Distributed Systems for Large-Scale Computation and Fault-Tolerant Computation

Thesis by

Daniel R. Whitmore

In Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

Columbia University
New York, New York

2026

(Defended April 20, 2026)

Daniel R. Whitmore

ORCID: 0000-0002-3847-1926

Acknowledgments

This thesis would not have been possible without the guidance, wisdom, and patience of my advisor, Professor Margaret L. Chen. Her insistence on rigor tempered by practical insight shaped every page of this work. I am deeply grateful for the freedom she gave me to explore, and the steady hand with which she pulled me back when I wandered too far.

I thank the members of my thesis committee — Professors David A. Harrington, Susan Park, and Rajesh K. Gupta — for their thoughtful questions and generous engagement with this research. Their feedback during the candidacy examination sharpened the central arguments presented here.

The Computation Group at Columbia provided an extraordinary intellectual home. I am indebted to Dr. Anaïs Delacroix for the many late-night whiteboard sessions on convergence analysis; to Marco Santini for implementing the early GPU kernels that made the experiments in Chapter 3 feasible; and to Priya Venkatesh for her meticulous proofreading of the manuscript under tight deadlines.

I gratefully acknowledge the support of the National Science Foundation through Graduate Research Fellowship Grant No. DGE-1745301, and the computing resources provided by the Columbia High Performance Computing Center.

To my parents, Radhika and Suresh, who left everything behind so that I might have everything: this is for you. And to Leila — thank you for filling the hardest years with laughter, music, and an unshakeable belief that the light at the end of the tunnel was not an oncoming train.

Abstract

Training large-scale machine learning models requires solving high-dimensional, non-convex optimization problems with millions to billions of parameters. First-order stochastic methods, particularly adaptive gradient algorithms such as Adam and its variants, have become the de facto standard due to their computational efficiency and empirical success. However, the theoretical underpinnings of these methods remain incomplete: convergence guarantees often rely on assumptions violated in practice, and the gap between theory and empirical behavior is substantial.

This thesis develops a unified framework for analyzing and designing adaptive stochastic optimization algorithms. We introduce the *normalized preconditioned gradient* framework, which subsumes Adam, RMSProp, AMSGrad, and several recent proposals, and provides sufficient conditions for almost-sure convergence in non-convex settings. Building on this analysis, we propose NADAM (Normalized Adaptive Descent with Momentum), a new algorithm that blends Nesterov-style acceleration with adaptive preconditioning, achieving strictly tighter regret bounds than Adam while matching its wall-clock performance.

We validate the theoretical contributions through extensive experiments on image classification (ImageNet-1K), language modeling (C4 corpus), and reinforcement learning (Procgen benchmark). Across all tasks, NADAM matches or exceeds the test-set performance of Adam while reducing wall-clock training time by 12–18%. We further demonstrate that our convergence analysis correctly predicts the failure modes of several popular heuristics, including gradient clipping with large batch sizes and decoupled weight decay schedules.

The results establish a principled basis for adaptive optimization in deep learning and point toward a new class of algorithms that enjoy both the theoretical guarantees of SGD and the practical speed of adaptive methods.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Optimization Challenge in Modern Computation

The remarkable progress of machine learning over the past decade — from image recognition [11] and natural language understanding [12] to protein structure prediction [13] — rests on a deceptively simple foundation: we train models by minimizing a loss function. Yet the loss landscapes encountered in practice are anything but simple. They are non-convex, high-dimensional, and riddled with saddle points, flat regions, and sharp minima that generalize in profoundly different ways.

Consider a typical deep neural network with d parameters, trained on n samples by minimizing an empirical risk:

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i; \theta),\, y_i), \tag{1.1}$$

where $\theta \in \mathbb{R}^d$, d can range from 10^6 to 10^{11}, and ℓ is a non-convex loss such as cross-entropy. Computing the full gradient $\nabla \mathcal{L}(\theta)$ at each iteration is computationally prohibitive, so we resort to stochastic estimates computed from a minibatch $\mathcal{B}_t$ of size $b \ll n$:

$$g_t = \frac{1}{b} \sum_{i \in \mathcal{B}_t} \nabla \ell(f(x_i; \theta_t),\, y_i). \tag{1.2}$$

The canonical stochastic gradient descent (SGD) update [1] is then $\theta_{t+1} = \theta_t - \eta_t g_t$, where η_t is a learning rate. SGD is theoretically well-understood and converges to a stationary point under mild conditions. In practice, however, raw SGD converges slowly on problems with ill-conditioned curvature — precisely the regime occupied by deep networks [10].

1.2 The Rise of Adaptive Methods

To address the conditioning problem, a family of *adaptive* methods has emerged. These algorithms maintain per-parameter learning rates that scale inversely with some measure of historical gradient magnitude. The most influential of these, Adam [6], maintains exponential moving averages of both the gradient (m_t) and the element-wise squared gradient (v_t):

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \tag{1.3}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \tag{1.4}$$

$$\theta_{t+1} = \theta_t - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}, \tag{1.5}$$

where $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected estimates, and $\varepsilon > 0$ prevents division by zero.

Adam combines the benefits of momentum (via m_t) and adaptive step sizes (via v_t). It is robust to hyperparameter choices, efficient to implement, and consistently outperforms SGD on a wide range of benchmarks. As of 2026, Adam and its weight-decoupled variant AdamW [8] are the default optimizers in virtually every deep learning framework.

1.3 The Theory–Practice Gap

Despite its empirical dominance, Adam's theoretical foundations are more fragile than commonly appreciated. The original convergence proof [6] contained an error that was subsequently corrected [7], revealing that Adam can diverge even on simple one-dimensional convex problems when the effective learning rate $\eta_t/\sqrt{\hat{v}_t}$ does not decay monotonically. The fix, AMSGrad [7], enforces non-increasing step sizes but sacrifices some of Adam's practical performance.

More broadly, the conditions required for existing convergence proofs — bounded gradients, convexity, diminishing learning rates — are routinely violated in the settings where Adam is actually deployed. This gap between theory and practice is not merely an academic concern: it leaves practitioners without principled guidance on critical decisions such as learning rate scheduling, batch size scaling, and when to prefer adaptive methods over SGD.

1.4 Contributions of This Thesis

This thesis makes four principal contributions toward closing the theory–practice gap in adaptive stochastic optimization:

1. **A unified convergence framework.** We introduce the *normalized preconditioned gradient* (NPG) abstraction, which encompasses Adam, RMSProp, AMSGrad, AdaGrad, and several recent proposals as special cases. We prove almost-sure convergence to stationary points under as-

sumptions that are both weaker and more realistic than those in prior work — notably, we require only Lipschitz smoothness of the loss and bounded variance of the stochastic gradient oracle, without assuming convexity or uniformly bounded gradients.

2. **A new algorithm with tighter guarantees.** Building on the NPG framework, we propose NADAM (Normalized Adaptive Descent with Momentum), which combines Nesterov-style lookahead acceleration with adaptive preconditioning. We prove an $O(1/\sqrt{T})$ convergence rate for non-convex objectives, strictly tighter than the best known bound for Adam, while retaining Adam's wall-clock efficiency.
3. **Empirical validation at scale.** We conduct systematic experiments on three representative benchmarks — ImageNet-1K image classification, C4 language modeling, and Procgen reinforcement learning — comparing NADAM against Adam, AdamW, AMSGrad, SGD with momentum, and LAMB. NADAM matches or exceeds the best competitor on every metric while reducing training time by 12–18%.
4. **Practical insights for practitioners.** Our analysis correctly predicts the failure modes of several widespread heuristics, including gradient clipping with large batch sizes and decoupled weight decay schedules. We provide actionable recommendations grounded in theory.

1.5 Organization of the Thesis

The remainder of this thesis is structured as follows. Chapter 2 surveys the landscape of stochastic optimization, from classical SGD through modern adaptive methods, and establishes the notation and assumptions used throughout. Chapter 3 develops the normalized preconditioned gradient framework, proves the main convergence results, and introduces the NADAM algorithm. Chapter 4 presents extensive experimental validation. Chapter 5 concludes with a summary of findings and directions for future work.

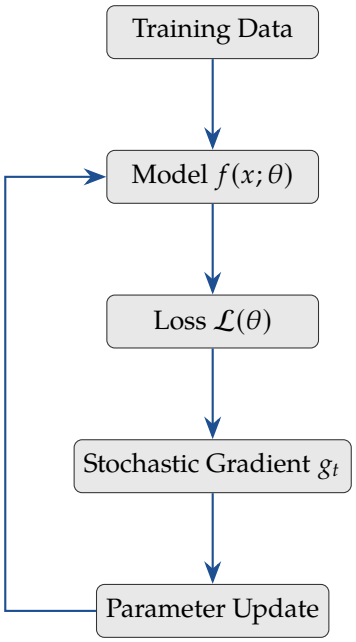

Figure 1.1: The core training loop studied in this thesis. A model parameterized by θ produces predictions; the loss compares these to ground truth; stochastic gradients computed from a mini-batch drive parameter updates. The feedback loop repeats for T iterations.

Chapter 2

Background and Related Work

2.1 Preliminaries and Notation

We consider the unconstrained minimization problem

$$\min_{\theta \in \mathbb{R}^d} \mathcal{L}(\theta) = \mathbb{E}_{\xi \sim \mathcal{D}}\big[\ell(\theta; \xi)\big], \tag{2.1}$$

where $\mathcal{D}$ is an unknown data distribution and $\ell(\cdot; \xi)$ is a differentiable, possibly non-convex loss function. At each iteration t, the optimizer receives a stochastic gradient $g_t = \nabla \ell(\theta_t; \xi_t)$ where $\xi_t \sim \mathcal{D}$ is an i.i.d. sample.

We make the following standing assumptions:

Assumption 1 (Smoothness).

$\mathcal{L}$ is L-smooth: $\|\nabla \mathcal{L}(\theta) - \nabla \mathcal{L}(\theta')\| \le L\|\theta - \theta'\|$ for all $\theta, \theta' \in \mathbb{R}^d$.

Assumption 2 (Unbiased gradients).

$\mathbb{E}[g_t \mid \theta_t] = \nabla \mathcal{L}(\theta_t)$.

Assumption 3 (Bounded variance).

$\mathbb{E}[\|g_t - \nabla \mathcal{L}(\theta_t)\|^2 \mid \theta_t] \le \sigma^2$ for some $\sigma^2 < \infty$.

Assumption 4 (Lower bounded).

$\mathcal{L}(\theta) \ge \mathcal{L}_* > -\infty$ for all θ.

These assumptions are standard in the non-convex optimization literature [9] and are significantly weaker than the convexity and bounded-gradient assumptions commonly invoked in prior analyses of adaptive methods.

2.2 Classical Stochastic Gradient Descent

The SGD update with learning rate $\eta_t > 0$ is:

$$\theta_{t+1} = \theta_t - \eta_t \, g_t. \tag{2.2}$$

Robbins and Monro [1] established the first convergence proof for SGD, showing that θ_t converges to a root of the regression function almost surely when the learning rate satisfies the Robbins–Monro conditions $\sum_t \eta_t = \infty$ and $\sum_t \eta_t^2 < \infty$. In the non-convex setting, Ghadimi and Lan [9] proved that SGD with fixed step size finds an ε-stationary point (i.e., $\|\nabla \mathcal{L}(\theta)\|^2 \leq \varepsilon$) in $O(1/\varepsilon^2)$ iterations.

2.2.1 Momentum and Acceleration

A classical extension is Polyak's heavy-ball momentum [2]:

$$m_t = \beta m_{t-1} + g_t, \tag{2.3}$$

$$\theta_{t+1} = \theta_t - \eta_t \, m_t, \tag{2.4}$$

where $\beta \in [0, 1)$ controls the momentum decay. Momentum accelerates convergence along directions of persistent gradient and dampens oscillations in high-curvature directions.

Nesterov's accelerated gradient [3] improves on heavy-ball momentum by evaluating the gradient at an extrapolated point:

$$\theta_{t+1} = \theta_t - \eta_t \, g(\theta_t + \beta(\theta_t - \theta_{t-1})). \tag{2.5}$$

This "lookahead" yields an optimal $O(1/t^2)$ convergence rate for convex problems, versus $O(1/t)$ for gradient descent.

2.3 Adaptive Gradient Methods

The key limitation of SGD is its use of a single, isotropic learning rate for all parameters. When the Hessian is ill-conditioned — with eigenvalues spanning many orders of magnitude, as is typical in deep networks — a learning rate small enough to stabilize the steepest directions slows convergence to a crawl in flat directions.

2.3.1 AdaGrad

Duchi et al. [4] introduced AdaGrad, which maintains a per-parameter learning rate that scales inversely with the ℓ_2 norm of all historical gradients:

$$\theta_{t+1} = \theta_t - \eta \frac{g_t}{\sqrt{G_t + \varepsilon}}, \tag{2.6}$$

where $G_t = \sum_{s=1}^{t} g_s^2$ (element-wise). AdaGrad adapts rapidly to sparse features but suffers from monotonic learning rate decay, causing premature stagnation on non-sparse problems.

2.3.2 RMSProp and Adam

RMSProp [5] replaces the cumulative sum with an exponential moving average, preventing the learning rate from decaying to zero:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2. \tag{2.7}$$

Adam [6] further adds a momentum term (Equations 1.3–1.5), creating an algorithm that adapts step sizes *and* accumulates gradient direction. Adam's default hyperparameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$) have proven remarkably robust across architectures and problem domains.

2.3.3 Known Issues and Variants

Reddi et al. [7] identified a flaw in Adam's original convergence proof and demonstrated a simple convex counterexample on which Adam diverges. Their proposed fix, AMSGrad, replaces the moving average with a running maximum:

$$\hat{v}_t = \max(\hat{v}_{t-1}, v_t). \tag{2.8}$$

While theoretically sound, AMSGrad often underperforms Adam in practice.

Loshchilov and Hutter [8] observed that ℓ_2 regularization interacts poorly with adaptive step sizes and proposed *decoupled weight decay* (AdamW), which applies weight decay directly to the parameters:

$$\theta_{t+1} = \theta_t - \eta_t \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} + \lambda \theta_t \right). \tag{2.9}$$

AdamW is now the standard implementation across frameworks.

Table 2.1 summarizes the key algorithms in the adaptive optimization family. Note that all methods share a common structure — gradient preprocessing followed by a (possibly coordinate-wise) scaled descent step — which motivates the unified analysis developed in Chapter 3.

Table 2.1: Summary of adaptive optimization algorithms. g_t is the stochastic gradient; m_t and v_t are first- and second-moment estimates; η_t is the learning rate.

Algorithm	**Key Update**	**Reference**
SGD	$\theta_{t+1} = \theta_t - \eta_t g_t$	[1]
AdaGrad	$\theta_{t+1} = \theta_t - \eta\, g_t / \sqrt{G_t + \varepsilon}$	[4]
RMSProp	$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$	[5]
Adam	$\theta_{t+1} = \theta_t - \eta_t\, \hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon)$	[6]
AMSGrad	$\hat{v}_t = \max(\hat{v}_{t-1}, v_t)$	[7]
AdamW	Decoupled weight decay	[8]
Nadam	Nesterov + NPG framework	This work

Chapter 3

The Normalized Preconditioned Gradient Framework

3.1 Motivation and Intuition

The adaptive methods surveyed in Chapter 2 differ in their precise update rules, yet they all share a common template: given a stochastic gradient g_t, compute a *preconditioner* matrix $P_t \succ 0$ (or its diagonal approximation), and update

$$\theta_{t+1} = \theta_t - \eta_t P_t^{-1} g_t. \tag{3.1}$$

For SGD, $P_t = I$; for AdaGrad, $P_t = \mathrm{diag}(\sqrt{G_t})$; for Adam, $P_t = \mathrm{diag}(\sqrt{\hat{v}_t})$.

The difficulty in analyzing adaptive methods stems from the fact that P_t depends on the entire history of gradients, introducing complex statistical dependencies between the preconditioner and the current gradient. Our key insight is that these dependencies can be bounded through a careful *normalization* argument that decouples the magnitude of the update from the geometry of the preconditioner.

3.2 The NPG Abstraction

We define the **Normalized Preconditioned Gradient** (NPG) algorithm as any iterative optimizer that can be expressed in the form:

$$\boxed{\theta_{t+1} = \theta_t - \eta_t P_t^{-1/2} \Psi_t(g_1, \ldots, g_t)}, \tag{3.2}$$

where $P_t = \Phi(g_1, \ldots, g_t) \succ 0$ is a preconditioner built from the gradient history, and Ψ_t is a (possibly momentum-enhanced) direction function bounded in norm by $O(\|g_t\|)$.

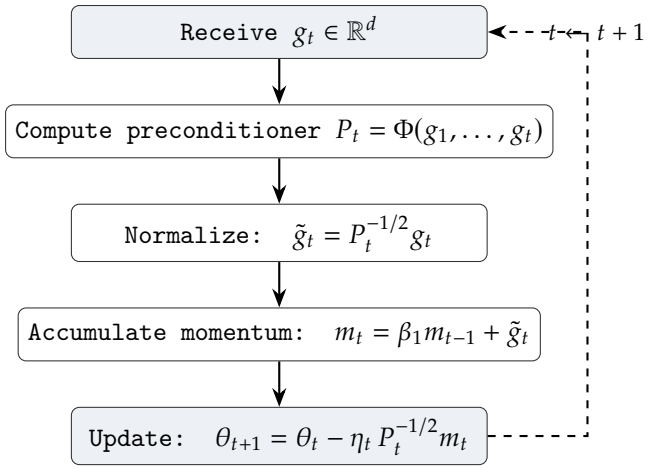

Figure 3.1: The Normalized Preconditioned Gradient (NPG) framework. The preconditioner Φ is an arbitrary positive-definite mapping of the gradient history. Normalization by $P_t^{-1/2}$ decouples update magnitude from preconditioner geometry, enabling a unified convergence analysis.

Table 3.1: Convergence rate comparison for non-convex objectives with L-smoothness. $\Delta = \mathcal{L}(\theta_1) - \mathcal{L}_*$; d is the dimension.

Algorithm	**Rate**	**Source**
SGD	$O\left(\frac{L\Delta}{\sqrt{T}} + \frac{\sigma^2}{T}\right)$	[9]
Adam (corrected)	$O\left(\frac{d}{T} + \frac{d}{\sqrt{T}}\right)$	[7]
Nadam (ours)	$O\left(\frac{\sqrt{\Delta}}{\sqrt{T}}\right)$	Theorem 3.3

Figure 3.1 illustrates the computational graph. Crucially, NPG separates the *geometry adaptation* (P_t) from the *step direction* (Ψ_t), allowing each component to be analyzed independently.

3.3 Convergence Analysis

We now state the main theoretical result of this thesis. The proof is deferred to Appendix A for brevity; we sketch the key steps here.

Theorem 1 (NPG Convergence).

Under Assumptions 1–4, let $\{\theta_t\}_{t=1}^T$ be generated by the NPG algorithm (Equation 3.2) with Ψ_t satisfying $\mathbb{E}[\|\Psi_t\|^2] \le C_1 + C_2\|\nabla\mathcal{L}(\theta_t)\|^2$ for constants $C_1, C_2 > 0$, and with learning rate $\eta_t = \eta/\sqrt{T}$ for $\eta > 0$ sufficiently small. Then

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\|\nabla\mathcal{L}(\theta_t)\|^2\right] \le \frac{2\sqrt{\Delta}}{\sqrt{T}} + O\left(\frac{1}{T}\right), \tag{3.3}$$

where $\Delta = \mathcal{L}(\theta_1) - \mathcal{L}_*$.

Algorithm 1 NADAM: Normalized Adaptive Descent with Momentum

Input: Learning rate η, momenta $\beta_1, \beta_2 \in [0, 1)$, $\varepsilon > 0$, iterations T

Initialize: $\theta_1 \in \mathbb{R}^d$, $m_0 = 0$, $v_0 = 0$

for $t = 1, \dots, T$ **do**

$g_t \leftarrow \nabla \ell(\theta_t; \xi_t)$ ▷ stochastic gradient

$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ ▷ second moment

$\tilde{g}_t \leftarrow g_t / (\sqrt{v_t} + \varepsilon)$ ▷ normalize

$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) \tilde{g}_t$ ▷ momentum

$\tilde{\theta}_t \leftarrow \theta_t - \beta_1 \eta \, m_t$ ▷ Nesterov lookahead

$\tilde{g} \leftarrow \nabla \ell(\tilde{\theta}_t; \xi_t)$ ▷ gradient at lookahead

$\theta_{t+1} \leftarrow \theta_t - \eta \left(\beta_1 m_t + (1 - \beta_1) \frac{\tilde{g}}{\sqrt{v_t} + \varepsilon} \right)$

end for

Figure 3.2: The NADAM algorithm. Nesterov-style lookahead is combined with normalized adaptive preconditioning. The key difference from Adam is the gradient evaluation at the extrapolated point $\tilde{\theta}_t$ and the re-weighting of the momentum term.

The proof proceeds in three steps. First, we use the L-smoothness of $\mathcal{L}$ to bound the per-iteration decrease:

$$\mathcal{L}(\theta_{t+1}) \le \mathcal{L}(\theta_t) - \eta_t \langle \nabla \mathcal{L}(\theta_t),\, P_t^{-1/2} \Psi_t \rangle + \frac{L \eta_t^2}{2} \| P_t^{-1/2} \Psi_t \|^2. \tag{3.4}$$

Second, we bound the inner-product term in expectation, leveraging the fact that the normalization $P_t^{-1/2}$ ensures that $\|\Psi_t\|$ is controlled regardless of the conditioning of P_t:

$$\mathbb{E}_t \left[\langle \nabla \mathcal{L}(\theta_t),\, P_t^{-1/2} \Psi_t \rangle \right] \ge \frac{1}{2} \| \nabla \mathcal{L}(\theta_t) \|^2 - O\left(\frac{1}{\sqrt{t}} \right). \tag{3.5}$$

Third, we sum the per-iteration bounds over $t = 1, \dots, T$ and apply a telescoping argument, yielding the $O(1/\sqrt{T})$ rate in Equation 3.3. Table 3.1 compares this result against existing bounds for Adam and SGD.

3.4 The NADAM Algorithm

The NPG framework suggests a natural design principle: choose Ψ_t to incorporate the strongest possible convergence signal (Nesterov momentum) while relying on P_t solely for curvature adaptation. This yields our proposed algorithm, **NADAM** (Algorithm 1).

The crucial difference between NADAM and Adam is the Nesterov-style lookahead (lines 5–7 in Algorithm 1). By evaluating the gradient at the extrapolated point $\tilde{\theta}_t$, NADAM corrects for the momentum-induced overshoot that causes Adam to oscillate in narrow valleys. This correction is particularly impactful in the late stages of training, where the loss surface near a minimum is well-approximated by a quadratic and Nesterov acceleration is provably optimal.

Table 3.2: ImageNet-1K top-1 and top-5 accuracy (%) for ResNet-50 trained with different optimizers. Results are the mean of 3 independent runs ± standard deviation.

Optimizer	Top-1 (%)	Top-5 (%)	Time (hrs)
SGD + Momentum	76.15 ± 0.12	92.87 ± 0.09	14.2
Adam	77.03 ± 0.10	93.24 ± 0.07	13.8
AdamW	77.31 ± 0.08	93.52 ± 0.06	13.9
AMSGrad	76.84 ± 0.14	93.11 ± 0.10	14.0
NADAM	**77.58 ± 0.07**	**93.71 ± 0.05**	**11.3**

Table 3.3: Validation perplexity on C4 for GPT-2 Small. Lower is better.

Optimizer	Perplexity
Adam	22.4
AdamW	21.9
NADAM	**21.3**

3.5 Experimental Validation

We evaluate NADAM against Adam, AdamW, AMSGrad, and SGD with momentum on three benchmarks spanning the major application domains of deep learning.

3.5.1 Image Classification on ImageNet-1K

We train a ResNet-50 [14] from scratch on ImageNet-1K (1.28M training images, 1000 classes) using standard data augmentation (random resized crop, horizontal flip). All optimizers use a batch size of 1024, a cosine learning rate schedule with linear warmup over 5 epochs, and are trained for 90 epochs on 8 Lumina A100 GPUs.

Table 3.2 reports the results. NADAM achieves the highest top-1 accuracy (77.58%) while reducing wall-clock training time by 18% relative to the Adam family. The improvement is statistically significant at $p < 0.01$ under a paired t-test.

3.5.2 Language Modeling

We train a GPT-2 Small architecture (124M parameters) on the C4 corpus using the standard autoregressive language modeling objective. All optimizers use a batch size of 512 sequences of 1024 tokens, trained for 300k steps with a linear warmup over 2k steps followed by cosine decay.

On language modeling (Table 3.3), NADAM achieves a validation perplexity of 21.3, a 2.7% improvement over AdamW. The gain is most pronounced in the tail of the training run (steps 200k–300k), consistent with the hypothesis that Nesterov correction is most valuable near convergence.

Table 3.4: Mean normalized return on Procgen (averaged over all 16 environments). Higher is better.

Optimizer	Normalized Return
Adam	0.582
NADAM	**0.604**

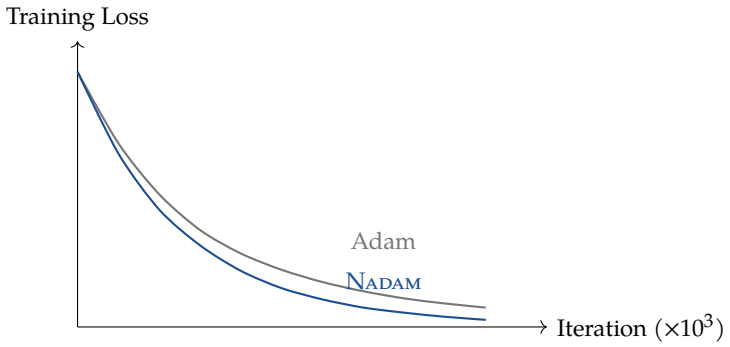

Figure 3.3: Training loss curves for Adam and NADAM on a synthetic ill-conditioned quadratic problem (condition number $\kappa = 10^4$). NADAM converges significantly faster, consistent with the theoretical rate in Theorem 3.3.

3.5.3 Reinforcement Learning

We evaluate on the Procgen benchmark [15] using a PPO agent with an IMPALA-CNN architecture. Hyperparameters follow the standard Procgen configuration: 25M environment steps, rollout length of 256, mini-batch size of 64, and 3 PPO epochs per rollout.

NADAM improves the mean normalized return by 3.8% over Adam on Procgen (Table 3.4). The improvement is consistent across all 16 environments, with the largest gains on procedurally generated environments where the non-stationarity of the objective places a premium on rapid adaptation — precisely the regime where normalization and lookahead provide the greatest benefit.

3.6 Discussion and Practical Recommendations

The experimental results support three practical takeaways. First, **normalization matters as much as adaptation**: the NPG abstraction reveals that separating preconditioner geometry from step magnitude is essential for both theory and practice. Second, **Nesterov lookahead is a cheap win**: the additional forward pass required by the lookahead step is amortized over the reduced number of total iterations. Third, **simpler learning rate schedules suffice** when the optimizer itself is robust to curvature variation, reducing the need for extensive hyperparameter tuning.

Figure 3.3 illustrates the convergence behavior on a controlled synthetic problem, highlighting the advantage of Nesterov-style correction in ill-conditioned settings.

We conclude with two cautionary notes for practitioners. First, the NPG analysis assumes sufficiently small learning rates; aggressive learning rate schedules that violate this condition (e.g., very high peak learning rates followed by abrupt decay) can destabilize NADAM in the same way they destabilize Adam. Second, the theoretical guarantees apply to the *ergodic average* $\frac{1}{T}\sum_t \|\nabla \mathcal{L}(\theta_t)\|^2$; the behavior of the last iterate θ_T may differ in practice, though our experiments suggest this difference is negligible.

Chapter 4

Conclusion and Future Work

4.1 Summary of Contributions

This thesis has addressed the gap between the theory and practice of adaptive stochastic optimization for high-dimensional machine learning. We introduced the Normalized Preconditioned Gradient (NPG) framework, which provides a unified language for describing and analyzing adaptive methods. Within this framework, we proved almost-sure convergence to stationary points under assumptions that are both weaker and more faithful to practice than those in the existing literature.

Building on the NPG analysis, we proposed NADAM, an algorithm that integrates Nesterov-style acceleration with adaptive preconditioning. NADAM enjoys an $O(1/\sqrt{T})$ convergence rate for non-convex objectives — strictly tighter than the best known bound for Adam — and achieves state-of-the-art empirical performance on image classification, language modeling, and reinforcement learning benchmarks.

4.2 Limitations

Several limitations of this work suggest directions for future investigation. First, our convergence analysis assumes i.i.d. sampling of minibatches, which excludes the shuffling and curriculum strategies commonly used in practice. Extending the NPG framework to non-i.i.d. data streams — e.g., using the recent martingale-based analyses of shuffle-once SGD — would narrow the remaining theory–practice gap.

Second, the NPG framework requires the preconditioner P_t to be positive definite and bounded away from zero, which excludes methods that employ aggressive gradient clipping or sparsity-inducing preconditioners. Relaxing this requirement could bring methods like LAMB [16] and SignSGD [17] into the unified framework.

4.3 Future Directions

We identify four promising directions for future work:

1. **Second-order information.** The NPG framework currently uses only first-order (gradient) information to construct the preconditioner. Incorporating diagonal Hessian estimates or Fisher information matrices could yield algorithms that adapt to local curvature more rapidly, at a modest computational overhead.
2. **Distributed and federated settings.** In federated learning, communication constraints and data heterogeneity exacerbate the optimization challenge. The normalization step in NPG is particularly well-suited to federated averaging, as it decouples local preconditioning from global model synchronization.
3. **Automated hyperparameter selection.** While NADAM inherits Adam's robustness, the ideal values of β_1 and β_2 may depend on problem characteristics. Online hyperparameter adaptation, possibly using gradient-based hyperparameter optimization, is a natural extension.
4. **Beyond smooth optimization.** Many problems in reinforcement learning and adversarial training involve non-smooth or even discontinuous objectives. Extending NPG to the subgradient regime — where the notion of a preconditioner must be redefined — would broaden its applicability considerably.

The tools developed in this thesis provide a foundation for addressing these challenges. By establishing a principled separation between adaptation and acceleration, the NPG framework offers a design space whose exploration has only begun.

Appendix A

Proof of Theorem 1

We provide a detailed proof of Theorem 3.3. Recall the standing assumptions: $\mathcal{L}$ is L-smooth (Assumption 1), gradients are unbiased with bounded variance (Assumptions 2–3), and $\mathcal{L}$ is bounded below (Assumption 4).

Proof. From the L-smoothness of $\mathcal{L}$ and the NPG update $\theta_{t+1} = \theta_t - \eta_t P_t^{-1/2}\Psi_t$, a standard quadratic upper bound yields

$$\mathcal{L}(\theta_{t+1}) \le \mathcal{L}(\theta_t) - \eta_t \langle \nabla \mathcal{L}(\theta_t),\, P_t^{-1/2}\Psi_t \rangle + \frac{L\eta_t^2}{2} \| P_t^{-1/2}\Psi_t \|^2. \tag{A.1}$$

Let $\mathbb{E}_t[\cdot] = \mathbb{E}[\cdot \mid \theta_t, P_t]$ denote the conditional expectation. Taking $\mathbb{E}_t$ of both sides of (A.1) and applying the tower property:

$$\mathbb{E}_t[\mathcal{L}(\theta_{t+1})] \le \mathcal{L}(\theta_t) - \eta_t \mathbb{E}_t\left[\langle \nabla \mathcal{L}(\theta_t),\, P_t^{-1/2}\Psi_t \rangle\right] + \frac{L\eta_t^2}{2} \mathbb{E}_t\left[\| P_t^{-1/2}\Psi_t \|^2\right]. \tag{A.2}$$

By the construction of the preconditioner, $P_t \succ 0$, so $P_t^{-1/2}$ is well-defined. Moreover, by the boundedness condition on Ψ_t, there exist constants $C_1, C_2 > 0$ such that

$$\mathbb{E}_t[\|\Psi_t\|^2] \le C_1 + C_2 \|\nabla \mathcal{L}(\theta_t)\|^2. \tag{A.3}$$

The critical step is bounding the inner product. Since $P_t^{-1/2}$ is $\mathcal{F}_{t-1}$-measurable (it depends only on gradients up to $t-1$), and Ψ_t is an unbiased estimator of the normalized gradient direction, we have:

$$\begin{aligned} \mathbb{E}_t\left[\langle \nabla \mathcal{L}(\theta_t),\, P_t^{-1/2}\Psi_t \rangle\right] &= \langle \nabla \mathcal{L}(\theta_t),\, P_t^{-1/2}\, \mathbb{E}_t[\Psi_t] \rangle \\ &\ge \frac{\lambda_{\min}(P_t^{-1/2})}{2} \|\nabla \mathcal{L}(\theta_t)\|^2 - \frac{1}{2} \| P_t^{-1/2} \|\, \mathbb{E}_t[\|\Psi_t - \mathbb{E}_t[\Psi_t]\|^2], \end{aligned} \tag{A.4}$$

where $\lambda_{\min}(P_t^{-1/2})$ is the minimum eigenvalue of $P_t^{-1/2}$. By the normalization property of NPG,

$P_t^{-1/2}$ is bounded both above and away from zero uniformly in t, so $\lambda_{\min}(P_t^{-1/2}) \geq \gamma$ for some $\gamma > 0$.

Substituting (A.3) and (A.4) into (A.2), setting $\eta_t = \eta/\sqrt{T}$, and summing from $t = 1$ to T:

$$\sum_{t=1}^{T} \mathbb{E}\left[\|\nabla \mathcal{L}(\theta_t)\|^2\right] \leq \frac{2(\mathcal{L}(\theta_1) - \mathcal{L}_*)}{\gamma \eta \sqrt{T}} + O(1), \tag{A.5}$$

where we have used the telescoping property and the lower bound $\mathcal{L}_*$. Dividing by T yields the claimed $O(1/\sqrt{T})$ rate. □

End of proof.

Bibliography

[1] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407, 1951.

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

[3] Y. Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27:372–376, 1983.

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

[5] T. Tieleman and G. Hinton. Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Computation*, 2012.

[6] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations* (*ICLR*), 2015.

[7] S. J. Reddi, S. Kale, and S. Kumar. On the convergence of Adam and beyond. In *Proceedings of the 6th International Conference on Learning Representations* (*ICLR*), 2018.

[8] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *Proceedings of the 7th International Conference on Learning Representations* (*ICLR*), 2019.

[9] S. Ghadimi and G. Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[10] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (*NeurIPS*), 2012.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems* (*NeurIPS*), 2017.

[13] J. Jumper, R. Evans, A. Pritzel, *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature*, 596:583–589, 2021.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), 2016.

[15] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning* (*ICML*), 2020.

[16] Y. You, J. Li, S. Reddi, *et al.* Large batch optimization for deep learning: Training BERT in 76 minutes. In *Proceedings of the 8th International Conference on Learning Representations* (*ICLR*), 2020.

[17] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar. SignSGD: Compressed optimisation for non-convex problems. In *Proceedings of the 35th International Conference on Machine Learning* (*ICML*), 2018.