deterministic gradients *et al.* (2018). Several sufficient conditions have been proposed to guarantee the global convergence of Adam, which can be further classified into the following two categories:

(B1) Learning rate decay: *Reddi et al.* Sashank et al. (2018) demonstrated that the main cause of divergences in Adam and RMSProp predominantly arises from the disparity between the two successive learning rates.

$$\Gamma_t = \frac{\sqrt{\mathbf{v}_t}}{\alpha_t} - \frac{\sqrt{\mathbf{v}_{t-1}}}{\alpha_{t-1}} \tag{1}$$

If the positive definiteness property of Γ_t is violated, Adam and RMSProp may experience divergence. Barakat and Bianchi in Barakat & Bianchi (2020) relaxed the $\Gamma_t > 0$ constraint and demonstrated that Adam can converge when $\frac{\sqrt{\mathbf{v}_t}}{\alpha_t} \geq \frac{c\sqrt{\mathbf{v}_{t-1}}}{\alpha_{t-1}}$ holds for all t and some positive real c. Prior research efforts Shi et al. (2020); Tian & Parikh (2022); Chen et al. (2018); Luo et al. (2019); Défossez et al. (2020); Zou et al. (2019) have focused on establishing convergence properties of optimisation algorithms such as RMSProp and Adam. These investigations typically utilize step size schedules of the form $\alpha_t = \frac{\alpha}{t^a}$ for all $t \in \{1, 2, \ldots, T\}$, $\alpha \in (0, 1)$, and $a > 0$, or other time-dependent variations like $\alpha_t = \alpha(1-\beta_1)\sqrt{\frac{1-\beta_2^t}{1-\beta_2}}$. While a non-increasing (diminishing) step size is crucial for convergence, empirical analysis suggests that rapid decay of the step size can lead to sub-optimal outcomes, as discussed in the next section.

(B2) Temporal decorrelation: *Zhou et al.* in Zhou et al. (2018) emphasized that the divergence observed in RMSProp stems from imbalanced learning rates rather than the absence of $\Gamma_t > 0$, as previously suggested by *Reddi et al.* Sashank et al. (2018). Leveraging this insight, *Zhou et al.* Zhou et al. (2018) proposed AdaShift, which incorporates a temporal decorrelation technique to address the inappropriate correlation between $\mathbf{v}_t$ and the current second-order moment $\mathbf{g}_t^2$. This approach requires the adaptive learning rate α_t to be independent of $\mathbf{g}_t^2$. However, it's worth noting that the convergence analysis of AdaShift was primarily limited to RMSProp for resolving the convex counterexample presented by *Reddi et al.* Sashank et al. (2018).

In contrast to the aforementioned modifications and restrictions, we introduce an alternative sufficient condition (abbreviated as **SC**) to ensure the global convergence of the original Adam. The proposed **SC** (refer to **Section** 4.1) depends on a constant learning rate α [our proposed learning rate] and on the parameter β_1. Our **SC** doesn't necessitate rapidly decaying step sizes and positive definiteness of Γ_t. It's easier to verify and more practical compared to (**B2**).

NOTE: A recent study by Chen et al. (2022) proposes a similar convergence rate to ours using a constant step size approach for mini-batch Adam, where they employ $\alpha_t = \frac{\alpha}{\sqrt{T}}$, with α being any positive real number. However, our work asks the questions:

1. For better convergence[1] of Adam with non-decreasing or constant step sizes, is $\alpha_t = \frac{\alpha}{\sqrt{T}}$ sufficient?
2. Are there any other terms related to the dynamics of the model and data that need to be integrated as constants in the learning rates proposed above?
3. If such terms exist, will they lead to better convergence?

Our work specifies the *exact* learning rate that ensures convergence with fewer assumptions and contains information about the model dynamics and data distribution in the form of *Lipschitz smoothness of the loss.*

This article is organized as follows: Section 2 presents basic definitions and pseudocodes that will be used throughout our theoretical analysis. In **Section** 3, we provide the motivation for this work and outline the major contributions of the article. **Section** 4 discusses the convergence guarantees for Adam with a fixed step size. In **Section** 5, we present the details of the designed experiments and analyse the results. Finally, **Section** 6 concludes the article.

2 Notations, Definitions and Pseudocodes

In the following, we denote as $\mathbf{x}^T\mathbf{y}$ and $\|\mathbf{x}\|_2$ the scaler product and L_2-norm of the Hilbert space $\mathbb{R}^n$ respectively. For any differentiable and real-valued function $f : \mathbb{R}^n \mapsto \mathbb{R}$ and any point $\mathbf{x} \in \mathbb{R}^n$, we will denote as $\nabla_{\mathbf{x}} f \in \mathbb{R}^n$ the gradient of f at $\mathbf{x}$. The double derivative of f at $\mathbf{x}$ is denoted by $\nabla^2_{\mathbf{x}} f \in \mathbb{R}^{n\times n}$ which is a square matrix, also called the *Hessian matrix*. Next, we define the Lipschitz property, a standard assumption in optimisation literature, which we assume in all our proofs for non-convex objectives. Additionally, this section presents the pseudocode for the Adam optimizer.

Definition 1. ***K - Smoothness.*** *If $f : \mathbb{R}^d \mapsto \mathbb{R}$ is at least once differentiable, then we call it K smooth for some constant $K > 0$, if for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, the following inequality holds:*

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \nabla_{\boldsymbol{x}} f(\mathbf{x})^T(\mathbf{x} - \mathbf{y}) + \frac{K}{2}||\mathbf{x} - \mathbf{y}||^2$$

If a function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is K-smooth, then K is essentially the Lipschitz constant of the gradient of f. Henceforth, we will refer to K as the Lipschitz smoothness of the function, which is the same as the Lipschitz constant of the gradient of the function. Alternatively, we can define K-smoothness as:

Definition 2. ***K - Smoothness.*** *A function $f : \mathbb{R}^d \mapsto \mathbb{R}$ is called Lipschitz smooth if there exists a constant $K > 0$ such that:*

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n, \|\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) - \nabla_{\boldsymbol{y}} f(\boldsymbol{y})\|_2 \leq K\|\boldsymbol{x} - \boldsymbol{y}\|_2$$

The smallest K for which the previous inequality is true is called the Lipschitz constant of ∇f or Lipschitz smoothness of f and will be denoted $K(f)$. Note, from here onwards K and $K(f)$ can be used interchangeably, denoting the function's Lipschitz smoothness.

Definition 3. ***Square root of Penrose inverse.*** *If $\boldsymbol{v} \in \mathbb{R}^d$ and $V = diag(\boldsymbol{v})$, then we define $V^{\frac{-1}{2}} := \sum_{j\in support(\boldsymbol{v})} \frac{1}{\sqrt{\boldsymbol{v}_j}} \boldsymbol{b}_j \boldsymbol{b}_j^T$, where $\{\boldsymbol{b}_j\}_{j=1}^d$ is the standard basis of $\mathbb{R}^d$.*

Definition 4. ***Sign function.*** *We define a function $sign : \mathbb{R}^d \mapsto \{-1, 1\}^d$ as follows:*

$$sign(\mathbf{z})_j = \begin{cases} 1 & \text{if } \mathbf{z}_j \geq 0 \\ -1 & \text{else} \end{cases}$$

[1]Better convergence refers to the gradient norm of the loss approaching zero more closely as iterations progress. If two algorithms, $\mathcal{A}_1$ and $\mathcal{A}_2$, drive the gradient norm of the loss to 0.1 and 0.001 respectively, $\mathcal{A}_2$ demonstrates superior convergence. To assess the convergence speed accurately, one must analyze the loss curve plot.

Algorithm 1: Adam Algorithm Pseudocode Kingma & Ba (2015)

Input: Learning rate: $\alpha \in (0, 1]$, $\beta_1, \beta_2 \in [0, 1)$, initial starting point $\mathbf{w}_0 \in \mathbb{R}^d$, a constant vector $\rho \mathbf{1}_d > \mathbf{0} \in \mathbb{R}^d$, we assume we have access to a noisy oracle for gradients of $f : \mathbb{R}^d \mapsto \mathbb{R}$

Initialization: $\mathbf{m}_0 = 0$, $\mathbf{v}_0 = 0$

for t ***from*** *1* ***to*** T*:* **do**

$\quad \mathbf{g}_t = \nabla f(\mathbf{w}_t)$

$\quad \mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1)\mathbf{g}_t$

$\quad \mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2)\mathbf{g}_t^2$

$\quad \mathbf{V}_t = \texttt{diag}(\mathbf{v}_t)$

$\quad \mathbf{w}_{t+1} = \mathbf{w}_t - \alpha(\mathbf{V}_t^{1/2} + \texttt{diag}\left(\rho \mathbf{1}_d\right))^{-1} \mathbf{m}_t$

End

3 Motivation & Contributions

In practical scenarios, such as learning latent variables from vast datasets with unknown distributions, the goal is to solve the optimisation problem:

$$\underset{\mathbf{w} \in \mathbb{R}^d}{min} \mathcal{L}(\mathbf{w}) := \mathbb{E}_{\zeta \sim \mathcal{P}}[l(\zeta, \mathbf{w})] \tag{2}$$

Here $\mathcal{L} : \mathbb{R}^n \mapsto \mathbb{R}$ is a non-convex loss function, ζ is a random variable with an unknown distribution $\mathcal{P}$ and $l(\zeta, \mathbf{w})$ quantifies the performance of parameter vector $\mathbf{w} \in \mathbb{R}^n$ on the random variable ζ. Common iterative optimisation algorithms like Adam and RMSProp are often employed to solve this problem. It's observed that using a more aggressive constant step size often leads to favourable outcomes Jing (2020); Li et al. (2021). However, determining the optimal learning rate requires an exhaustive search. Specifically, when optimizing for a parameter set $\mathbf{w} \in \mathbb{R}^d$, the goal is to find a parameter $\mathbf{w}^*$ aligning with a minimum in the loss landscape $\mathcal{L}(\mathbf{w})$. Gradient descent-based algorithms, including those with momentum, iteratively update parameters using $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha_t g(\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}_t))$, where α_t is the step size at iteration t and $g : \mathbb{R}^d \mapsto \mathbb{R}^d$ is a function incorporating the gradient and momentum terms. Convergence of the parameter sequence $\{\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_{T-1}\}$ to $\mathbf{w}^*$ necessitates the sequence $\{(\mathbf{w}_{t+1} - \mathbf{w}_t)\}$ to converge to $\mathbf{0}$ as $t \to T$. This corresponds to the sequence $\{\alpha_t g(\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}_t))\}$ converging to $\mathbf{0} \in \mathbb{R}^d$. The literature often opts for a non-increasing step size α_t, such as $\alpha_t \propto \frac{1}{t^a}$, where $a > 0$, to theoretically prove convergence. Despite non-zero gradient norms, α_t can cause $\{\alpha_t g(\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}_t))\}$ to approach $\mathbf{0}$, even if $\{\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}_t)\}$ does not. Although accumulating past gradients helps mitigate the decay of the learning rate, there's a risk that learning rates with rapid decay may dominate. Our empirical analysis in **Section** 5 demonstrates that Adam, with non-increasing learning rates, does not aggressively drive the gradient norm of the loss towards zero. With a fixed step size $\alpha_t = \alpha > 0$, if $\mathbf{w}_t \to \mathbf{w}^*$, then $\|\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}_t)\|_2 \to \mathbf{0}$. This ensures convergence to a saddle point of $\mathcal{L}(\mathbf{w})$ with a fixed step size, an assurance not available with a non-increasing and iteration-dependent step size.

Consequently, our research aims to identify the optimal constant step size, ensuring convergence in both deterministic and stochastic Adam iterations beyond a time threshold $t > T(\beta_1, \rho)$ where $T(\beta_1, \rho)$ is a natural number.

A summary of our contributions: The major contributions of this article are summarized as follows.

1. We derive an exact constant step size to guarantee convergence of deterministic as well as stochastic Adam. To the best of our knowledge, this study is the first to theoretically guarantee Adam's global convergence (gradient norm of loss function converges to 0) with an exact constant step size, which depends on the dynamics of the model and data.

2. Our study offers runtime bounds for deterministic and stochastic Adam to achieve approximate criticality with smooth non-convex functions.

3. We introduced a simple method to estimate the Lipschitz smoothness [2] of the loss function *w.r.t* the network parameters. We offer a probabilistic guarantee for the convergence of our estimated Lipschitz constant to its true value.
4. We empirically show that even with past gradient accumulation, deterministic Adam can be affected by rapidly decaying learning rates. This indicates that these rapid decay rates play a dominant role in driving convergence.
5. We also demonstrate empirically that with our analysed step size, Adam quickly converges to a favourable saddle point with high validation accuracy.

4 Convergence Guarantee For Adam With Fix Step Size

Previously, it has been shown in Sashank et al. (2018); De et al. (2018); Défossez et al. (2020) that deterministic RMSProp and Adam can converge under certain conditions with adaptive step size. Here, we give the first result about convergence to criticality for deterministic and stochastic Adam with constant step size, albeit under a certain technical assumption about the loss function (and hence on the noisy first-order oracle). *The proofs of the theorems discussed in this section, along with additional experiments, are provided in the* ***Appendix****, which is included as part of the* ***Supplementary Material*** *for this paper.*

4.1 Novel Sufficient Conditions (SC) for Convergence of Adam

Below are the commonly employed assumptions for analyzing the convergence of stochastic algorithms for non-convex problems:

1. The minimum value of the problem $\mathcal{L}^* = argmin_{\mathbf{w}\in\mathbb{R}^d}\mathcal{L}(\mathbf{w})$ is lower bounded, *i.e* $\mathcal{L}^* > -\infty$.
2. The stochastic gradient $\mathbf{g}_t$ of the loss function is an unbiased estimate, *i.e* $\mathbb{E}[\mathbf{g}_t] = \nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}_t)$.
3. The second-order moment of stochastic gradient $\mathbf{g}_t$ is uniformly upper-bounded, *i.e* $\mathbb{E}[\|\mathbf{g}_t\|_2^2] \leq (\mathbb{E}[\|\mathbf{g}_t\|_2])^2 \leq \gamma^2$.

In addition, we suppose that that the parameters β_1 and learning rate α satisfy the following restrictions:

1. The parameter β_1 satisfies $\beta_1 < \frac{\epsilon}{\epsilon+\gamma}$ for some $\epsilon > 0$.
2. Our proposed step size $\alpha_t = \alpha \in \mathbb{R}^+$ remains constant throughout the analysis.

Theorem 1. ***Deterministic Adam converges with proper choice of constant step size.*** *Let the loss function $\mathcal{L} : \mathbb{R}^n \mapsto \mathbb{R}$ be $K-$smooth and let $\gamma < \infty$ be an upper bound on the norm of the gradient of $\mathcal{L}$. Also assume that $\mathcal{L}$ has a well-defined minimiser $\boldsymbol{w}^*$ such that $\boldsymbol{w}^* = argmin_{\boldsymbol{w}\in\mathbb{R}^d}\ \mathcal{L}(\boldsymbol{w})$. Then the following holds for Algorithm (1):*

For any $\epsilon, \rho > 0$ if we let $\alpha = \sqrt{2(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}(\boldsymbol{w}^*))/K\delta^2 T}$, then there exists a natural number $T(\beta_1, \rho)$ (depends on β_1 and ρ) such that $\min_{t=1,\ldots,T}\|\nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w}_t)\|_2 \leq \epsilon$ for some $t \geq T(\beta_1, \rho)$, where $\delta^2 = \frac{\gamma^2}{\rho^2}$.

Theorem 2. ***Stochastic Adam converges with proper choice of constant step size.*** *Let the loss function $\mathcal{L} : \mathbb{R}^n \mapsto \mathbb{R}$ be $K-$smooth and be of the form $\mathcal{L} = \sum_{j=1}^m \mathcal{L}_j$ such that (a) each $\mathcal{L}_j$ is at-least once differentiable, (b) the gradients satisfy $sign(\mathcal{L}_r(\boldsymbol{w})) = sign(\mathcal{L}_s(\boldsymbol{w}))$ for all $r, s \in \{1, 2, \ldots, m\}$, (c) $\mathcal{L}$ has a well-defined minimiser $\boldsymbol{w}^*$ such that $\boldsymbol{w}^* = argmin_{\boldsymbol{w}\in\mathbb{R}^d}\ \mathcal{L}(\boldsymbol{w})$. Let the gradient oracle, upon invocation at $\boldsymbol{w}_t \in \mathbb{R}^d$, randomly selects j_t from the set $\{1, 2, \ldots, m\}$ uniformly, and then provides $\nabla f_{j_t}(x_t) = \boldsymbol{g}_t$ as the result.*

Then, for any $\epsilon, \rho > 0$ if we let $\alpha = \sqrt{2(\mathcal{L}(\boldsymbol{w}_0) - \mathcal{L}(\boldsymbol{w}^*))/K\delta^2 T}$, then there exists a natural number $T(\beta_1, \rho)$ (depends on β_1 and ρ) such that
$\min_{t=1,\ldots,T}\mathbb{E}[\|\nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w}_t)\|_2] \leq \epsilon$ for some $t \geq T(\beta_1, \rho)$, where $\delta^2 = \frac{\gamma^2}{\rho^2}$.

[2] Our derived step size, crucial for ensuring convergence, is determined by the Lipschitz smoothness of the loss function.

Remark 1: With our analysis, we showed that both deterministic and stochastic Adam with constant step size converges with rate $\mathcal{O}\left(\frac{1}{T^{1/4}}\right)$. This convergence rate is similar to the convergence rate of SGD proposed by *Li et al.* in Li et al. (2014). Our motivation behind these theorems was primarily to understand the conditions under which Adam can converge, especially considering the negative results presented in *Reddi et al.* Sashank et al. (2018). However, we acknowledge that it is still an open problem to tighten the analysis of deterministic as well as stochastic Adam and obtain faster convergence rates than we have shown in the theorem. One thing to keep in mind is that our convergence rate is obtained for a fixed step size, and having a decaying step size provides additional opportunities for analysis to tighten the bound Shi et al. (2020); Tian & Parikh (2022); Chen et al. (2018); Luo et al. (2019); Défossez et al. (2020); Zou et al. (2019).
Remark 2: Based on the preceding two theorems, it becomes evident that the Adam achieves convergence when equipped with an appropriately chosen constant step size $\alpha = \sqrt{2(\mathcal{L}(\mathbf{w}_0) - \mathcal{L}(\mathbf{w}^*))/K\delta^2 T}$, **according to our analysis**. Subsequently, we analyse the given step size and conduct extensive experiments to ascertain that, with the designated step size, the gradient norm effectively and more aggressively tends towards zero compared to other learning rate scheduling techniques (Listed in Table 1). Hence, our empirical investigations validate that the chosen step size ensures convergence in practice. We compare our step size with several state-of-the-art schedulers and an array of constant step sizes in **Section** 5.

4.2 Analysis of Constant Step Size

The optimal learning rate, as per **Theorem** 1 and **Theorem** 2, depends on factors like the Lipschitz smoothness of the loss[3] (K), initial and final loss values, the total number of iterations/epochs (T), and an additional term denoted as δ^2. For simplicity, we omitted the δ^2 term as it depends on an oracle of gradients and set the final loss term to zero. Thus, our approximate learning rate is now expressed as $\alpha \approx \sqrt{\frac{2\mathcal{L}(\mathbf{w}_0)}{\hat{K}T}}$, making it practical while still capturing the core concept of **Theorem** 1 and **Theorem** 2 . Here $\hat{K}$ is the approximated smoothness of the loss, defined in the next section.

4.3 Approximating The Lipschitz Smoothness of Loss

Determining the appropriate Lipschitz smoothness for a neural network is a key area of research in deep learning, with various methods proposed for estimation, as demonstrated in recent works Virmaux & Scaman (2018); Fazlyab et al. (2019); Latorre et al. (2020); *et al.* (2019a); Gouk et al. (2021). Estimating the learning rate prior to training requires careful estimation of the Lipschitz smoothness of the loss function. For locally Lipschitz smooth functions (i.e. functions whose restriction to some neighbourhood around any point is Lipschitz), the Lipschitz smoothness may be computed using its Hessian matrix.

Theorem 3. *(Rademacher Federer (2014) [22, Theorem 3.1.6]) If $f : \mathbb{R}^d \mapsto \mathbb{R}$ is a locally Lipschitz smooth function, then f is differentiable almost everywhere. Moreover, if f is at least twice differentiable and $\nabla_{\boldsymbol{x}} f$ is Lipschitz continuous, then*

$$K(f) = sup_{\boldsymbol{x}\in\mathbb{R}^d} \|\nabla^2_{\boldsymbol{x}} f\|_2 \tag{3}$$

where $\|\boldsymbol{A}\|_2$ is the spectral norm of the matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$.

For large networks, directly optimizing according to **Theorem** 3 can be time-consuming. Instead, we propose a novel technique to approximate the Lipschitz smoothness. It's important to note that this work does not aim to estimate the best Lipschitz smothness for the loss or network; rather, it focuses on the convergence analysis of Adam. We introduce Algorithm 2 to approximate the Lipschitz smoothness of the loss[4], supported by mathematical proof that our estimate *converges in distribution* to the original Lipschitz smoothness. We shall now prove that our estimated Lipschitz smoothness ($\hat{K}$) converges to the real Lipschitz smoothness (K) in distribution.

Theorem 4. *Given $\|.\|_2$ and $\nabla^2_{\boldsymbol{w}}\mathcal{L}$ are continuous functions and $\|\nabla^2_{\boldsymbol{w}}\mathcal{L}\|_2$ is bounded from above. If $\mathcal{L}$ is locally Lipschitz smooth then, $K(\mathcal{L}) = sup_{\boldsymbol{w}\in\mathbb{R}^d} \|\nabla^2_{\boldsymbol{w}}\mathcal{L}\|_2$. Let $\hat{K}$ be a random variable defined as :*

$$\hat{K} = \max_{1\leq j\leq N} \|\nabla^2_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w}_j)\|_2 \tag{4}$$

[3]The Lipschitz smoothness of the loss is with respect to network parameters.

[4]**Algorithm** 2 is given for the full-batch scenario. The algorithm for the mini-batch case is deferred to the **Appendix** due to space constraints.

where $\boldsymbol{w}_j$, $j \in \{1, 2, \ldots, N\}$ are i.i.d samples drawn from a full domain distribution $\mathcal{W}$. Then, as $N \to \infty$, the random variable $\hat{K}$ converges in distribution to $K(\mathcal{L})$.
Mathematically:

$$\lim_{N\to\infty} P(\hat{K} \leq k) = \begin{cases} 0 & \text{if } k < K(\mathcal{L}) \\ 1 & \text{if } k \geq K(\mathcal{L}) \end{cases} \tag{5}$$

Algorithm 2: Estimating Lipschitz Smoothness of Loss Function

Input: Dataset: $\mathcal{X} \sim \mathcal{P}$, Loss function: $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$, A network parameterized by $\mathbf{w} \in \mathbb{R}^d \sim \mathcal{W}$, No. of iterations: N
Output: $\hat{K}$
Initialization: $\hat{K} = 0$, Iteration Number $n = 0$
for n ***from 1 to*** N*:* **do**
  Randomly sample the network weights: $\mathbf{w}_n \sim \mathcal{W}$
  Compute loss for the current pass: $\mathcal{L}(\mathbf{w}_n) = \mathbb{E}_{\zeta\sim\mathcal{P}}[l(\zeta, \mathbf{w}_n)]$
  Compute the spectral norm of hessian at the current instant using power method by Müntz et al. (1913): $\|\nabla^2_{\mathbf{w}}\mathcal{L}(\mathbf{w}_n)\|_2$
  Estimate the Lipschitz smoothness ($\hat{K}$): $\hat{K} = max(\|\nabla^2_{\mathbf{w}}\mathcal{L}(\mathbf{w}_n)\|_2, \hat{K})$
End

From **Theorem** 4, $\hat{K} \xrightarrow{d} K(\mathcal{L})$[5]. optimisation of the loss gradient can be achieved through iterative techniques like steepest descent Boyd & Vandenberghe (2004), backtracking Boyd & Vandenberghe (2004), and Wolfe's and Goldstein's conditions Wolfe (1969), aiming to identify the direction (weights $\mathbf{w}$) maximizing the gradient norm. However, non-convex loss surfaces in deep learning may lead to convergence to sub-optimal local maxima. Also, losses with unbounded gradients can lead to excessively high Lipschitz constants. Our approach addresses this by leveraging the *convergence in distribution* phenomenon from probability theory. We randomly select directions from a distribution $\mathcal{W}$ and take the maximum gradient norm across all evaluated samples, theoretically converging to the true Lipschitz constant in distribution. Note, to respect **Theorem** 3, the distribution of the weights $\mathcal{W}$ needs to be full domain.

5 Experimental Setup

We assess Adam's performance with our selected step size through experiments on fully connected networks with ReLU activations. Additionally, we expand our analysis to include classification tasks on MNIST and CIFAR-10 datasets using LeNet LeCun et al. (1998) and VGG-9 Simonyan & Zisserman (2014), respectively. To maintain strict experimental control, we ensure that all network layers have identical widths denoted as h. For experiments, in **Section** 5.2 and 5.3, we used Kaiming uniform distribution He et al. (2015) for initialization. Default PyTorch parameters are chosen [Link]. No regularization is applied. Before delving into further details about the experiments, we will first describe the model architectures utilized for experiments in the subsequent sections.

5.1 Outline of the Experiments Conducted

We conducted classification experiments on the MNIST dataset for various network sizes, encompassing different values of network depth and h. CIFAR-10 experiments are carried out only on VGG-9 architecture. All experiments are implemented using PyTorch. We compare the performance of Adam using **our proposed step size** with:

1. A list of commonly used schedulers and several non-increasing step sizes that were utilized to demonstrate theoretical convergence for Adam in past literature. The schedulers include **(i)** Linear LR,

[5] **Theorem** 4 holds for mini-batch case too. Refer to **Appendix** for more insights.

(ii) Step LR, **(iii)** Square root Decay, **(iv)** Inverse Time Decay, **(v)** Exponential Decay, **(v)** One Cycle LR, and **(vi)** Cosine Decay Smith & Topin (2019). The full list with hyperparameters are given in **Table** 1, **Appendix** C.1.

2. An array of constant step sizes $\{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$ which are widely used in literature.

Each experiment was executed for 100 epochs. For various schedulers, we fine-tuned their hyperparameters to identify configurations where they exhibited the best performance and retained those values for the final comparison with our step size. These choices are made to provide a fair comparison with our constant step size approach, as overly rapid decay rates may not effectively minimise the gradient norm of the loss function. Below is the list of experiments we have conducted:

1. **Full-batch experiments (Section 5.2)**: To efficiently compare learning rate schedules, we created smaller versions of the MNIST and CIFAR-10 datasets: mini-MNIST and mini-CIFAR-10, respectively. Mini-MNIST comprises 5500 training images and 1000 test images, selected from the original MNIST dataset. Mini-CIFAR-10 includes 500 training images and 100 test images per class, representing about 10% of the full CIFAR-10 dataset. Despite their reduced size, both mini datasets adequately represent their respective originals for experimentation.

2. **Mini-batch experiments (Section 5.3)**: To see if our findings from the full-batch experiments apply to mini-batch scenarios according to **Theorem** 2, we conducted similar experiments using a mini-batch setup with a fixed batch size of 5,000 for CIFAR10 and MNIST both. We utilized the entire training sets of CIFAR10 and MNIST for these experiments. We tested our models on CIFAR10 using the VGG-9 architecture and on MNIST using the LeNet architecture. We also justified why such a large batch size is needed for empirical convergence.

3. **Effect of initialization on our learning rate**: As our learning rate $\alpha = \sqrt{\frac{2\mathcal{L}(\mathbf{w}_0)}{\hat{K}'T}}$ depends on the initial loss value and the estimated Lipschitz constant, we conducted experiments to demonstrate the independence of our learning rate with respect to different network initialization techniques. *Please refer to* **Appendix** C.4 *for the results.*

4. **Additional experiments on CNN architectures (Section 5.4)**: To examine the potential generalizability of our theoretical findings across architectures, we trained an image classifier on CIFAR-10 and MNIST datasets employing VGG-like networks and LeNet architectures, respectively.

5.2 Comparing Performances in Full-Batch Setting

In **Figure** 1 and 2, we illustrate the variations in gradient norm, training loss, and test loss across iterations for different schedulers. These experiments were conducted on single-layer, three-layer, and five-layer classifiers with 1000 nodes in each hidden layer, trained on mini-MNIST. Additional comparisons for various neural network architectures with different widths and depths are provided in the **Appendix**, where similar qualitative trends can be observed. **(1) Comparison against various learning rate schedulers**: We compared our step size against several well-known schedulers listed in **Table** 1, **Appendix** C.1. Except for cosine decay and one-cycle learning rate, all other schedulers are non-increasing. Our goal was to evaluate their efficacy in driving the gradient norm of the loss function towards zero. **(2) Comparison against various fixed learning rates**: We tested our learning rate against several other fixed learning rates, $\{10^{-2}, 10^{-3}, 10^{-3}, 10^{-4}, 10^{-5}\}$. We wanted to demonstrate that instead of searching extensively for the best learning rate to train a model, one could just use our learning rate and achieve decent results.

Looking at **Figures** 1 and 2, it is clear that our suggested learning rate helps decrease the gradient norm of loss more aggressively towards zero as compared to other schedulers and non-increasing learning rates, supporting our **Motivation**. Even with the accumulation of a few past gradients, deterministic Adam proves ineffective in mitigating the rapid decay of the learning rate. This is evident from **Figure** 1, where one can observe a notable difference in gradient norms after 100 epochs between the best-performing scheduler (Exponential in this case) and our learning rate is in order of 10^{-2}.

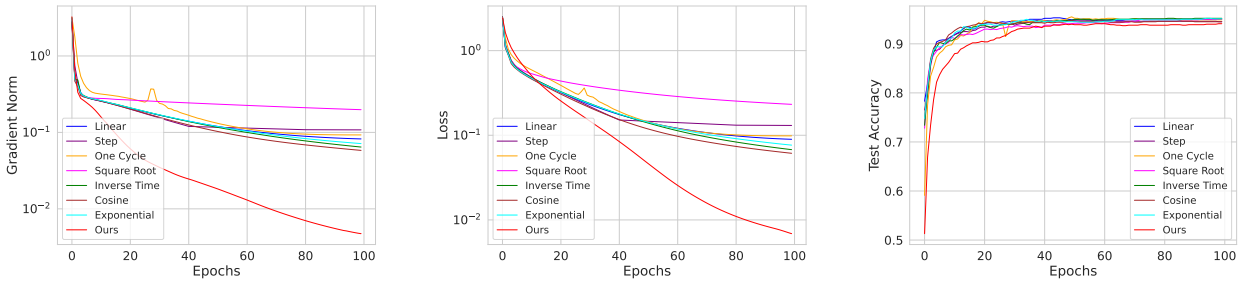

Figure 1: *Comparison with schedulers*: **Full-batch** experiments on a 3-layer network with 10^3 nodes/layer, trained on MNIST. *Left*: Grad. norm v/s epochs, *Middle*: Training loss v/s epochs, *Right*: Test acc. v/s epochs.

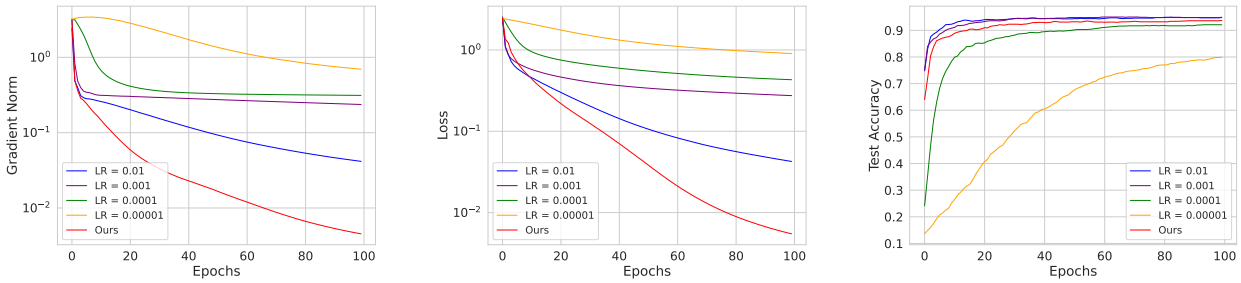

Figure 2: *Comparision with const. step sizes*: **Full-batch** experiments on a 3-layer network with 10^3 nodes/layer, trained on MNIST. *Left*: Grad. norm v/s epochs, *Middle*: Training loss v/s epochs, *Right*: Test acc. v/s epochs.

Additionally, these figures demonstrate that our learning rate not only reaches any stationary point but tends to approach a local minimum or its neighbourhood where the model achieves good validation accuracy. The plot of loss versus epoch indicates that our proposed learning rate leads to faster convergence in practice as compared to other schedulers. However, it's worth noting that achieving faster convergence in general is still a challenge. Nevertheless, using this learning rate results in quicker convergence than other schedulers and a range of fixed learning rates.

5.3 Comparing Performances in Mini-Batch Setting

In this section, we replicate the same set of experiments in mini-batch setting. We choose 5,000 as batch size for both the CIFAR10 and MNIST datasets. From **Figure** 3 and **Figure** 4, we observed that even in the mini-batch setting, our learning rate effectively reduces the gradient norm of loss more aggressively compared to other schedulers and constant step sizes. This trend aligns with the behaviour observed in the full-batch setting across all plots, including gradient norm, training loss, and validation accuracy.

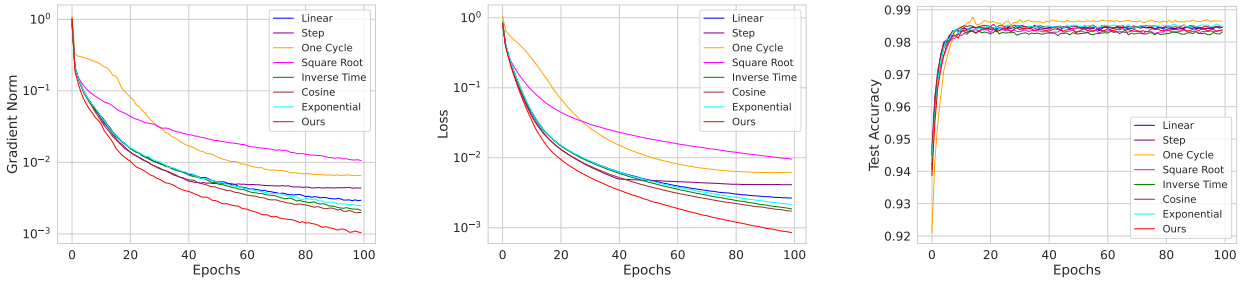

Figure 3: *Comparison with schedulers*: **Mini-batch** experiments on a 3-layer network with 10^3 nodes/layer, trained on MNIST. *Left*: Grad. norm v/s epochs, *Middle*: Training loss v/s epochs, *Right*: Test acc. v/s epochs.

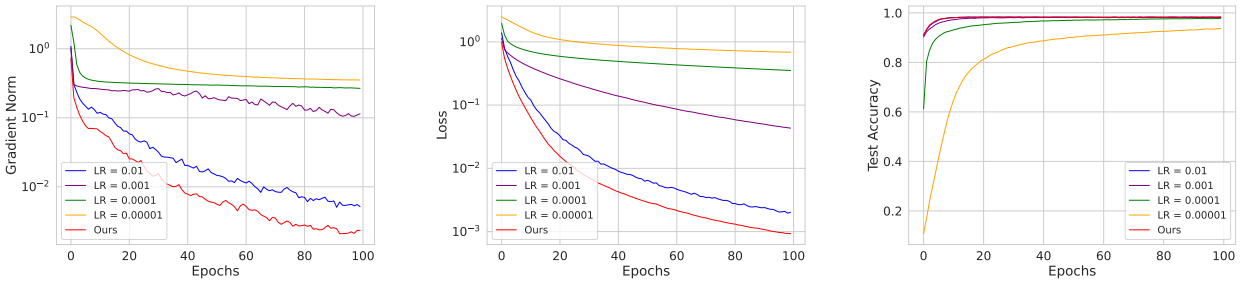

Figure 4: *Comparison with const. step sizes*: **Mini-batch** experiments on a 1-layer network with 10^3 nodes/layer, trained on MNIST. *Left*: Grad. norm v/s epochs, *Middle*: Training loss v/s epochs, *Right*: Test acc. v/s epochs.

Why such a large batch size is chosen? Despite the non-convergence issue demonstrated by *Reddi et al.* Sashank et al. (2018), it does not rule out convergence if the minibatch size increases over time, thereby reducing the variance of stochastic gradients. Increasing minibatch size has been shown to aid convergence in some optimisation algorithms not based on exponential moving average (EMA) methods *et al.* (2018); Hazan et al. (2015). In all our minibatch experiments (including those in the **Appendix**), we choose a

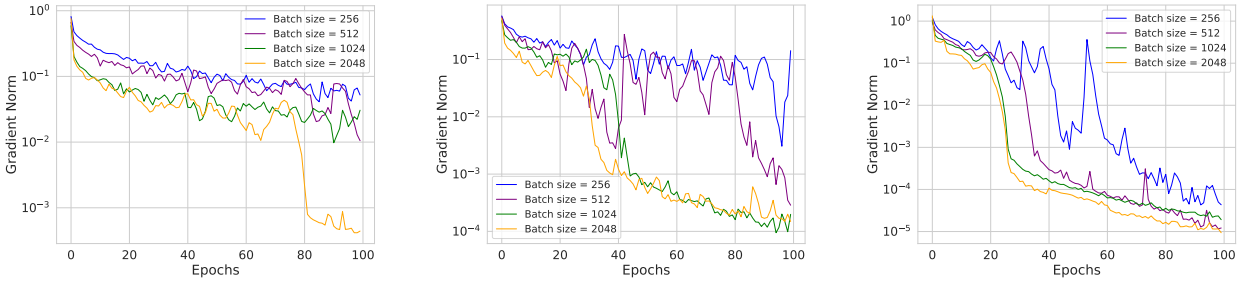

Figure 5: Increasing batch size experiments on *Left*: a fc network with 5 layers and 3×10^3 nodes/layer, trained on MNIST, *Middle*: LeNet trained on MNIST and *Right*: VGG-9 trained on CIFAR-10.

large batch size because our theoretical analysis for the minibatch setting, as described in **Theorem** 2, assumes that the loss is $K - Lipschitz$ and is represented by $\mathcal{L} = \sum_{j=1}^{m} \mathcal{L}_j$. Our proposed learning rate relies on the Lipschitz constant of the loss function. To accurately estimate this constant, it is crucial to estimate the stochastic gradients with low variance, as the Lipschitz constant is essentially the supremum of the true gradient norm over the network weights (**Theorem** 3). This approximation of gradients affects the estimation of the Lipschitz constant. Therefore, to make our step size effective in the minibatch setting, it is crucial to estimate the gradients with low variance, which in turn provides the correct estimation of the Lipschitz constant. From **Figure** 5, our empirical analysis suggests that increasing batch size will lead to faster convergence, as increasing batch size decreases variance. We now show results with our learning rate with increasing minibatch size in **Figure** 5.

5.4 LeNet on MNIST and VGG-9 on CIFAR-10

To test whether these results might qualitatively hold for convolutional neural networks (CNN) models, we trained an image classifier on CIFAR-10 and MNIST using VGG-9 and LeNet, respectively. We use minibatches of size 5,000 for the LeNet and 2,500 for VGG-9. As illustrated in **Figures** 6, 7, 8, 9, 10, and 11, our proposed learning rate scheduler performs well with CNNs, outperforming state-of-the-art schedulers and various constant step sizes. We can clearly see a gap of at least 10^{-1} order between our learning rate and the best-performing scheduler's gradient norm curve for LeNet experiments. This training was conducted for 100 epochs. Therefore, our empirical results demonstrate that our proposed learning rate can be effectively applied to CNNs and mini-batch scenarios, leading to better convergence.

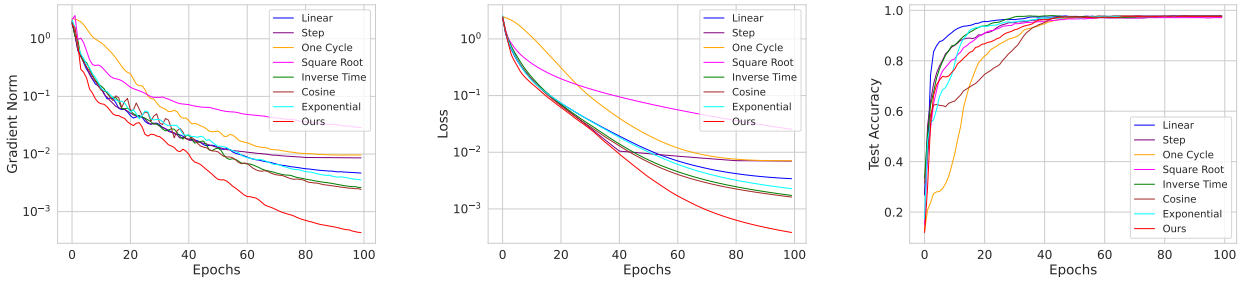

Figure 6: *Comparison with schedulers:* **Full-batch** experiments on LeNet architecture with MNIST data. *Left*: Grad. norm v/s epochs, *Middle*: Training loss v/s epochs, *Right*: Test acc. v/s epochs.

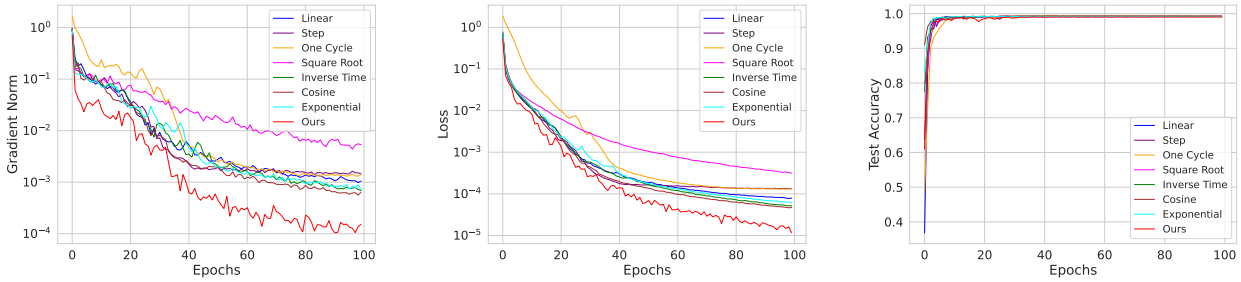

Figure 7: *Comparison with schedulers:* **Mini-batch** experiments on LeNet architecture with MNIST data. *Left*: Grad. norm v/s epochs, *Middle*: Training loss v/s epochs, *Right*: Test acc. v/s epochs.

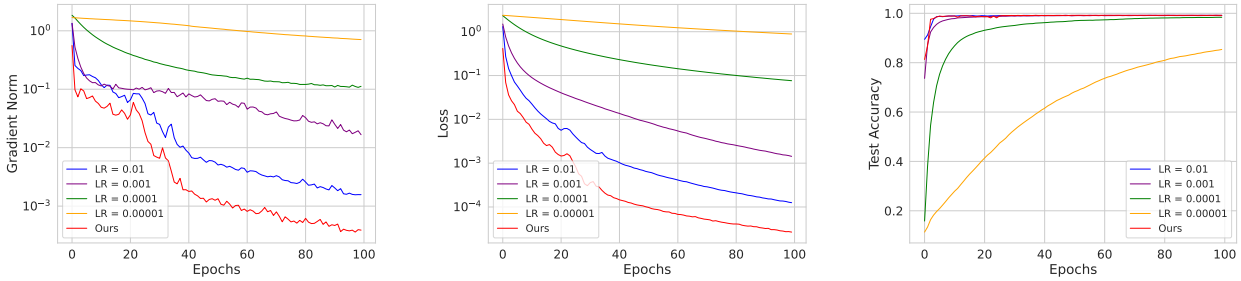

Figure 8: *Comparison with constant step sizes:* **Mini-batch** experiments on LeNet architecture with MNIST data. *Left*: Grad. norm v/s epochs, *Middle*: Training loss v/s epochs, *Right*: Test acc. v/s epochs.

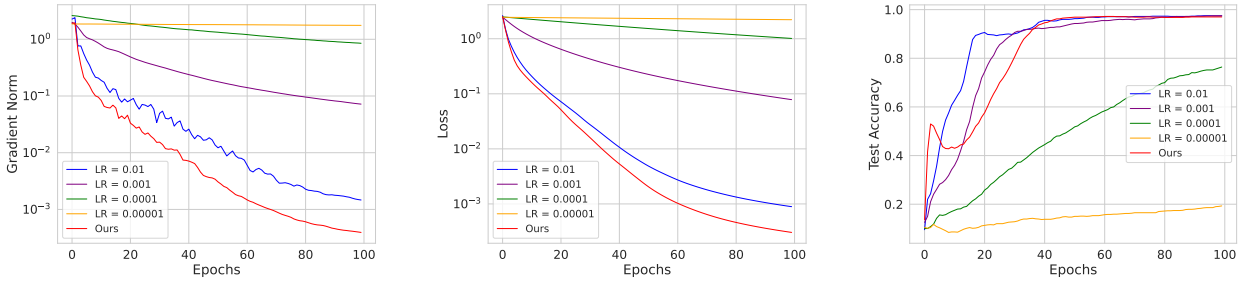

Figure 9: *Comparison with constant step sizes:* **Full-batch** experiments on LeNet architecture with MNIST data. *Left*: Grad. norm v/s epochs, *Middle*: Training loss v/s epochs, *Right*: Test acc. v/s epochs.

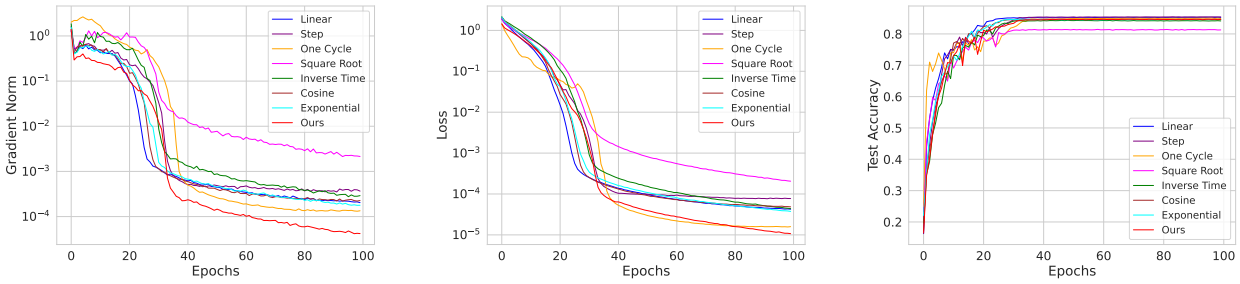

Figure 10: *Comparison with schedulers:* **Mini-batch** experiments on VGG-9 architecture with CIFAR-10 data. *Left*: Grad. norm v/s epochs, *Middle*: Training loss v/s epochs, *Right*: Test acc. v/s epochs.

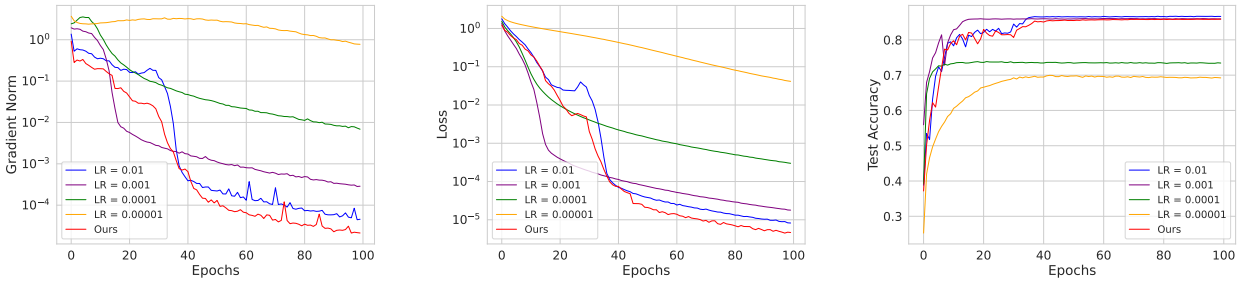

Figure 11: *Comparison with constant step sizes:* **Mini-batch** experiments on VGG-9 architecture with CIFAR-10 data. *Left*: Grad. norm v/s epochs, *Middle*: Training loss v/s epochs, *Right*: Test acc. v/s epochs.

6 Conclusion and Limitations

This article presents the first known theoretical guarantees for the convergence of Adam with an exact constant step size in non-convex settings. It provides insights into why using non-decreasing step sizes may yield sub-optimal results and advocates for sticking to a fixed step size for convergence in Adam. **(i)** We analysed Adam's convergence properties and established a simple condition on the step size that depends on the characteristics of the model and data (in the form of the loss's Lipschitz smoothness and initial loss value). This condition ensures convergence in both deterministic and stochastic non-convex settings. We tag it as a sufficient condition **SC** in our text. **(ii)** We proposed a novel method for efficiently approximating the Lipschitz constant of the loss function with respect to the parameters, which is crucial for our proposed learning rate. **(iii)** Our empirical findings suggest that even with the accumulation of the past few gradients, the key driver for convergence in Adam is the non-increasing nature of step sizes. **(iv)** Finally, our theoretical claims are validated through extensive experiments on training deep neural networks on various datasets. Our experiments validate the effectiveness of the proposed step size. It drives the gradient norm towards zero more aggressively than the commonly used schedulers and a range of arbitrarily chosen constant step sizes. Our derived step size is easy to use and estimate and can be used to train a wide range of tasks.

Limitations: **(i)** It is still open to tightening the convergence rate of Adam. **(ii)** In **Section** 5.3, we empirically demonstrate that increasing the batch size leads to improved convergence. In our proof for the convergence of stochastic Adam, we chose to exclude the batch size factor for simplicity. However, we aim to theoretically demonstrate in the future that, according to our analysis, increasing batch size results in faster convergence.

References

Anas Barakat and Pascal Bianchi. Convergence rates of a momentum algorithm with bounded adaptive step size for nonconvex optimization. In Sinno Jialin Pan and Masashi Sugiyama (eds.), *Proceedings of The*

12th Asian Conference on Machine Learning, volume 129 of *Proceedings of Machine Learning Research*, pp. 225–240. PMLR, 18–20 Nov 2020. URL https://proceedings.mlr.press/v129/barakat20a.html.

Léon Bottou. Curiously fast convergence of some stochastic gradient descent algorithms. In *Proceedings of the symposium on learning and data science*, 2009.

Léon Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*. Springer, 2012.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Congliang Chen, Li Shen, Fangyu Zou, and Wei Liu. Towards practical adam: Non-convexity, convergence theory, and mini-batch acceleration. *The Journal of Machine Learning Research*, 23(1):10411–10457, 2022.

Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of adam-type algorithms for non-convex optimization. *arXiv preprint arXiv:1808.02941*, 2018.

Soham De, Abhay Yadav, David Jacobs, and Tom Goldstein. Automated inference with adaptive batches. In *Artificial Intelligence and Statistics*, 2017.

Soham De, Anirbit Mukherjee, and Enayat Ullah. Convergence guarantees for rmsprop and adam in non-convex optimization and an empirical comparison to nesterov acceleration. *arXiv preprint arXiv:1807.06766*, 2018.

Alexandre Défossez, Léon Bottou, Francis Bach, and Nicolas Usunier. A simple convergence proof of adam and adagrad. *arXiv preprint arXiv:2003.02395*, 2020.

Timothy Dozat. Incorporating nesterov momentum into adam, 2016.

Aaron Defazio *et al.* Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014.

Babanezhad Harikandeh *et al.* Stopwasting my gradients: Practical svrg. *Advances in Neural Information Processing Systems*, 2015.

Jeremy Bernstein *et al.* signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, 2018.

John Johnson *et al.* Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 2011.

Mahyar Fazlyab *et al.* Efficient and accurate estimation of lipschitz constants for deep neural networks. *Advances in Neural Information Processing Systems*, 2019a.

Rie Johnson *et al.* Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 2013.

Sharan Vaswani *et al.* Fast and faster convergence of sgd for over-parameterized models and an accelerated perceptron. In *The 22nd international conference on artificial intelligence and statistics*, 2019b.

Tijmen Tieleman *et al.* Rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA Neural Networks Mach. Learn*, 17, 2012.

Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/95e1533eb1b20a97777749fb94fdb944-Paper.pdf.

Herbert Federer. *Geometric measure theory*. Springer, 2014.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110:393–416, 2021.

Elad Hazan, Kfir Levy, and Shai Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. *Advances in neural information processing systems*, 28, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Li *et al.* Jing. Implicit rank-minimizing autoencoder. In *Advances in Neural Information Processing Systems*, pp. 14736–14746, 2020.

DP Kingma and J Ba. Adam: A method for stochastic optimization,"u 3rd international conference for learning representations, 2015.

Fabian Latorre, Paul Rolland, and Volkan Cevher. Lipschitz constant estimation of neural networks via sparse polynomial optimization. *arXiv preprint arXiv:2004.08688*, 2020.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 661–670, 2014.

Yunfan Li, Peng Hu, Zitao Liu, Dezhong Peng, Joey Tianyi Zhou, and Xi Peng. Contrastive clustering. In *Proceedings of the AAAI conference on artificial intelligence*, pp. 8547–8555, 2021.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. *arXiv preprint arXiv:1902.09843*, 2019.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, 2015.

H Brendan McMahan and Matthew Streeter. Adaptive bound optimization for online convex optimization. *arXiv preprint arXiv:1002.4908*, 2010.

Herman Müntz et al. Solution directe de l'équation séculaire et de quelques problemes analogues transcendants. *CR Acad. Sci. Paris*, 156:43–46, 1913.

J Reddi Sashank, Kale Satyen, and Kumar Sanjiv. On the convergence of adam and beyond. In *International conference on learning representations*, 2018.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Naichen Shi, Dawei Li, Mingyi Hong, and Ruoyu Sun. Rmsprop converges with proper hyper-parameter. In *International Conference on Learning Representations*, 2020.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pp. 369–386. SPIE, 2019.

Ran Tian and Ankur P Parikh. Amos: An adam-style optimizer with adaptive weight decay towards model-oriented scale. *arXiv preprint arXiv:2210.11693*, 2022.

Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper_files/paper/2018/file/d54e99a6c03704e95e6965532dec148b-Paper.pdf`.

Philip Wolfe. Convergence conditions for ascent methods. *SIAM review*, 11(2):226–235, 1969.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Zhiming Zhou, Qingru Zhang, Guansong Lu, Hongwei Wang, Weinan Zhang, and Yong Yu. Adashift: Decorrelation and convergence of adaptive learning rate methods. *arXiv preprint arXiv:1810.00143*, 2018.

Fangyu Zou, Li Shen, Zequn Jie, Weizhong Zhang, and Wei Liu. A sufficient condition for convergences of adam and rmsprop. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pp. 11127–11135, 2019.