Convergence guarantees for RMSProp and ADAM in non-convex optimization and their comparison to Nesterov acceleration on autoencoders

Amitabh Basu[1], Soham De[2], Anirbit Mukherjee[1], and Enayat Ullah[3]

[1]Department of Applied Mathematics and Statistics, Johns Hopkins University
[2]Department of Computer Science, University of Maryland
[3]Department of Computer Science, Johns Hopkins University
basu.amitabh@jhu.edu, sohamde@cs.umd.edu, amukhe14@jhu.edu, enayat@jhu.edu

July 19, 2018

Abstract

RMSProp and ADAM continue to be extremely popular algorithms for training neural nets but their theoretical foundations have remained unclear. In this work we make progress towards that by giving proofs that these adaptive gradient algorithms are guaranteed to reach criticality for smooth non-convex objectives and we give bounds on the running time.

We then design experiments to compare the performances of RMSProp and ADAM against Nesterov Accelerated Gradient method on a variety of autoencoder setups. Through these experiments we demonstrate the interesting sensitivity that ADAM has to its momentum parameter β_1. We show that in terms of getting lower training and test losses, at very high values of the momentum parameter ($\beta_1 = 0.99$) (and large enough nets if using mini-batches) ADAM outperforms NAG at any momentum value tried for the latter. On the other hand, NAG can sometimes do better when ADAM's β_1 is set to the most commonly used value: $\beta_1 = 0.9$. We also report experiments on different autoencoders to demonstrate that NAG has better abilities in terms of reducing the gradient norms and finding weights which increase the minimum eigenvalue of the Hessian of the loss function.

1 Introduction

Many optimization questions arising in machine learning can be cast as a finite sum optimization problem of the form: $\min_{\mathbf{x}} f(\mathbf{x})$ where $f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x})$ with no convexity assumptions on the functions f_i. A well-studied algorithm to solve such problems is "Stochastic Gradient Descent (SGD)", which uses updates of the form: $\mathbf{x}_{t+1} := \mathbf{x}_t - \alpha\nabla\tilde{f}_{i_t}(\mathbf{x}_t)$, where α is a step size, and $\tilde{f}_{i_t}$ is a function chosen randomly from $\{f_1, f_2, \ldots, f_N\}$ at time t. Often in neural networks, a "momentum" version of SGD is used called the Heavy-Ball (HB) method which is commonly written as a two step update process given as, $\mathbf{v}_{t+1} = \mu\mathbf{v}_t - \alpha\nabla\tilde{f}_{i_t}(\mathbf{x}_t)$ followed by $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{v}_{t+1}$ with an appropriate choice of $\mu > 0$; $\mathbf{v}$ is typically called the "velocity vector". In the context of neural nets another variant of SGD that is popular is the 'Nesterov's Accelerated Gradient (NAG)" method which can also be thought of as a momentum method [SMDH13] and it has a similar two step update of the form $\mathbf{v}_{t+1} = \mu\mathbf{v}_t - \alpha\nabla\tilde{f}_{i_t}(\mathbf{x}_t + \mu\mathbf{v}_t)$ followed by $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{v}_{t+1}$ (see Algorithm 1 in subsection 3.2 for the specific form of NAG used in our experiments).

A deterministic form of HB and NAG method have been shown to have superior convergence rates compared to gradient descent in the convex deterministic setting [Nes83, Pol87]. In the convex stochastic case, on the other hand, there isn't any difference in convergence speeds between SGD and NAG or HB [YYS16, KNJK18, WKH94, OL94]. Convergence properties of HB and NAG have been studied on non-convex objectives in the deterministic setting [ZK93, Och16, OW17, JNJ17] as well as in the non-convex stochastic setting [YLL16, GPS+18].

To the best of our knowledge there is no clear theoretical justification in the non-convex stochastic case of the benefits of NAG and HB over regular SGD. But in practice these momentum methods have been repeatedly shown to be vital to get good performance on deep neural nets starting from the seminal paper of [SMDH13].

The performance of SGD or its momentum variants (NAG and HB), however, are typically quite sensitive to the selection of the hyper-parameters: learning rate (step size), momentum and batch size [SMDH13]. Thus, two other algorithms RMSProp (Algorithm 2) [TH12] and ADAM (Algorithm 3) [KB14] have become very popular for optimizing deep neural networks [MDB17, XBK+15, DN17, GDG+15, RMC15, BAP+17, KZS+15]. The reason for their widespread popularity seems to be the fact that they are significantly easier to tune than SGD or its variants (NAG and HB). These algorithms use as their update direction a vector which is the image under a linear transformation (often called the "diagonal pre-conditioner") constructed out of the history of the gradients, of a linear combination of all the gradients seen till now. The folklore wisdom has been to say that this "pre-conditioning" makes them much less sensitive to the selection of its hyper-parameters

These adaptive gradient algorithms [DHS11] have been heavily used by the deep learning community for the last several years on a variety of tasks. But even very recently it has been noted in [BWAA18] that RMSProp and ADAM lack theoretical justifications in the non-convex setting - even with exact/deterministic gradients. This work of ours attempts to remedy this situation by giving convergence guarantees for these adaptive gradient algorithms in the context of non-convex optimization.

On the other hand, [RKK18] have shown in the setting of convex regret minimization that there are certain sequences of convex regrets where ADAM and RMSProp fail to converge to zero asymptotically average regret. Further, in more realistic settings of stochastic non-convex optimization, it is known that ADAM generalizes poorly for large enough nets [WRS+17].

Works like [WRS+17] and [KS17] have shown cases where SGD (no momentum) and HB (classical momentum) generalize much better than RMSProp and ADAM with stochastic gradients. [WRS+17] also show that RMSProp generalizes better than ADAM on a couple of neural network tasks (most notably in the character-level language modeling task). But in general it's not clear and no heuristics are known to the best of our knowledge to decide whether these insights about relative performances (generalization or training) between algorithms carry over to the full-batch setting. In contrast we demonstrate that for the task of autoencoding on MNIST data with full or mini batch gradients (a) RMSProp fails to generalize pretty soon as the network size keeps increasing and that (b) ADAM's hyperparameters can be set to outperform NAG.

Recently [KNJK18] showed experimental evidence that mini-batch NAG (Algorithm 2 in their paper) performs better than mini-batch SGD. They also suggest a variant that they call Accelerated SGD (ASGD) (Algorithm 3 in their paper) and show that mini-batch ASGD gives a slight increase in generalization performance over mini-batch NAG but with the added advantage of significantly faster rate of reduction in training loss.

In contrast, in our experiments we choose to compare NAG against adaptive gradient methods on a variety of autoencoder setups, in both full and mini-batch settings. In the full-batch setting, we observed that ADAM with very high values of the momentum parameter ($\beta_1 = 0.99$) outperforms NAG with any momentum value, in terms of getting lower training and test losses. In the mini-batch experiments we see exactly the same behaviour for large enough nets. We also point out in our experiments that, in general, NAG has better abilities in terms of reducing the gradient norms and finding weights which increase the minimum eigenvalue of the Hessian of the loss function (in both full and mini-batch settings).

2 Summary of the theoretical results

Firstly we define the smoothness property that we assume in our proofs for all our non-convex objectives.

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

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{L}{2} \|\mathbf{y} - \mathbf{x}\|^2 .$$

Theorem 2.1. ADAM converges to criticality (Proof in subsection 5.1) Let $f : \mathbb{R}^d \to \mathbb{R}$ be $L-$smooth and let $\sigma < \infty$ be an upperbound on the norm of the gradient of f. Assume also that f has a minimizer, i.e., there exists $\mathbf{x}_*$ such that $f(\mathbf{x}_*) = \min_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x})$. Then the following holds for Algorithm 3.

For any $\epsilon > 0$, $\beta_1 < \frac{\epsilon}{\epsilon + \sigma}$ and $\xi > \frac{\sigma^2 \beta_1}{-\beta_1 \sigma + \epsilon(1-\beta_1)}$, there exist step sizes $\alpha_t > 0$, $t = 1, 2, \ldots$ and a natural number T (depending on β_1, ξ) such that $\|\nabla f(\mathbf{x}_t)\| \leq \epsilon$ for some $t \leq T$.

In particular if one sets $\beta_1 = \frac{\epsilon}{\epsilon + 2\sigma}$, $\xi = 2\sigma$, and $\alpha_t = \frac{\|\mathbf{g}_t\|^2}{L(1-\beta_1^t)^2} \frac{4\epsilon}{3(\epsilon+2\sigma)^2}$ where $\mathbf{g}_t$ is the gradient of the objective at the t^{th} iterate, then T can be taken to be $\frac{9L\sigma^2}{\epsilon^6}[f(\mathbf{x}_2) - f(\mathbf{x}_*)]$, where $\mathbf{x}_2$ is the second iterate of the algorithm.

Often in folklore it has been said that ADAM gains over RMSProp because of its “bias correction term" which refers to the step length of ADAM having an iteration dependence of the following form, $\sqrt{1-\beta_2^t}/(1-\beta_1^t)$. In the above theorem, we note that the $1/(1-\beta_1^t)$ term of this "bias correction term" naturally comes out from theory!

Theorem 2.2. Convergence of RMSProp - the version with standard speeds (Proof in subsection 5.2) Let $f : \mathbb{R}^d \to \mathbb{R}$ be $L-$smooth and let $\sigma < \infty$ be an upperbound on the norm of the gradient of f. Assume also that f has a minimizer, i.e., there exists $\mathbf{x}_*$ such that $f(\mathbf{x}_*) = \min_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x})$. Then the following holds for Algorithm 2.

For any $\epsilon, \xi > 0$, using a constant step length of $\alpha_t = \alpha = \frac{(1-\beta_2)\xi}{L\sqrt{\sigma^2+\xi}}$ for $t = 1, 2, ...,$ guarantees that $\|\nabla f(\mathbf{x}_t)\| \leq \epsilon$ for some $t \leq \frac{2L}{\epsilon^2}[f(\mathbf{x}_1) - f(\mathbf{x}_*)] \left(1 + \frac{\sigma^2}{\xi}\right)$, where $\mathbf{x}_1$ is the first iterate of the algorithm.

One might wonder if the ξ parameter introduced in Algorithm 2 is necessary to get convergence guarantees for RMSProp. Towards that in the following theorem we show convergence of another variant of RMSProp which does not use the ξ parameter and instead uses other assumptions on the objective function and step size modulation and defines the “pre-conditioner" via a pseudo-inverse. But these tweaks to eliminate the need of ξ come at the cost of the convergence rates getting weaker.

Theorem 2.3. Convergence of RMSProp - the version with no ξ shift (Proof in subsection 5.3) Let $f : \mathbb{R}^d \to \mathbb{R}$ be $L-$smooth and let $\sigma < \infty$ be an upperbound on the norm of the gradient of f. Assume also that f has a minimizer, i.e., there exists $\mathbf{x}_*$ such that $f(\mathbf{x}_*) = \min_{\mathbf{x} \in \mathbb{R}^d} f(\mathbf{x})$, and the function f be bounded from above and below by constants B_ℓ and B_u as $B_l \leq f(\mathbf{x}) \leq B_u$ for all $\mathbf{x} \in \mathbb{R}^d$. Then for any $\epsilon > 0$, $\exists T = O(\frac{1}{\epsilon^4})$ s.t the Algorithm 4 is guaranteed to reach a $t^{th}-$iterate s.t $1 \leq t \leq T$ and $\|\nabla f(\mathbf{x}_t)\| \leq \epsilon$.

3 Experimental setup

For testing the empirical performance of ADAM and RMSProp, we perform experiments on fully connected autoencoders using ReLU activations and shared weights. Let $\mathbf{z} \in \mathbb{R}^d$ be the input vector to the autoencoder, $\{W_i\}_{i=1,..,\ell}$ denote the weight matrices of the net and $\{\mathbf{b}_i\}_{i=1,..,2\ell}$ be the bias vectors. Then the output $\hat{\mathbf{z}} \in \mathbb{R}^d$ of the autoencoder is defined as $\hat{\mathbf{z}} = W_1^\top \sigma(\dots \sigma(W_{\ell-1}^\top \sigma(W_\ell^\top \mathbf{a} + \mathbf{b}_{\ell+1}) + \mathbf{b}_{\ell+2}) \dots) + \mathbf{b}_{2\ell}$ where $\mathbf{a} = \sigma(W_\ell \sigma(\dots \sigma(W_2 \sigma(W_1 \mathbf{z} + \mathbf{b}_1) + \mathbf{b}_2) \dots) + \mathbf{b}_\ell)$. This defines an autoencoder with $2\ell - 1$ hidden layers using ℓ weight matrices and 2ℓ bias vectors. Thus, the parameters of this model are given by $\mathbf{x} = [\text{vec}(W_1)^\top \dots \text{vec}(W_\ell)^\top \mathbf{b}_1^\top \dots \mathbf{b}_{2\ell}^\top]^\top$ (where we imagine all vectors to be column vectors by default). The loss function, for an input $\mathbf{z}$ is then given by: $f(\mathbf{z}; \mathbf{x}) = \|\mathbf{z} - \hat{\mathbf{z}}\|^2$.

Such autoencoders are a fairly standard setup that have been used in previous work [AZNG15, Bal12, KG17, VLL+10]. There have been relatively fewer comparisons of ADAM and RMSProp with other methods on a regression setting. We were motivated by [RMA+17] who had undertaken a theoretical analysis of autoencoders and in their experiments had found RMSProp to have good reconstruction error for MNIST when used on even just 2 layer ReLU autoencoders.

To keep our experiments as controlled as possible, we make all layers in a network have the same width (which we denote as h). Thus, given a size d for the input image, the weight matrices (as defined above) are given by: $W_1 \in \mathbb{R}^{h \times d}$, $W_i \in \mathbb{R}^{h \times h}, i = 2, \dots, \ell$. This allowed us to study the effect of increasing depth ℓ or width h without having to deal with added confounding factors. For all experiments, we use the standard "Glorot initialization" for the weights [GB10], where each element in the weight matrix is initialized by sampling from a uniform distribution with $[-\text{limit}, \text{limit}]$, limit $= \sqrt{6/(\text{fan}_{\text{in}} + \text{fan}_{\text{out}})}$, where fan_{in} denotes the number of input units in the weight matrix, and fan_{out} denotes the number of output units in the weight matrix. All bias vectors were initialized to zero. No regularization was used.

We performed autoencoder experiments on the MNIST dataset for various network sizes (i.e., different values of ℓ and h). We implemented all experiments using TensorFlow [ABC+16] using an NVIDIA GeForce GTX 1080 Ti graphics card. We compared the performance of ADAM and RMSProp with Nesterov's Accelerated Gradient (NAG). All experiments were run for 10^5 iterations. We tune over the hyper-parameters for each optimization algorithm using a grid search as described in subsection 3.1. To pick the best set of hyper-parameters, we choose the ones corresponding to the lowest loss on the training set at the end of 10^5 iterations. Further, to cut down on the computation time so that we can test a number of different neural net architectures, we crop the MNIST image from 28×28 down to a 22×22 image by removing 3 pixels from each side (almost all of which is whitespace).

Full-batch experiments We are interested in first comparing these algorithms in the full-batch setting. To do this in a computationally feasible way, we consider a subset of the MNIST dataset (we call this: mini-MNIST), which we build by extracting the first 5500 images in the training set and first 1000 images in the test set in MNIST. Thus, the training and testing datasets in mini-MNIST is 10% of the size of the MNIST dataset. Thus the training set in mini-MNIST contains 5500 images, while the test set contains 1000 images. This subset of the dataset is a fairly reasonable approximation of the full MNIST dataset (i.e., contains roughly the same distribution of labels as in the full MNIST dataset), and thus a legitimate dataset to optimize on.

Mini-batch experiments To test if our conclusions on the full-batch case extend to the mini-batch case, we then perform the same experiments in a mini-batch setup where we fix the mini-batch size at 100. For the mini-batch experiment, we consider the full training set of MNIST, instead of the mini-MNIST dataset considered for the full-batch experiments.

3.1 Hyperparameter Tuning

Here we describe how we tune the hyper-parameters of each optimization algorithm. NAG has two hyper-parameters, the step size α and the momentum μ. The main hyper-parameters for RMSProp are the step size α, the decay parameter β_2 and the perturbation ξ. ADAM, in addition to the ones in RMSProp, also has a momentum parameter β_1. We vary the step-sizes of ADAM in the conventional way of $\alpha_t = \alpha\sqrt{1-\beta_2^t}/(1-\beta_1^t)$.

For tuning the step size, we follow the same method used in [WRS+17]. We start out with a logarithmically-spaced grid of five step sizes. If the best performing parameter was at one of the extremes of the grid, we tried new grid points so that the best performing parameters were at one of the middle points in the grid. While it is computationally infeasible even with substantial resources to follow a similarly rigorous tuning process for all other hyper-parameters, we do tune over them somewhat as described below.

NAG The initial set of step sizes used for NAG were: $\{3\text{e}{-3}, 1\text{e}{-3}, 3\text{e}{-4}, 1\text{e}{-4}, 3\text{e}{-5}\}$. We tune the momentum parameter over values $\mu \in \{0.9, 0.99\}$.

RMSProp The initial set of step sizes used were: $\{3\text{e}{-4}, 1\text{e}{-4}, 3\text{e}{-5}, 1\text{e}{-5}, 3\text{e}{-6}\}$. We tune over $\beta_2 \in \{0.9, 0.99\}$. We set the perturbation value $\xi = 10^{-10}$, following the default values in TensorFlow, except for the experiments in Section 4.1. In Section 4.1, we show the effect on convergence and generalization properties of ADAM and RMSProp when changing this parameter ξ.

Note that ADAM and RMSProp uses an accumulator for keeping track of decayed squared gradient $\mathbf{v}_t$. For ADAM this is recommended to be initialized at $\mathbf{v}_0 = 0$. However, we found in the TensorFlow implementation of RMSProp that it sets $\mathbf{v}_0 = \mathbf{1}_d$. Instead of using this version of the algorithm, we used a modified version where we set $\mathbf{v}_0 = 0$. We typically found setting $\mathbf{v}_0 = 0$ to lead to faster convergence in our experiments.

ADAM The initial set of step sizes used were: $\{3\text{e}{-4}, 1\text{e}{-4}, 3\text{e}{-5}, 1\text{e}{-5}, 3\text{e}{-6}\}$. For ADAM, we tune over β_1 values of $\{0.9, 0.99\}$. For ADAM, We set $\beta_2 = 0.999$ for all our experiments as is set as the default in TensorFlow. Unless otherwise specified we use for the perturbation value $\xi = 10^{-8}$ for ADAM, following the default values in TensorFlow.

Contrary to what is the often used values of β_1 for ADAM (usually set to 0.9), we found that we often got better results on the autoencoder problem when setting $\beta_1 = 0.99$.

3.2 Pseudocodes for the experiments

Nesterov's Accelerated Gradient (NAG) Algorithm

Algorithm 1 TensorFlow's implementation of NAG

Input : A step size α, momentum $\mu \in [0,1)$, and an initial starting point $\mathbf{x}_1 \in \mathbb{R}^d$, and we are given oracle access to the gradients of $f : \mathbb{R}^d \to \mathbb{R}$.
function NAG($\mathbf{x}_1, \alpha, \mu$)
  Initialize : $\mathbf{v}_1 = \mathbf{0}$
  for $t = 1, 2, \ldots$ do
    $\mathbf{v}_{t+1} = \mu \mathbf{v}_t + \nabla f(\mathbf{x}_t)$
    $\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha(\nabla f(\mathbf{x}_t) + \mu \mathbf{v}_{t+1})$
  end for
  Output : x_{T+1}
end function

RMSProp Algorithm

Algorithm 2 RMSProp implementation used for experiments

Input : A constant vector $\mathbb{R}^d \ni \xi \mathbf{1}_d > 0$, parameter $\beta_2 \in [0,1)$, step size α, initial starting point $\mathbf{x}_1 \in \mathbb{R}^d$, and we are given oracle access to the gradients of $f : \mathbb{R}^d \to \mathbb{R}$.
function RMSProp($\mathbf{x}_1, \beta_2, \alpha, \xi$)
  Initialize : $\mathbf{v}_0 = \mathbf{0}$
  for $t = 1, 2, \ldots$ do
    $\mathbf{m}_t = \mathbf{g}_t = \nabla f(\mathbf{x}_t)$
    $\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2)\mathbf{g}_t^2$
    $V_t = \mathtt{diag}(\mathbf{v}_t + \xi \mathbf{1}_d)$
    $\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha V_t^{-\frac{1}{2}} \mathbf{m}_t$
  end for
  Output : x_{T+1}
end function

ADAM Algorithm

Algorithm 3 ADAM implementation used for experiments

Input : A constant vector $\mathbb{R}^d \ni \xi \mathbf{1}_d > 0$, parameters $\beta_1, \beta_2 \in [0,1)$, a sequence of step sizes $\{\alpha_t\}_{t=1,2..}$, initial starting point $\mathbf{x}_1 \in \mathbb{R}^d$, and we are given oracle access to the gradients of $f : \mathbb{R}^d \to \mathbb{R}$.
function ADAM($\mathbf{x}_1, \beta_1, \beta_2, \alpha, \xi$)
  Initialize : $\mathbf{m}_0 = \mathbf{0}$, $\mathbf{v}_0 = \mathbf{0}$
  for $t = 1, 2, \ldots$ do
    $\mathbf{g}_t = \nabla f(\mathbf{x}_t)$
    $\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1)\mathbf{g}_t$
    $\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2)\mathbf{g}_t^2$
    $V_t = \mathtt{diag}(\mathbf{v}_t)$
    $\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha_t \Big(V_t^{\frac{1}{2}} + \mathtt{diag}(\xi \mathbf{1}_d) \Big)^{-1} \mathbf{m}_t$
  end for
  Output : x_{T+1}
end function

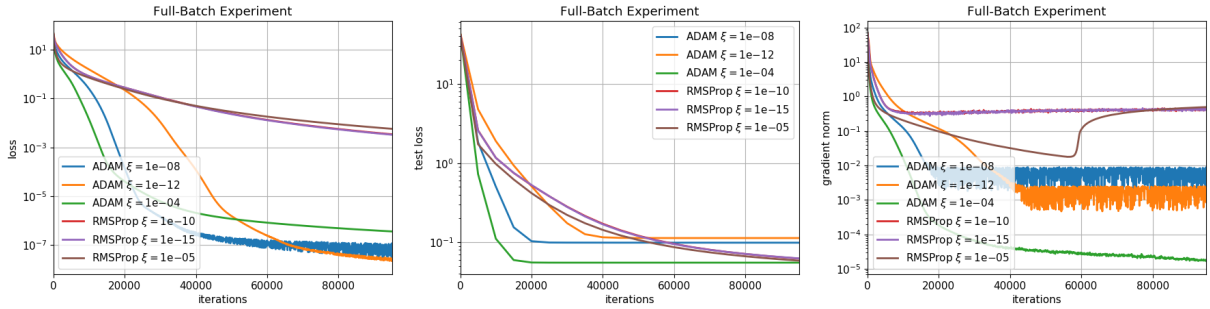


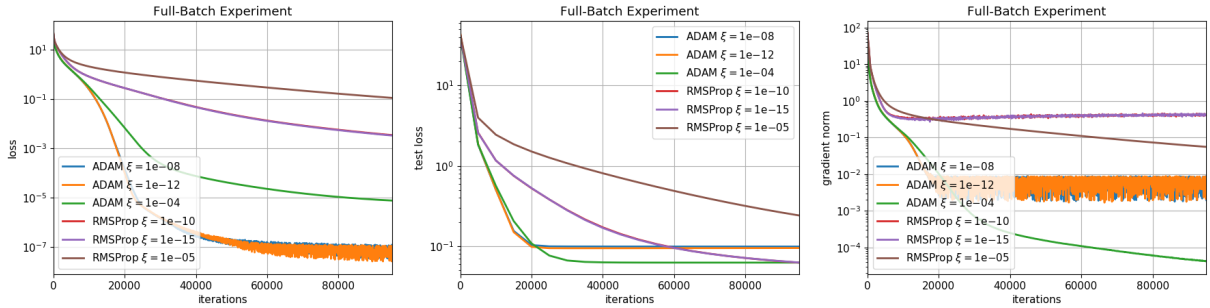
Figure 1: Optimally tuned parameters for different ξ values. 1 hidden layer network of 1000 nodes; *Left*: Loss on training set; *Middle*: Loss on test set; *Right*: Gradient norm on training set

4 Experimental Results

4.1 RMSProp and ADAM are sensitive to choice of ξ

The ξ parameter is a feature of the default implementations of RMSProp and ADAM such as in TensorFlow. Most interestingly this strictly positive parameter is crucial for our proofs. In this section we present experimental evidence that attempts to clarify that this isn't merely a theoretical artefact but its value indeed has visible effect on the behaviours of these algorithms. We see in Figure 1 that on increasing the value of this fixed shift parameter ξ, ADAM in particular, is strongly helped towards getting lower gradient norms and lower test losses though it can hurt its ability to get lower training losses. The plots are shown for optimally tuned values for the other hyper-parameters.

Next, in Figure 2, we show the effect of changing ξ on a 1 hidden layer network of 1000 nodes, while keeping all other hyper-parameters fixed (such as learning rate, β_1, β_2). These other hyper-parameter values were fixed at the best values of these parameters for the default values of ξ, i.e., $\xi = 10^{-10}$ for RMSProp and $\xi = 10^{-8}$ for ADAM.



(a) Loss on training set (b) Loss on test set (c) Gradient norm on training set

Figure 2: Fixed parameters with changing ξ values. 1 hidden layer network of 1000 nodes

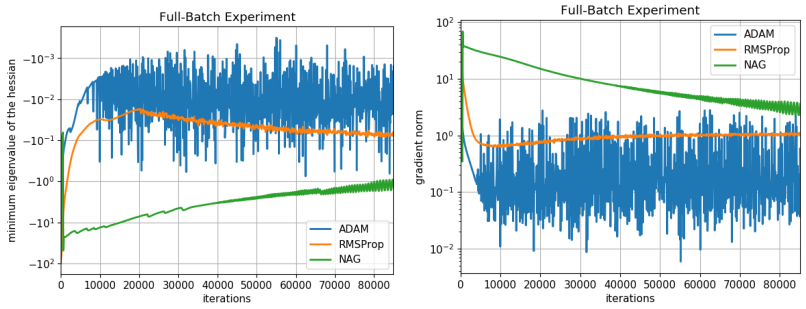
4.2 Experimental evidence of NAG's ability with λ_{min}(Hessian) of the loss function

To check whether NAG, ADAM or RMSProp is capable of consistently moving from a "bad" saddle point to a "good" saddle point region, we track the most negative eigenvalue of the Hessian $\lambda_{\min}(Hessian)$.

Here we show plots to demonstrate that on autoencoders with one layer of activations, NAG seems to have a distinctive ability to raise the $\lambda_{min}(\text{Hessian})$ of the loss function while consistently lowering the gradient norms.

Even for a very small neural network with around 10^5 parameters, it is still intractable to store the full Hessian matrix in memory to compute the eigenvalues. Instead, we use the Scipy library function `scipy.sparse.linalg.eigsh` that can use a function that computes the matrix-vector products to compute the eigenvalues of the matrix [LSY98]. Thus, for finding the eigenvalues of the Hessian, it is sufficient to be able to do Hessian-vector products instead of computing the full Hessian. This can be done exactly in a fairly efficient way [Tow08].

We display a representative plot in Figure 3 which shows that NAG in particular has a distinct ability to gradually, but consistently, keep increasing the minimum eigenvalue of the Hessian while continuing to decrease the gradient norm. In contrast, RSMProp and ADAM quickly get to a high value of the minimum eigenvalue and a small gradient norm, but somewhat stagnate there. However, the gradient norms are consistently bigger and the minimum eigenvalues are consistently smaller for NAG, compared to RMSProp and ADAM. In short, the trend looks better for NAG, but in actual numbers RMSProp and ADAM do better.



(a) Minimum Hessian eigenvalue (b) Gradient norm on training set

Figure 3: Tracking the smallest eigenvalue of the Hessian on a 1 hidden layer network of size 300

4.3 Comparing performance in the full-batch setting

In Figure 4, we show how the training loss, test loss and gradient norms vary through the iterations for RMSProp, ADAM (at $\beta_1 = 0.9$ and 0.99) and NAG (at $\mu = 0.9$ and 0.99) on a 3 hidden layer autoencoder with 1000 nodes in each hidden layer trained on mini-MNIST. Appendix A.1 and A.2 have more such comparisons for various neural net architectures with varying depth and width and input image sizes, where the following qualitative results also extend.

We conclude the following from the full-batch experiments of training autoencoders on mini-MNIST:

- Pushing β_1 closer to 1 significantly helps ADAM in getting lower training and test losses and at these values of β_1, it has better performance on these metrics than all the other algorithms. One sees cases like the one displayed in Figure 4 where ADAM at $\beta_1 = 0.9$ was getting comparable or slightly worse test and training errors than NAG. But once β_1 gets closer to 1, ADAM's performance sharply improves and gets better than other algorithms.
- Increasing momentum μ helps NAG get lower gradient norms though on larger nets it might hurt its training or test performance. NAG does seem to get the lowest gradient norms compared to the other algorithms, except for single hidden layer networks like in Figure 3.

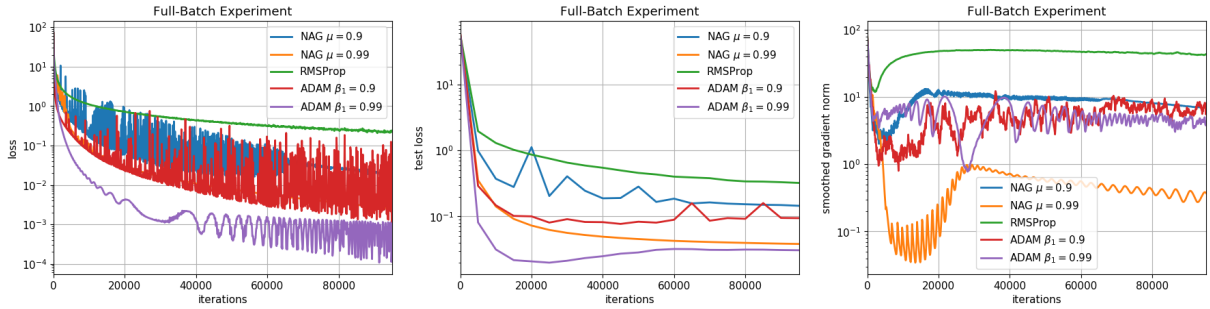


Figure 4: Full-batch experiments on a 3 hidden layer network with 1000 nodes in each layer; *Left*: Loss on training set; *Middle*: Loss on test set; *Right*: Gradient norm on training set

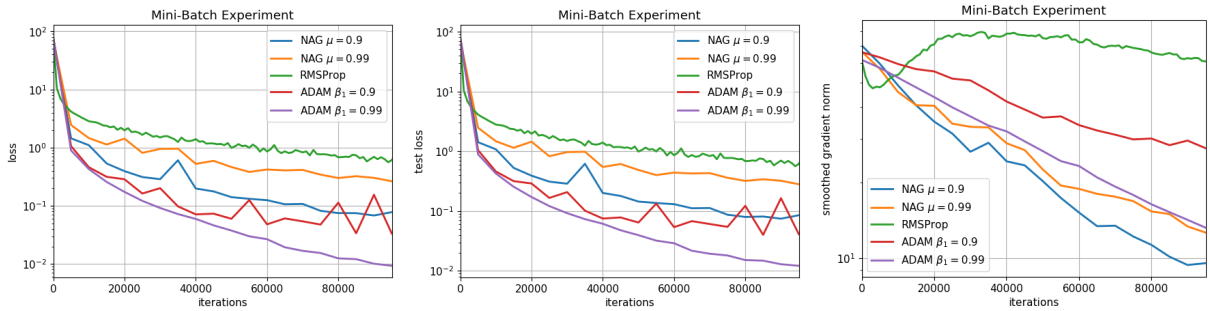


Figure 5: Mini-batch experiments on a network with 5 hidden layers of 1000 nodes each; *Left*: Loss on training set; *Middle*: Loss on test set; *Right*: Gradient norm on training set

4.4 Corroborating the full-batch behaviors of the algorithms in the mini-batch setting

In Figure 5, we show how training loss, test loss and gradient norms vary when using mini-batches of size 100, on a 5 hidden layer autoencoder with 1000 nodes in each hidden layer trained on the full MNIST dataset. More such mini-batch comparisons on deeper and shallower autoencoder architectures with varying widths can be seen in Appendix A.3, showing that qualitatively the same pattern follows.

We conclude the following from the mini-batch experiments of training autoencoders on full MNIST dataset:

- Mini-batching does seem to help NAG do better than ADAM on small nets. However, for larger nets, the full-batch behavior continues, i.e., when ADAM's momentum parameter β_1 is pushed closer to 1, it gets better generalization (significantly lower test losses) than NAG at any momentum tested.
- In general, for all metrics (test loss, training loss and gradient norm reduction) both ADAM as well as NAG seem to improve in performance when their momentum parameter (μ for NAG and β_1 for ADAM) is pushed closer to 1. This effect, which was present in the full-batch setting, seems to get more pronounced here.
- As in the full-batch experiments, NAG continues to have the best ability to reduce gradient norms while for larger enough nets, ADAM at large momentum continues to have the best training error.

5 Proofs of convergence of ADAM and RMSProp when using exact gradients

5.1 Proving ADAM (Proof of Theorem 2.1)

Proof.

Let us assume to the contrary that $\|g_t\| > \epsilon$ for all $t = 1, 2, 3\ldots.$ We will show that this assumption will lead to a contradiction. By $L-$smoothness of the objective we have the following relationship between the values at consecutive updates,

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) + \langle \nabla f(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \rangle + \frac{L}{2} \|\mathbf{x}_{t+1} - \mathbf{x}_t\|^2$$

Substituting the update rule using a dummy step length $\eta_t > 0$ we have,

$$\begin{aligned} & f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \eta_t \langle \nabla f(\mathbf{x}_t), \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \rangle + \frac{L\eta_t^2}{2} \left\| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \right\|^2 \\ \implies & f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t) \leq \eta_t \left(-\langle \mathbf{g}_t, \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \rangle + \frac{L\eta_t}{2} \left\| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \right\|^2 \right) \end{aligned} \tag{1}$$

The RHS in equation 1 above is a quadratic in η_t with two roots: 0 and $\frac{\langle \mathbf{g}_t, \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \rangle}{\frac{L}{2} \left\| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \right\|^2}$. So the quadratic's minimum value is at the midpoint of this interval, which gives us a candidate $t^{th}-$step length i.e $\alpha_t^* := \frac{1}{2} \cdot \frac{\langle \mathbf{g}_t, \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \rangle}{\frac{L}{2} \left\| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \right\|^2}$ and the value of the quadratic at this point is $-\frac{1}{4} \cdot \frac{(\langle \mathbf{g}_t, \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \rangle)^2}{\frac{L}{2} \left\| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \right\|^2}$. That is with step lengths being this α_t^* we have the following guarantee of decrease of function value between consecutive steps,

$$f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t) \leq -\frac{1}{2L} \cdot \frac{(\langle \mathbf{g}_t, \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \rangle)^2}{\left\| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \right\|^2} \tag{2}$$

Now we separately lower bound the numerator and upper bound the denominator of the RHS above.

Upperbound on $\left\| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \right\|$

We have, $\lambda_{max}\left(\left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1}\right) \leq \frac{1}{\xi + \min_{i=1..d} \sqrt{(\mathbf{v}_t)_i}}$ Further we note that the recursion of $\mathbf{v}_t$ can be solved as, $\mathbf{v}_t = (1-\beta_2) \sum_{k=1}^{t} \beta_2^{t-k} \mathbf{g}_k^2$. Now we define, $\epsilon_t := \min_{k=1,..,t, i=1,..,d} (\mathbf{g}_k^2)_i$ and this gives us,

$$\lambda_{max}\Big(\Big(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\Big)^{-1}\Big) \leq \frac{1}{\xi + \sqrt{(1-\beta_2^t)\epsilon_t}} \tag{3}$$

We solve the recursion for $\mathbf{m}_t$ to get, $\mathbf{m}_t = (1-\beta_1)\sum_{k=1}^{t} \beta_1^{t-k}\mathbf{g}_k$. Then by triangle inequality and defining $\sigma_t := \max_{i=1,..,t} \|\nabla f(\mathbf{x}_i)\|$ we have, $\|\mathbf{m}_t\| \leq (1-\beta_1^t)\sigma_t$. Thus combining this estimate of $\|\mathbf{m}_t\|$ with equation 3 we have,

$$\left\|\Big(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\Big)^{-1}\mathbf{m}_t\right\| \leq \frac{(1-\beta_1^t)\sigma_t}{\xi + \sqrt{\epsilon_t(1-\beta_2^t)}} \leq \frac{(1-\beta_1^t)\sigma_t}{\xi} \tag{4}$$

Lowerbound on $\langle \mathbf{g}_t, \Big(V_t^{\frac{1}{2}} + \textbf{diag}(\xi \mathbf{1}_d)\Big)^{-1}\mathbf{m}_t\rangle$
To analyze this we define the following sequence of functions for each $i = 0, 1, 2.., t$

$$Q_i = \langle \mathbf{g}_t, \Big(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\Big)^{-1}\mathbf{m}_i\rangle$$

This gives us the following on substituting the update rule for $\mathbf{m}_t$,

$$\begin{aligned} Q_i - \beta_1 Q_{i-1} &= \langle \mathbf{g}_t, \Big(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\Big)^{-1}(\mathbf{m}_i - \beta_1 \mathbf{m}_{i-1})\rangle \\ &= (1-\beta_1)\langle \mathbf{g}_t, \Big(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\Big)^{-1}\mathbf{g}_i\rangle \end{aligned}$$

At $i = t$ we have,

$$Q_t - \beta_1 Q_{t-1} \geq (1-\beta_1)\|\mathbf{g}_t\|^2 \lambda_{min}\Big(\Big(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\Big)^{-1}\Big)$$

Lets define, $\sigma_{t-1} := \max_{i=1,..,t-1} \|\nabla f(\mathbf{x}_i)\|$ and this gives us for $i \in \{1, .., t-1\}$,

$$Q_i - \beta_1 Q_{i-1} \geq -(1-\beta_1)\|\mathbf{g}_t\|\sigma_{t-1}\lambda_{max}\Big(\Big(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\Big)^{-1}\Big)$$

We note the following identity,

$$Q_t - \beta_1^t Q_0 = (Q_t - \beta_1 Q_{t-1}) + \beta_1(Q_{t-1} - \beta_1 Q_{t-2}) + \beta_1^2(Q_{t-2} - \beta_1 Q_{t-3}) + .. + \beta_1^{t-1}(Q_1 - \beta_1 Q_0)$$

Now we use the lowerbounds proven on $Q_i - \beta_1 Q_{i-1}$ for $i \in \{1, .., t-1\}$ and $Q_t - \beta_1 Q_{t-1}$ to lowerbound the above sum as,

$$
\begin{aligned}
Q_t - \beta_1^t Q_0 &\geq (1-\beta_1)\left\|\mathbf{g}_t\right\|^2 \lambda_{min}\left(\left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1}\right) \\
&\quad - (1-\beta_1)\left\|\mathbf{g}_t\right\| \sigma_{t-1} \lambda_{max}\left(\left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1}\right) \sum_{j=1}^{t-1} \beta_1^j \\
&\geq (1-\beta_1)\left\|\mathbf{g}_t\right\|^2 \lambda_{min}\left(\left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1}\right) \\
&\quad - (\beta_1 - \beta_1^t)\left\|\mathbf{g}_t\right\| \sigma_{t-1} \lambda_{max}\left(\left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1}\right) \qquad (5)
\end{aligned}
$$

We can evaluate the following lowerbound, $\lambda_{min}\left(\left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1}\right) \geq \frac{1}{\xi + \sqrt{\max_{i=1,..,d}(\mathbf{v}_t)_i}}$. Next we remember that the recursion of $\mathbf{v}_t$ can be solved as, $\mathbf{v}_t = (1-\beta_2)\sum_{k=1}^{t} \beta_2^{t-k} \mathbf{g}_k^2$ and we define, $\sigma_t := \max_{i=1,..,t} \left\|\nabla f(\mathbf{x}_i)\right\|$ to get,

$$
\lambda_{min}\left(\left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1}\right) \geq \frac{1}{\xi + \sqrt{(1-\beta_2^t)\sigma_t^2}} \qquad (6)
$$

Now we combine the above and equation 3 and the known value of $Q_0 = 0$ (from definition and initial conditions) to get from the equation 5,

$$
\begin{aligned}
Q_t &\geq -(\beta_1 - \beta_1^t)\left\|\mathbf{g}_t\right\| \sigma_{t-1} \frac{1}{\xi + \sqrt{(1-\beta_2^t)\epsilon_t}} \\
&\quad + (1-\beta_1)\left\|\mathbf{g}_t\right\|^2 \frac{1}{\xi + \sqrt{(1-\beta_2^t)\sigma_t^2}} \\
&\geq \left\|\mathbf{g}_t\right\|^2 \left(\frac{(1-\beta_1)}{\xi + \sigma\sqrt{(1-\beta_2^t)}} - \frac{(\beta_1 - \beta_1^t)\sigma}{\xi \left\|\mathbf{g}_t\right\|}\right) \qquad (7)
\end{aligned}
$$

In the above inequalities we have set $\epsilon_t = 0$ and we have set, $\sigma_t = \sigma_{t-1} = \sigma$. Now we examine the following part of the lowerbound proven above,

$$
\begin{aligned}
\frac{(1-\beta_1)}{\xi + \sqrt{(1-\beta_2^t)\sigma^2}} - \frac{(\beta_1 - \beta_1^t)\sigma}{\xi \left\|\mathbf{g}_t\right\|} &= \frac{\xi \left\|\mathbf{g}_t\right\| (1-\beta_1) - \sigma(\beta_1 - \beta_1^t)(\xi + \sigma\sqrt{(1-\beta_2^t)})}{\xi \left\|\mathbf{g}_t\right\| (\xi + \sigma\sqrt{(1-\beta_2^t)})} \\
&= \sigma(\beta_1 - \beta_1^t) \frac{\xi\left(\frac{\left\|\mathbf{g}_t\right\|(1-\beta_1)}{\sigma(\beta_1 - \beta_1^t)} - 1\right) - \sigma\sqrt{(1-\beta_2^t)}}{\xi \left\|\mathbf{g}_t\right\| (\xi + \sigma\sqrt{(1-\beta_2^t)})} \\
&= \sigma(\beta_1 - \beta_1^t)\left(\frac{\left\|\mathbf{g}_t\right\|(1-\beta_1)}{\sigma(\beta_1 - \beta_1^t)} - 1\right) \frac{\xi - \left(\frac{\sigma\sqrt{(1-\beta_2^t)}}{-1 + \frac{(1-\beta_1)\left\|\mathbf{g}_t\right\|}{(\beta_1 - \beta_1^t)\sigma}}\right)}{\xi \left\|\mathbf{g}_t\right\| (\xi + \sigma\sqrt{(1-\beta_2^t)})}
\end{aligned}
$$

Now we remember the assumption that we are working under i.e $\|g_t\| > \epsilon$. Also by definition $0 < \beta_1 < 1$ and hence we have $0 < \beta_1 - \beta_1^t < \beta_1$. This implies, $\frac{(1-\beta_1)\left\|\mathbf{g}_t\right\|}{(\beta_1 - \beta_1^t)\sigma} > \frac{(1-\beta_1)\epsilon}{\beta_1 \sigma} > 1$ where the last inequality

follows because of our choice of ϵ as stated in the theorem statement. This allows us to define a constant, $\frac{\epsilon(1-\beta_1)}{\beta_1\sigma} - 1 := \theta_1 > 0$ s.t $\frac{(1-\beta_1)\|\mathbf{g}_t\|}{(\beta_1-\beta_1^t)\sigma} - 1 > \theta_1$

Similarly our definition of ξ allows us to define a constant $\theta_2 > 0$ to get,

$$\left(\frac{\sigma\sqrt{(1-\beta_2^t)}}{-1+\frac{(1-\beta_1)\|\mathbf{g}_t\|}{(\beta_1-\beta_1^t)\sigma}}\right) < \frac{\sigma}{\theta_1} = \xi - \theta_2$$

Putting the above back into the lowerbound for Q_t in equation 7 we have,

$$Q_t \geq \|\mathbf{g}_t\|^2 \left(\frac{\sigma(\beta_1-\beta_1^2)\theta_1\theta_2}{\xi\sigma(\xi+\sigma)}\right) \tag{8}$$

Now we substitute the above and equation 4 into equation 2 to get,

$$\begin{aligned}
f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t) &\leq -\frac{1}{2L} \cdot \frac{(\langle \mathbf{g}_t, \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t\rangle)^2}{\left\|\left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t\right\|^2} \\
&\leq -\frac{1}{2L} \frac{Q_t^2}{\left\|\left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t\right\|^2} \\
&\leq -\frac{1}{2L} \frac{\|\mathbf{g}_t\|^4 \left(\frac{(\beta_1-\beta_1^2)\theta_1\theta_2}{\xi(\xi+\sigma)}\right)^2}{\left(\frac{(1-\beta_1^t)\sigma}{\xi}\right)^2} \\
&\leq -\frac{\|\mathbf{g}_t\|^4}{2L} \left(\frac{(\beta_1-\beta_1^2)^2\theta_1^2\theta_2^2}{(\xi+\sigma)^2(1-\beta_1^t)^2\sigma^2}\right)
\end{aligned} \tag{9}$$

$$\begin{aligned}
&\left(\frac{(\beta_1-\beta_1^2)^2\theta_1^2\theta_2^2}{2L(\xi+\sigma)^2(1-\beta_1^t)^2\sigma^2}\right)\|\nabla f(\mathbf{x}_t)\|^4 \leq [f(\mathbf{x}_t) - f(\mathbf{x}_{t+1})] \\
\implies &\sum_{t=2}^{T}\left(\frac{(\beta_1-\beta_1^2)^2\theta_1^2\theta_2^2}{2L(\xi+\sigma)^2\sigma^2}\right)\|\nabla f(\mathbf{x}_t)\|^4 \leq [f(\mathbf{x}_2) - f(\mathbf{x}_{T+1})] \\
\implies &\min_{t=2,..,T}\|\nabla f(\mathbf{x}_t)\|^4 \leq \frac{2L(\xi+\sigma)^2\sigma^2}{T(\beta_1-\beta_1^2)^2\theta_1^2\theta_2^2}[f(\mathbf{x}_2) - f(\mathbf{x}_*)]
\end{aligned}$$

So it follows from the above equation that we are assured to find a $\mathbf{x}_{result} := \arg\min_{t=2,..,T}\|\nabla f(\mathbf{x}_t)\|^2$ s.t $\|\nabla f(\mathbf{x}_{result})\| \leq \epsilon$ when the number of steps T is,

$$T \geq \frac{2L\sigma^2(\xi+\sigma)^2}{2\epsilon^4(\beta_1-\beta_1^2)^2\theta_1^2\theta_2^2}[f(\mathbf{x}_2) - f(\mathbf{x}_*)]$$

Thus we have proven the absurdity of the initial assumption that this ADAM never finds a point where the gradient norm is at or below ϵ. As a consequence we have proven the first part of the theorem which

guarantees the existence of positive step lengths, α_t s.t ADAM finds an approximately critical point in finite time.

Now choose $\theta_1 = 1$ i.e $\frac{\epsilon}{2} = \frac{\beta_1 \sigma}{1-\beta_1}$ i.e $\beta_1 = \frac{\epsilon}{\epsilon+2\sigma} \implies \beta_1(1-\beta_1) = \frac{\epsilon}{\epsilon+2\sigma}(1 - \frac{\epsilon}{\epsilon+2\sigma}) = \frac{2\sigma\epsilon}{(\epsilon+2\sigma)^2}$. This also gives a easier-to-read condition on ξ in terms of these parameters i.e $\xi > \sigma$. Now choose $\xi = 2\sigma$ i.e $\theta_2 = \sigma$ and making these substitutions gives us,

$$T \geq \frac{18L\sigma^4}{2\epsilon^4 \left(\frac{2\sigma\epsilon}{(\epsilon+2\sigma)}\right)^2 \sigma^2}[f(\mathbf{x}_2) - f(\mathbf{x}_*)] \geq \frac{18L}{8\epsilon^6 \left(\frac{1}{\epsilon+2\sigma}\right)^2}[f(\mathbf{x}_2) - f(\mathbf{x}_*)] \geq \frac{9L\sigma^2}{\epsilon^6}[f(\mathbf{x}_2) - f(\mathbf{x}_*)]$$

We substitute these choices in the step length found earlier to get,

$$\begin{aligned}
\alpha_t^* &= \frac{1}{L} \cdot \frac{\langle \mathbf{g}_t, \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \rangle}{\left\| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \right\|^2} = \frac{1}{L} \cdot \frac{Q_t}{\left\| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \right\|^2} \\
&\geq \frac{1}{L} \frac{\|\mathbf{g}_t\|^2 \left(\frac{\sigma^2(\beta_1 - \beta_1^2)}{\xi\sigma(\xi+\sigma)}\right)}{\left(\frac{(1-\beta_1^t)\sigma}{\xi}\right)^2} = \frac{\|\mathbf{g}_t\|^2}{L(1-\beta_1^t)^2} \frac{4\epsilon}{3(\epsilon+2\sigma)^2} := \alpha_t
\end{aligned}$$

In the theorem statement we choose to call as the final α_t the lowerbound proven above and thus we have the statement as claimed in the theorem. We check below that this smaller value of α_t still guarantees a decrease in the function value that is sufficient for the statement of the theorem to hold.

A consistency check! Let us substitute the above final value of the step length $\alpha_t = \frac{1}{L} \frac{\|\mathbf{g}_t\|^2 \left(\frac{\sigma^2(\beta_1 - \beta_1^2)}{\xi\sigma(\xi+\sigma)}\right)}{\left(\frac{(1-\beta_1^t)\sigma}{\xi}\right)^2} = \frac{\xi}{L(1-\beta_1^t)^2} \|\mathbf{g}_t\|^2 \left(\frac{(\beta_1 - \beta_1^2)}{\sigma(\xi+\sigma)}\right)$, the bound in equation 4 (with σ_t replaced by σ), and the bound in equation 8 (at the chosen values of $\theta_1 = 1$ and $\theta_2 = \sigma$) in the original equation 2 to measure the decrease in the function value between consecutive steps,

$$\begin{aligned}
f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t) &\leq \alpha_t \left(-\langle \mathbf{g}_t, \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \rangle + \frac{L\alpha_t}{2} \left\| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \right\|^2 \right) \\
&\leq \alpha_t \left(-Q_t + \frac{L\alpha_t}{2} \left\| \left(V_t^{\frac{1}{2}} + \text{diag}(\xi \mathbf{1}_d)\right)^{-1} \mathbf{m}_t \right\|^2 \right) \\
&\leq \frac{\xi}{L(1-\beta_1^t)^2} \|\mathbf{g}_t\|^2 \left(\frac{(\beta_1 - \beta_1^2)}{\sigma(\xi+\sigma)}\right) \left(- \|\mathbf{g}_t\|^2 \left(\frac{\sigma(\beta_1 - \beta_1^2)\theta_1\theta_2}{\xi\sigma(\xi+\sigma)}\right)\right) \\
&+ \frac{L}{2} \left(\frac{\xi}{L(1-\beta_1^t)^2} \|\mathbf{g}_t\|^2 \left(\frac{(\beta_1 - \beta_1^2)}{\sigma(\xi+\sigma)}\right) \frac{(1-\beta_1^t)\sigma}{\xi} \right)^2
\end{aligned}$$

The RHS above can be simplified to be shown to be equal to the RHS in equation 9 at the same values of θ_1 and θ_2 as used above. And we remember that the bound on the running time was derived from this equation 9. □

5.2 Proving RMSProp - the version with standard speed (Proof of Theorem 2.2)

Proof. By the $L-$smoothness condition and the update rule in Algorithm 2 we have,

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \alpha_t \langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \rangle + \alpha_t^2 \frac{L}{2} \| V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \|^2$$
$$\implies f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t) \leq \alpha_t \left(\frac{L\alpha_t}{2} \| V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \|^2 - \langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \rangle \right) \tag{10}$$

For $0 < \delta_t^2 < \frac{1}{\sqrt{1-\beta_2^t}\sqrt{\sigma_t^2+\xi}}$ we would now show a strictly positive lowerbound on the following function,

$$\frac{2}{L} \left(\frac{\langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \rangle - \delta_t^2 \| \nabla f(\mathbf{x}_t) \|^2}{\| V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \|^2} \right) \tag{11}$$

We define $\sigma_t := \max_{i=1,..,t} \| \nabla f(\mathbf{x}_i) \|$ and we solve the recursion for $\mathbf{v}_t$ as, $\mathbf{v}_t = (1-\beta_2) \sum_{k=1}^t \beta_2^{t-k} (\mathbf{g}_k^2 + \xi)$. This lets us write the following bounds,

$$\begin{aligned} \langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \rangle &\geq \lambda_{min}(V_t^{-\frac{1}{2}}) \| \nabla f(\mathbf{x}_t) \|^2 \geq \frac{\| \nabla f(\mathbf{x}_t) \|^2}{\sqrt{\max_{i=1,..,d}(\mathbf{v}_t)_i}} \\ &\geq \frac{\| \nabla f(\mathbf{x}_t) \|^2}{\sqrt{\max_{i=1,..,d}((1-\beta_2) \sum_{k=1}^t \beta_2^{t-k} (\mathbf{g}_k^2 + \xi \mathbf{1}_d)_i)}} \\ &\geq \frac{\| \nabla f(\mathbf{x}_t) \|^2}{\sqrt{1-\beta_2^t}\sqrt{\sigma_t^2+\xi}} \end{aligned} \tag{12}$$

Now we define, $\epsilon_t := \min_{k=1,..,t, i=1,..,d} (\nabla f(\mathbf{x}_k))_i^2$ and this lets us get the following sequence of inequalities,

$$\| V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \|^2 \leq \lambda_{max}^2 (V_t^{-\frac{1}{2}}) \| \nabla f(\mathbf{x}_t) \|^2 \leq \frac{\| \nabla f(\mathbf{x}_t) \|^2}{(\min_{i=1,..,d}(\sqrt{(\mathbf{v}_t)_i}))^2} \leq \frac{\| \nabla f(\mathbf{x}_t) \|^2}{(1-\beta_2^t)(\xi+\epsilon_t)} \tag{13}$$

So combining equations 13 and 12 into equation 11 and from the exit line in the loop we are assured that $\| \nabla f(\mathbf{x}_t) \|^2 \neq 0$ and combining these we have,

$$\begin{aligned} \frac{2}{L} \left(\frac{-\delta_t^2 \| \nabla f(\mathbf{x}_t) \|^2 + \langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \rangle}{\| V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \|^2} \right) &\geq \frac{2}{L} \left(\frac{-\delta_t^2 + \frac{1}{\sqrt{1-\beta_2^t}\sqrt{\sigma_t^2+\xi}}}{\frac{1}{(1-\beta_2^t)(\xi+\epsilon_t)}} \right) \\ &\geq \frac{2(1-\beta_2^t)(\xi+\epsilon_t)}{L} \left(-\delta_t^2 + \frac{1}{\sqrt{1-\beta_2^t}\sqrt{\sigma_t^2+\xi}} \right) \end{aligned}$$

Now our definition of δ_t^2 allows us to define a parameter $0 < \beta_t := \frac{1}{\sqrt{1-\beta_2^t}\sqrt{\sigma_t^2+\xi}} - \delta_t^2$ and rewrite the above equation as,

$$\frac{2}{L} \left(\frac{-\delta_t^2 \| \nabla f(\mathbf{x}_t) \|^2 + \langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \rangle}{\| V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \|^2} \right) \geq \frac{2(1-\beta_2^t)(\xi+\epsilon_t)\beta_t}{L} \tag{14}$$

This now allows us to see that a step length $\alpha_t > 0$ for the $t^{th}-$step can be defined as, $\alpha_t = \frac{2(1-\beta_2^t)(\xi+\epsilon_t)\beta_t}{L}$ and this is such that the above equation can be written as, $\frac{L\alpha_t}{2}\|V_t^{-\frac{1}{2}}\nabla f(\mathbf{x}_t)\|^2 - \langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}}\nabla f(\mathbf{x}_t)\rangle \leq -\delta_t^2 \|\nabla f(\mathbf{x}_t)\|^2$. This when substituted back into equation 10 we have,

$$f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t) \leq -\delta_t^2 \alpha_t \|\nabla f(\mathbf{x}_t)\|^2 = -\frac{2\delta_t^2(1-\beta_2^t)(\xi+\epsilon_t)\beta_t}{L}\|\nabla f(\mathbf{x}_t)\|^2$$

This gives us,

$$\begin{aligned} \|\nabla f(\mathbf{x}_t)\|^2 &\leq \frac{L}{2\delta_t^2(1-\beta_2^t)(\xi+\epsilon_t)\beta_t}[f(\mathbf{x}_t) - f(\mathbf{x}_{t+1})] \\ \implies \sum_{t=1}^{T}\|\nabla f(\mathbf{x}_t)\|^2 &\leq \sum_{t=1}^{T}\frac{L}{2\delta_t^2(1-\beta_2^t)(\xi+\epsilon_t)\beta_t}[f(\mathbf{x}_t) - f(\mathbf{x}_{t+1})] \end{aligned} \tag{15}$$

Now we note that with $0 < \delta_t^2 < \frac{1}{\sqrt{1-\beta_2^t}\sqrt{\sigma_t^2+\xi}}$ we can always choose a constant value of δ_t^2 parameterized by a constant $\gamma^2 > 0$ s.t , $\delta_t^2 + \gamma^2 = \min_{t=1,..,T}\frac{1}{\sqrt{1-\beta_2^t}\sqrt{\sigma_t^2+\xi}} = \frac{1}{\sqrt{1-\beta_2^T}\sqrt{\sigma^2+\xi}}$. Previously we had defined $\beta_t = \frac{1}{\sqrt{1-\beta_2^t}\sqrt{\sigma_t^2+\xi}} - \delta_t^2$ and that now implies $\beta_t > \gamma^2$ Thus substituting these choices into equation 15 and minimizing the denominator of the RHS we have,

$$\sum_{t=1}^{T}\|\nabla f(\mathbf{x}_t)\|^2 \leq \frac{L}{2\xi(1-\beta_2)(\frac{1}{\sqrt{1-\beta_2^T}\sqrt{\sigma^2+\xi}} - \gamma^2)\gamma^2}[f(\mathbf{x}_1) - f(\mathbf{x}_*)]$$

We choose the smallest possible RHS by choosing, $\gamma^2 = \frac{1}{2\sqrt{1-\beta_2^T}\sqrt{\sigma^2+\xi}}$ and for this the above equation becomes,

$$\begin{aligned} \sum_{t=1}^{T}\|\nabla f(\mathbf{x}_t)\|^2 &\leq \frac{4L(\sigma^2+\xi)(1-\beta_2^T)}{2\xi(1-\beta_2)}[f(\mathbf{x}_1) - f(\mathbf{x}_*)] \\ \implies \min_{t=1,,.T}\|\nabla f(\mathbf{x}_t)\|^2 &\leq \frac{2L(\sigma^2+\xi)(1-\beta_2^T)}{T\xi(1-\beta_2)}[f(\mathbf{x}_1) - f(\mathbf{x}_*)] \end{aligned}$$

Thus for any given $\epsilon > 0$, T satisfying, $\frac{2L(\sigma^2+\xi)(1-\beta_2^T)}{T\xi(1-\beta_2)}[f(\mathbf{x}_1) - f(\mathbf{x}_*)] \leq \epsilon^2$ is a sufficient condition to ensure that the algorithm finds a point $\mathbf{x}_{result} := \arg\min_{t=1,,.T}\|\nabla f(\mathbf{x}_t)\|^2$ with $\|\nabla f(\mathbf{x}_{result})\|^2 \leq \epsilon^2$. Using $\beta_2^T < \beta_2$, this condition on T, $\frac{1-\beta_2^T}{T} \leq \frac{\epsilon^2\xi(1-\beta_2)}{2L(\sigma^2+\xi)[f(\mathbf{x}_1)-f(\mathbf{x}_*)]}$ can be written as, $T \geq \frac{2L[f(\mathbf{x}_1)-f(\mathbf{x}_*)]\left(1+\frac{\sigma^2}{\xi}\right)}{\epsilon^2}$

Further we make a specific choice of $\gamma^2 = \frac{1}{2\sqrt{1-\beta_2^T}\sqrt{\sigma^2+\xi}}$ (and consequentially a value gets fixed for δ_t^2 since we had defined $\delta_t^2 + \gamma^2 = \frac{1}{\sqrt{1-\beta_2^T}\sqrt{\sigma^2+\xi}}$ and also of $\beta_t = \frac{1}{\sqrt{1-\beta_2^t}\sqrt{\sigma_t^2+\xi}} - \delta_t^2$). Now remembering that, $\beta_t > \gamma^2$, the step length defined earlier $\alpha_t = \frac{2(1-\beta_2^t)(\xi+\epsilon_t)\beta_t}{L}$ can be replaced by an underestimate of it (which in the theorem statement we call the final constant step length α),

$$\alpha = \frac{(1-\beta_2)\xi}{L\sqrt{\sigma^2+\xi}}$$

□

5.3 Proving RMSProp - the version with no added shift (Proof of Theorem 2.3)

Here we consider an alternative way to give guarantees for RMSProp which does not use the ξ parameter and instead uses other assumptions on the objective function and step size modulation. But this comes at the cost of the convergence rates getting weaker.

Algorithm 4 "RMSProp With Pseudo/Penrose-Inverse", RMSProp-Penrose

Input : A sequence of strictly positive decreasing step-sizes, $\{\alpha_t = \frac{\alpha}{\sqrt{t}}\}_{t=1}^{T}$ for some $\alpha > 0$, a parameter $\beta_2 \in [0,1)$, an initial starting point $\mathbf{x}_1 \in \mathbb{R}^d$ and we are given oracle access to the gradients of $f : \mathbb{R}^d \to \mathbb{R}$
function RMSProp-Penrose($(\mathbf{x}_1, \beta_2, \{\alpha_t\}_{t=1}^{T})$)
  Initialize : $\mathbf{v}_0 = 0, \mathbf{x}_{result} = \mathbf{x}_1$
  for $t = 1, 2, \ldots$ do
    $\mathbf{m}_t = \mathbf{g}_t = \nabla f(\mathbf{x}_t)$
    $\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t^2$
    $V_t^{-\frac{1}{2}} = \sum_{i \in \text{Support}(\mathbf{v}_t)} \frac{1}{\sqrt{(\mathbf{v}_t)_i}} \mathbf{e}_i \mathbf{e}_i^T$
    $\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha_t V_t^{-\frac{1}{2}} \mathbf{m}_t$
  end for
end function

Proof. From the $L-$smoothness condition on f we have between consecutive iterates of the above algorithm,

$$f(\mathbf{x}_{t+1}) \leq f(\mathbf{x}_t) - \alpha_t \langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \rangle + \frac{L}{2} \alpha_t^2 \| V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \|^2$$

$$\implies \langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \rangle \leq \frac{1}{\alpha_t} \left(f(\mathbf{x}_t) - f(\mathbf{x}_{t+1}) \right) + \frac{L \alpha_t}{2} \| V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \|^2 \tag{16}$$

$$\tag{17}$$

Now the recursion for $\mathbf{v}_t$ can be solved to get, $\mathbf{v}_t = (1-\beta_2) \sum_{k=1}^{t} \beta_2^{t-k} \mathbf{g}_k^2$. Then $\left\| V_t^{\frac{1}{2}} \right\| \geq \frac{1}{\max_{i \in \text{Support}(\mathbf{v}_t)} \sqrt{(\mathbf{v}_t)_i}} = \frac{1}{\max_{i \in \text{Support}(\mathbf{v}_t)} \sqrt{(1-\beta_2) \sum_{k=1}^{t} \beta_2^{t-k} (\mathbf{g}_k^2)_i}} = \frac{1}{\max_{i \in \text{Support}(\mathbf{v}_t)} \sigma \sqrt{(1-\beta_2) \sum_{k=1}^{t} \beta_2^{t-k}}} = \frac{1}{\sigma \sqrt{(1-\beta_2^t)}}$. Substituting this in a lowerbound on the LHS of equation 16 we get,

$$\frac{1}{\sigma \sqrt{(1-\beta_2^t)}} \| \nabla f(\mathbf{x}_t) \|^2 \leq \langle \nabla f(\mathbf{x}_t), V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \rangle \leq \frac{1}{\alpha_t} \left(f(\mathbf{x}_t) - f(\mathbf{x}_{t+1}) \right) + \frac{L \alpha_t}{2} \| V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \|^2$$

Summing the above we get,

$$\sum_{t=1}^{T} \frac{1}{\sigma \sqrt{(1-\beta_2^t)}} \| \nabla f(\mathbf{x}_t) \|^2 \leq \sum_{t=1}^{T} \frac{1}{\alpha_t} \left(f(\mathbf{x}_t) - f(\mathbf{x}_{t+1}) \right) + \sum_{t=1}^{T} \frac{L \alpha_t}{2} \| V_t^{-\frac{1}{2}} \nabla f(\mathbf{x}_t) \|^2 \tag{18}$$

Now we substitute $\alpha_t = \frac{\alpha}{\sqrt{t}}$ and invoke the definition of B_ℓ and B_u to write the first term on the RHS of equation 18 as,

$$\sum_{t=1}^{T}\frac{1}{\alpha_t}[f(\mathbf{x}_t)-f(\mathbf{x}_{t+1})] = \frac{f(\mathbf{x}_1)}{\alpha}+\sum_{t=1}^{T}\left(\frac{f(\mathbf{x}_{t+1})}{\alpha_{t+1}}-\frac{f(\mathbf{x}_{t+1})}{\alpha_t}\right)-\frac{f(x_{T+1})}{\alpha_{T+1}}$$
$$= \frac{f(\mathbf{x}_1)}{\alpha}-\frac{f(x_{T+1})}{\alpha_{T+1}}+\frac{1}{\alpha}\sum_{t=1}^{T}f(\mathbf{x}_{t+1})(\sqrt{t+1}-\sqrt{t})$$
$$\leq \frac{B_u}{\alpha}-\frac{B_\ell\sqrt{T+1}}{\alpha}+\frac{B_u}{\alpha}(\sqrt{T+1}-1)$$

Now we bound the second term in the RHS of equation 18 as follows. Lets first define a function $P(T)$ as follows, $P(T)=\sum_{t=1}^{T}\alpha_t\|V_t^{-\frac{1}{2}}\nabla f(\mathbf{x}_t)\|^2$ and that gives us,

$$P(T)-P(T-1)=\alpha_T\sum_{i=1}^{d}\frac{\mathbf{g}_{T,i}^2}{\mathbf{v}_{T,i}}=\alpha_T\sum_{i=1}^{d}\frac{\mathbf{g}_{T,i}^2}{(1-\beta_2)\sum_{k=1}^{T}\beta_2^{T-k}\mathbf{g}_{k,i}^2}$$
$$=\frac{\alpha_T}{(1-\beta_2)}\sum_{i=1}^{d}\frac{\mathbf{g}_{T,i}^2}{\sum_{k=1}^{T}\beta_2^{T-k}\mathbf{g}_{k,i}^2}\leq\frac{d\alpha}{(1-\beta_2)\sqrt{T}}$$
$$\implies \sum_{t=2}^{T}[P(t)-P(t-1)]=P(T)-P(1)\leq\frac{d\alpha}{(1-\beta_2)}\sum_{t=2}^{T}\frac{1}{\sqrt{t}}\leq\frac{d\alpha}{2(1-\beta_2)}(\sqrt{T}-2)$$
$$\implies P(T)\leq P(1)+\frac{d\alpha}{2(1-\beta_2)}(\sqrt{T}-2)$$

So substituting the above two bounds back into the RHS of the above inequality 18and removing the factor of $\sqrt{1-\beta_2^T}<1$ from the numerator, we can define a point $\mathbf{x}_{result}$ as follows,

$$\|\nabla f(\mathbf{x}_{result})\|^2:=\min_{t=1,..,T}\|\nabla f(\mathbf{x}_t)\|^2\leq\frac{1}{T}\sum_{t=1}^{T}\|\nabla f(\mathbf{x}_t)\|^2$$
$$\leq\frac{\sigma}{T}\left(\frac{B_u}{\alpha}-\frac{B_l\sqrt{T+1}}{\alpha}+\frac{B_u}{\alpha}(\sqrt{T+1}-1)\right.$$
$$\left.+\frac{L}{2}\left[P(1)+\frac{d\alpha}{2(1-\beta_2)}(\sqrt{T}-2)\right]\right)$$

Thus it follows that for $T=O(\frac{1}{\epsilon^4})$ the algorithm 4 is guaranteed to have found at least one point $\mathbf{x}_{result}$ such that, $\|\nabla f(\mathbf{x}_{result})\|^2\leq\epsilon^2$ □

6 Conclusion

To the best of our knowledge, we present the first theoretical guarantees of convergence to criticality for the immensely popular algorithms RMSProp and ADAM in their most commonly used setting of optimizing a non-convex objective.

By our experiments, we have sought to shed light on the important topic of the interplay between adaptivity and momentum in training nets. By choosing to study textbook autoencoder architectures where various parameters of the net can be changed controllably we have been able to highlight the following two aspects that (a) the value of the gradient shifting hyperparameter ξ has a significant influence on the performance of ADAM and RMSProp and (b) ADAM seems to perform particularly well (supersedes Nesterov accelerated gradient method) when its momentum parameter β_1 is very close to 1. For the task of training autoencoders on MNIST we have verified these conclusions across different widths and depths of nets as well as in the full-batch and the mini-batch setting (with large nets) and under compression of the input/out image size. Curiously enough, this regime of β_1 being close to 1 is currently not within the reach of our proof techniques of showing convergence for ADAM. Our experiments give strong reasons to try to advance theory in this direction in future work.

References

[ABC+16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[AZNG15] Devansh Arpit, Yingbo Zhou, Hung Ngo, and Venu Govindaraju. Why regularized auto-encoders learn sparse representation? *arXiv preprint arXiv:1505.05561*, 2015.

[Bal12] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 37–49, 2012.

[BAP+17] Parnia Bahar, Tamer Alkhouli, Jan-Thorsten Peter, Christopher Jan-Steffen Brix, and Hermann Ney. Empirical investigation of optimization algorithms in neural machine translation. *The Prague Bulletin of Mathematical Linguistics*, 108(1):13–25, 2017.

[BWAA18] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. signsgd: compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434*, 2018.

[DHS11] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[DN17] Michael Denkowski and Graham Neubig. Stronger baselines for trustable results in neural machine translation. *arXiv preprint arXiv:1706.09733*, 2017.

[GB10] 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*, pages 249–256, 2010.

[GDG+15] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[GPS+18] Sébastien Gadat, Fabien Panloup, Sofiane Saadane, et al. Stochastic heavy ball. *Electronic Journal of Statistics*, 12(1):461–529, 2018.

[JNJ17] Chi Jin, Praneeth Netrapalli, and Michael I Jordan. Accelerated gradient descent escapes saddle points faster than gradient descent. *arXiv preprint arXiv:1711.10456*, 2017.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arxiv. org, 2014.

[KG17] Oleksii Kuchaiev and Boris Ginsburg. Training deep autoencoders for collaborative filtering. *arXiv preprint arXiv:1708.01715*, 2017.

[KNJK18] Rahul Kidambi, Praneeth Netrapalli, Prateek Jain, and Sham M. Kakade. On the insufficiency of existing momentum schemes for stochastic optimization. In *International Conference on Learning Representations*, 2018.

[KS17] Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628*, 2017.

[KZS+15] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015.

[LSY98] Richard B Lehoucq, Danny C Sorensen, and Chao Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, volume 6. Siam, 1998.

[MDB17] Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*, 2017.

[Nes83] Yurii Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[Och16] Peter Ochs. Local convergence of the heavy-ball method and ipiano for non-convex optimization. *arXiv preprint arXiv:1606.09070*, 2016.

[OL94] Genevieve B Orr and Todd K Leen. Momentum and optimal stochastic search. In *Proceedings of the 1993 Connectionist Models Summer School*, pages 351–357. Psychology Press, 1994.

[OW17] Michael O'Neill and Stephen J Wright. Behavior of accelerated gradient methods near critical points of nonconvex problems. *arXiv preprint arXiv:1706.07993*, 2017.

[Pol87] Boris T Polyak. Introduction to optimization. translations series in mathematics and engineering. *Optimization Software*, 1987.

[RKK18] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

[RMA+17] Akshay Rangamani, Anirbit Mukherjee, Ashish Arora, Tejaswini Ganapathy, Amitabh Basu, Sang Chin, and Trac D Tran. Critical points of an autoencoder can provably recover sparsely used overcomplete dictionaries. *arXiv preprint arXiv:1708.03735*, 2017.

[RMC15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[SMDH13] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[TH12] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. *University of Toronto, Technical Report*, 2012.

[Tow08] Jamie Townsend. A new trick for calculating Jacobian vector products. https://j-towns.github.io/2017/06/12/A-new-trick.html, 2008. [Online; accessed 17-May-2018].

[VLL+10] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[WKH94] Wim Wiegerinck, Andrzej Komoda, and Tom Heskes. Stochastic dynamics of learning with momentum in neural networks. *Journal of Physics A: Mathematical and General*, 27(13):4425, 1994.

[WRS+17] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems*, pages 4151–4161, 2017.

[XBK+15] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[YLL16] Tianbao Yang, Qihang Lin, and Zhe Li. Unified convergence analysis of stochastic momentum methods for convex and non-convex optimization. *arXiv preprint arXiv:1604.03257*, 2016.

[YYS16] Kun Yuan, Bicheng Ying, and Ali H Sayed. On the influence of momentum acceleration on online learning. *Journal of Machine Learning Research*, 17(192):1–66, 2016.

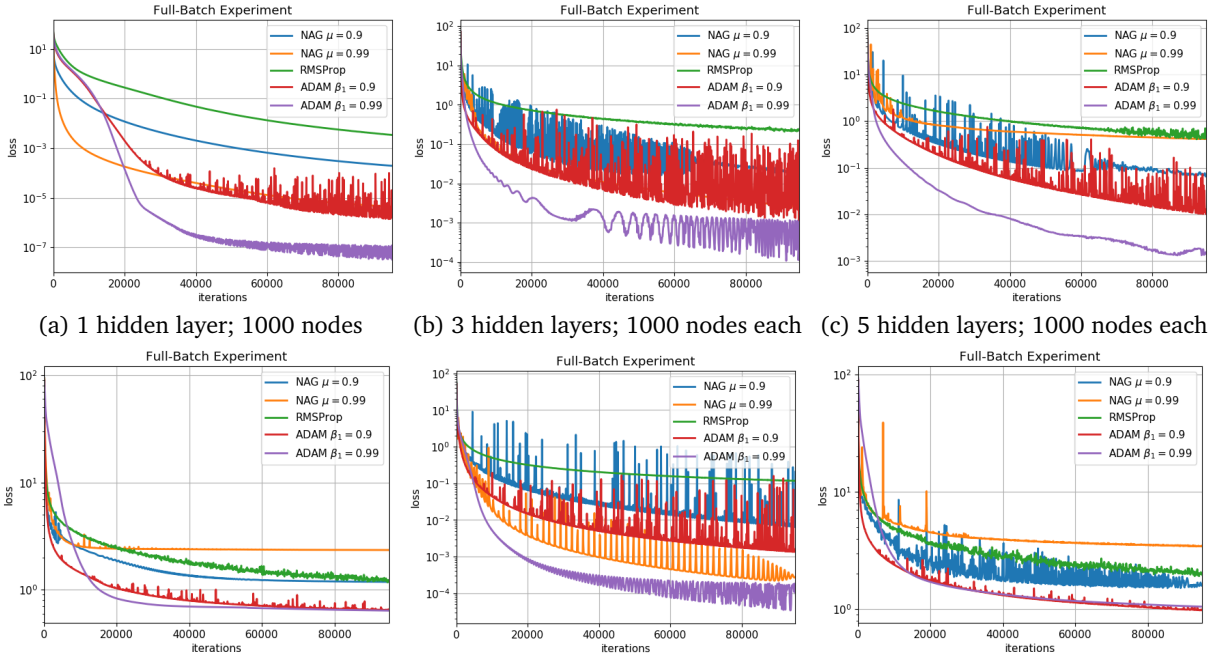
[ZK93] SK Zavriev and FV Kostyuk. Heavy-ball method in nonconvex optimization problems. *Computational Mathematics and Modeling*, 4(4):336–341, 1993.

Appendix

A Additional Experiments

A.1 Additional full-batch experiments on 22×22 sized images

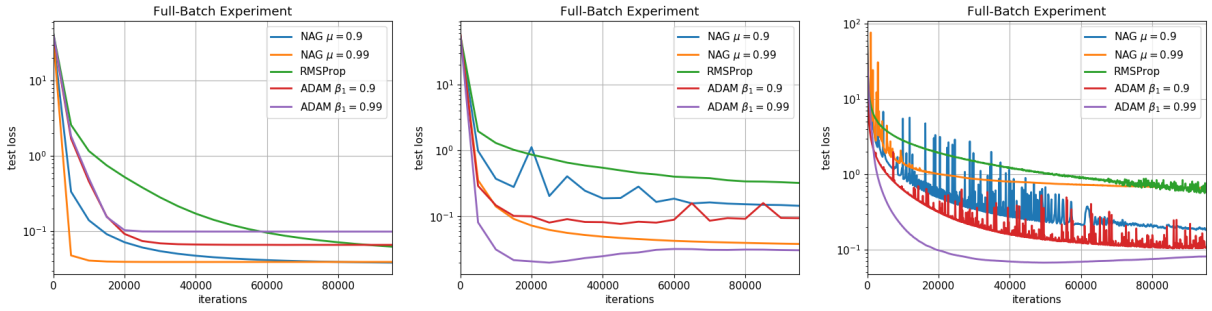
In Figures 6, 7 and 8, we show training loss, test loss and gradient norm results for a variety of additional network architectures. Across almost all network architectures, our main results remain consistent. ADAM with $\beta_1 = 0.99$ consistently reaches lower training loss values as well as better generalization than NAG.



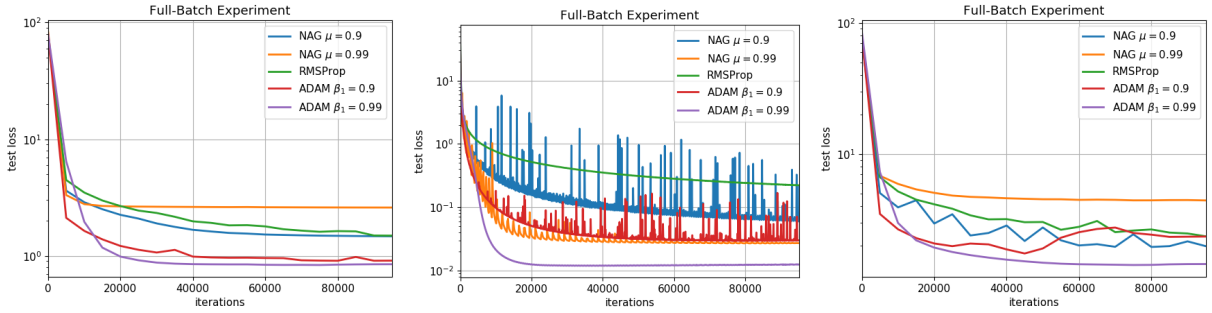
(a) 1 hidden layer; 1000 nodes (b) 3 hidden layers; 1000 nodes each (c) 5 hidden layers; 1000 nodes each

(d) 3 hidden layers; 300 nodes (e) 3 hidden layer; 3000 nodes (f) 5 hidden layer; 300 nodes

Figure 6: Loss on training set; Input image size 22×22

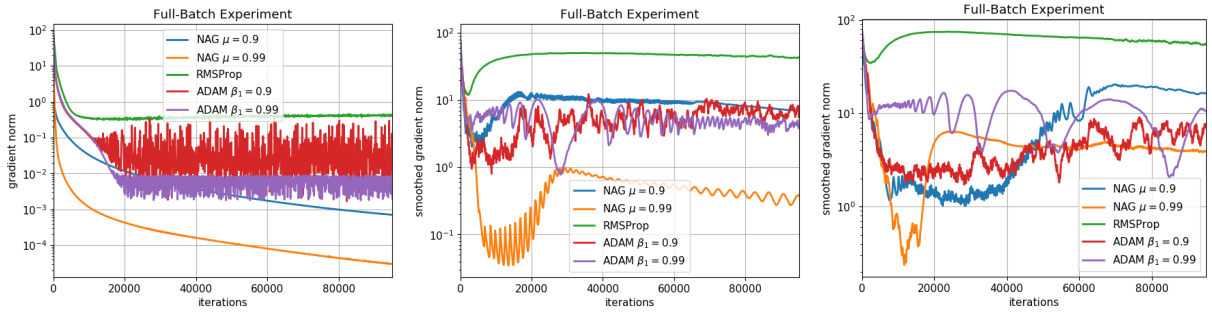


(a) 1 hidden layer; 1000 nodes (b) 3 hidden layers; 1000 nodes each (c) 5 hidden layers; 1000 nodes each

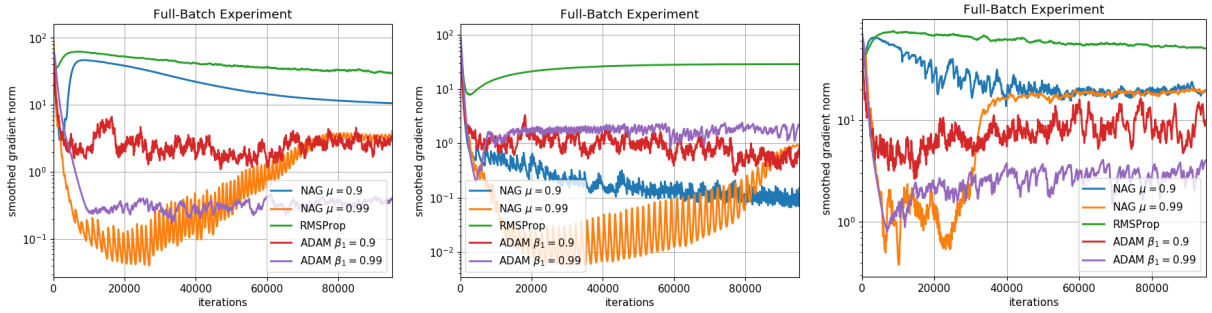


(d) 3 hidden layers; 300 nodes (e) 3 hidden layer; 3000 nodes (f) 5 hidden layer; 300 nodes

Figure 7: Loss on test set; Input image size 22×22



(a) 1 hidden layer; 1000 nodes (b) 3 hidden layers; 1000 nodes each (c) 5 hidden layers; 1000 nodes each



(d) 3 hidden layers; 300 nodes (e) 3 hidden layer; 3000 nodes (f) 5 hidden layer; 300 nodes

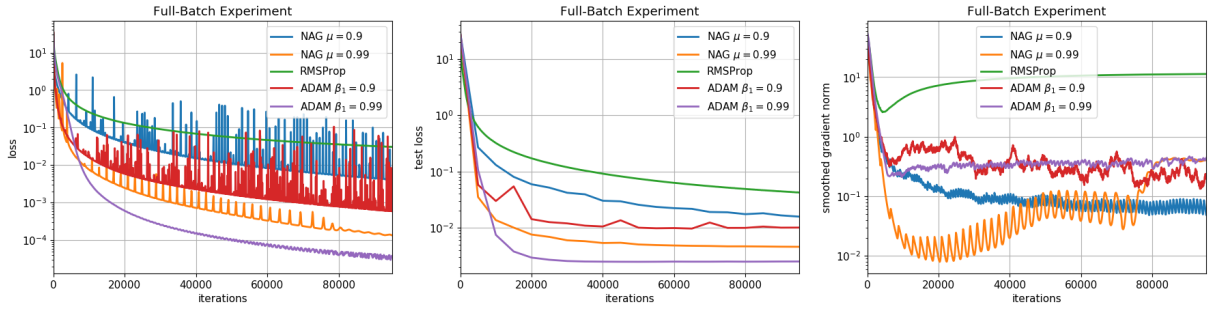
Figure 8: Norm of gradient on training set; Input image size 22×22

A.2 Are the full-batch results consistent across different input dimensions?

To test whether our conclusions are consistent across different input dimensions, we do two experiments where we resize the 22×22 MNIST image to 17×17 and to 12×12. Resizing is done using TensorFlow's `tf.image.resize_images` method, which uses bilinear interpolation.

A.2.1 Input images of size 17×17

Figure 9 shows results on input images of size 17×17 on a 3 layer network with 1000 hidden nodes in each layer. Our main results extend to this input dimension, where we see ADAM with $\beta_1 = 0.99$ both converging the fastest as well as generalizing the best, while NAG does better than ADAM with $\beta_1 = 0.9$.

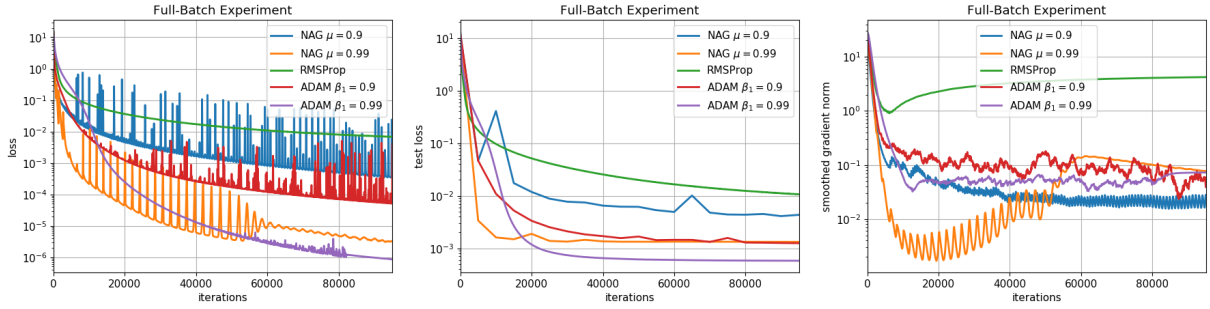


(a) Training loss (b) Test loss (c) Gradient norm

Figure 9: Full-batch experiments with input image size 17×17

A.2.2 Input images of size 12×12

Figure 10 shows results on input images of size 12×12 on a 3 layer network with 1000 hidden nodes in each layer. Our main results extend to this input dimension as well. ADAM with $\beta_1 = 0.99$ converges the fastest as well as generalizes the best, while NAG does better than ADAM with $\beta_1 = 0.9$.

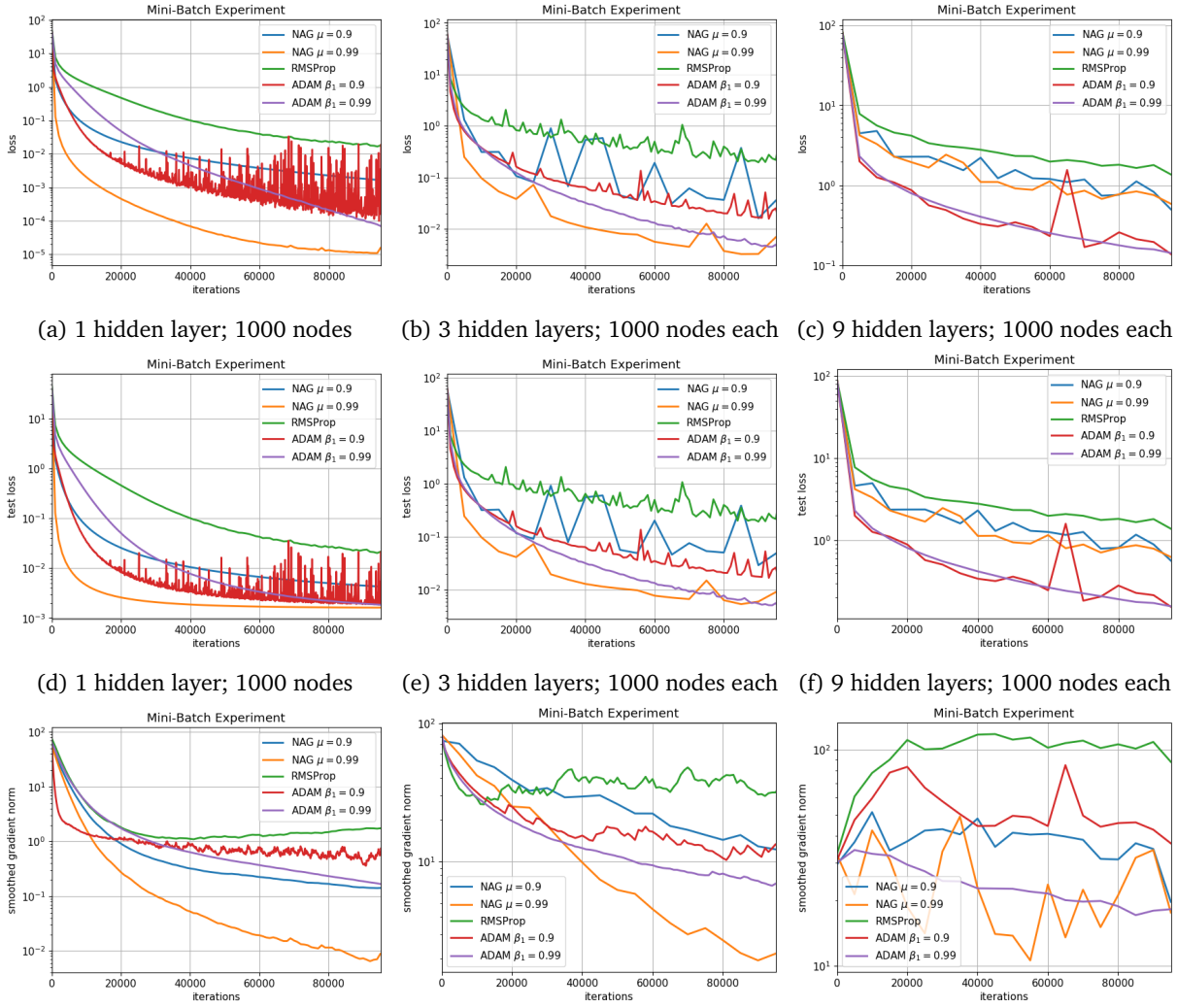


(a) Training loss (b) Test loss (c) Gradient norm

Figure 10: Full-batch experiments with input image size 12×12

A.3 Additional mini-batch experiments on 22×22 sized images

In Figure 11, we present results on additional neural net architectures on mini-batches of size 100 with an input dimension of 22×22. We see that most of our full-batch results extend to the mini-batch case.



(a) 1 hidden layer; 1000 nodes (b) 3 hidden layers; 1000 nodes each (c) 9 hidden layers; 1000 nodes each

(d) 1 hidden layer; 1000 nodes (e) 3 hidden layers; 1000 nodes each (f) 9 hidden layers; 1000 nodes each

(g) 1 hidden layer; 1000 nodes (h) 3 hidden layers; 1000 nodes each (i) 9 hidden layers; 1000 nodes each

Figure 11: Experiments on various networks with mini-batch size 100 on full MNIST dataset with input image size 22×22. First row shows the loss on the full training set, middle row shows the loss on the test set, and bottom row shows the norm of the gradient on the training set.