Reducing the variance in online optimization by transporting past gradients

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Most stochastic optimization methods use gradients once before discarding them. While variance reduction methods have shown that reusing past gradients can be beneficial when there is a finite number of datapoints, they do not easily extend to the online setting. One issue is the staleness due to using past gradients. We propose to correct this staleness using the idea of *implicit gradient transport* (IGT) which transforms gradients computed at previous iterates into gradients evaluated at the current iterate without using the Hessian explicitly. In addition to reducing the variance and bias of our updates over time, IGT can be used as a drop-in replacement for the gradient estimate in a number of well-understood methods such as heavy ball or Adam. We show experimentally that it achieves state-of-the-art results on a wide range of architectures and benchmarks. Additionally, the IGT gradient estimator yields the optimal asymptotic convergence rate for online stochastic optimization in the restricted setting where the Hessians of all component functions are equal.

1 Introduction

We wish to solve the following minimization problem:

$$\theta^* = \arg\min_\theta E_{x\sim p}[f(\theta, x)] \ , \tag{1}$$

where we only have access to samples x and to a first-order oracle that gives us, for a given θ and a given x, the derivative of $f(\theta, x)$ with respect to θ, i.e. $\frac{\partial f(\theta,x)}{\partial\theta} = g(\theta, x)$. It is known [35] that, when f is smooth and strongly convex, there is a converging algorithm for Problem 1 that takes the form $\theta_{t+1} = \theta_t - \alpha_t g(\theta_t, x_t)$, where x_t is a sample from p. This algorithm, dubbed stochastic gradient (SG), has a convergence rate of $O(1/t)$ (see for instance [4]), within a constant factor of the minimax rate for this problem. When one has access to the true gradient $g(\theta) = E_{x\sim p}[g(\theta, x)]$ rather than just a sample, this rate dramatically improves to $O(e^{-\nu t})$ for some $\nu > 0$.

In addition to hurting the convergence speed, noise in the gradient makes optimization algorithms harder to tune. Indeed, while full gradient descent is convergent for constant stepsize α, and also amenable to line searches to find a good value for that stepsize, the stochastic gradient method from [35] with a constant stepsize only converges to a ball around the optimum [38].[1] Thus, to achieve convergence, one needs to use a decreasing stepsize. While this seems like a simple modification, the precise decrease schedule can have a dramatic impact on the convergence speed. While theory prescribes $\alpha_t = O(t^{-\alpha})$ with $\alpha \in (1/2, 1]$ in the smooth case, practictioners often use larger stepsizes like $\alpha_t = O(t^{-1/2})$ or even constant stepsizes.

---

[1] Under some conditions, it does converge linearly to the optimum [e.g., 45]

When the distribution p has finite support, Eq. 1 becomes a finite sum and, in that setting, it is possible to achieve efficient variance reduction and drive the noise to zero, allowing stochastic methods to achieve linear convergence rates [24, 17, 50, 28, 42, 5]. Unfortunately, the finite support assumption is critical to these algorithms which, while valid in many contexts, does not have the broad applicability of the standard SG algorithm. Several works have extended these approaches to the online setting by applying these algorithms while increasing N [2, 14] but they need to revisit past examples mutiple times and are not truly online.

Another line of work reduces variance by averaging iterates [33, 22, 3, 10, 7, 6, 16]. While these methods converge for a constant stepsize in the stochastic case[2], their practical speed is heavily dependent on the fraction of iterates kept in the averaging, a hyperparameter that is thus hard to tune, and they are rarely used in deep learning.

Our work combines two existing ideas and adds a third: a) At every step, it updates the parameters using a weighted average of past gradients, like in SAG [24, 40], albeit with a different weighting scheme; b) It reduces the bias and variance induced by the use of these old gradients by transporting them to "equivalent" gradients computed at the current point, similar to [11]; c) It does so implicitly by computing the gradient at a parameter value different from the current one. The resulting gradient estimator can then be used as a plug-in replacement of the stochastic gradient within any optimization scheme. Experimentally, both SG using our estimator and its momentum variant outperform the most commonly used optimizers in deep learning.

2 Momentum and other approaches to dealing with variance

Stochastic variance reduction methods use an average of past gradients to reduce the variance of the gradient estimate. At first glance, it seems like their updates are similar to that of momentum [32], also known as the heavy ball method, which performs the following updates[3]:

$$\begin{aligned} v_t &= \gamma_t v_{t-1} + (1-\gamma_t) g(\theta_t, x_t), \qquad v_0 = g(\theta_0, x_0) \\ \theta_{t+1} &= \theta_t - \alpha_t v_t \ . \end{aligned}$$

When $\gamma_t = \gamma$, this leads to $\theta_{t+1} = \theta_t - \alpha_t \left(\gamma^t g(\theta_0, x_0) + (1-\gamma)\sum_{i=1}^{t} \gamma^{t-i} g(\theta_i, x_i)\right)$. Hence, the heavy ball method updates the parameters of the model using an average of past gradients, bearing similarity with SAG [24], albeit with exponential instead of uniform weights.

Interestingly, while momentum is a popular method for training deep networks, its theoretical analysis in the stochastic setting is limited [44], except in the particular setting when the noise converges to 0 at the optimum [26]. Also surprising is that, despite the apparent similarity with stochastic variance reduction methods, current convergence rates are slower when using $\gamma > 0$ in the presence of noise [39], although this might be a limitation of the analysis.

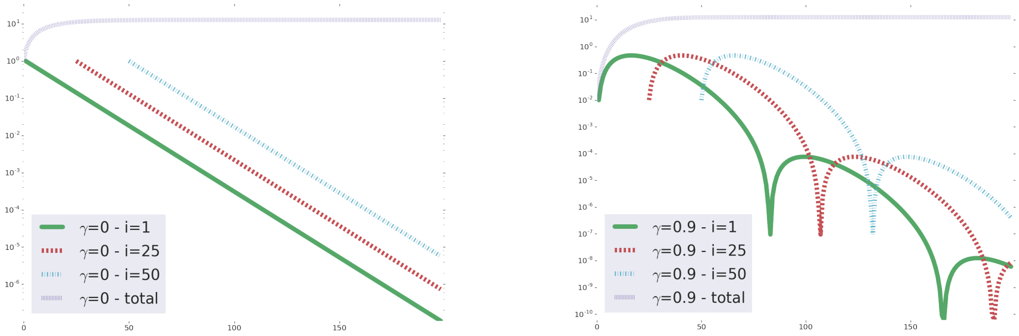
2.1 Momentum and variance

We propose here an analysis of how, on quadratics, using past gradients as done in momentum does not lead to a decrease in variance. If gradients are stochastic, then $\Delta_t = \theta_t - \theta^*$ is a random variable. Denoting ϵ_i the noise at timestep i, i.e. $g(\theta_i, x_i) = g(\theta_i) + \epsilon_i$, and writing $\Delta_t - E[\Delta_t] = \alpha \sum_{i=0}^{t} N_{i,t}\epsilon_i$, with $N_{i,t}$ the impact of the noise of the i-th datapoint on the t-th iterate, we may now analyze the total impact of each ϵ_i on the iterates. Figure 1 shows the impact of ϵ_i on $\Delta_t - E[\Delta_t]$ as measured by $N_{i,t}^2$ for three datapoints ($i = 1$, $i = 25$ and $i = 50$) as a function of t for stochastic gradient ($\gamma = 0$, left) and momentum ($\gamma = 0.9$, right). As we can see, when using momentum, the variance due to a given datapoint first increases as the noise influences both the next iterate (through the parameter update) and the subsequent updates (through the velocity). Due to the weight $1 - \gamma$ when a point is first sampled, a larger value of γ leads to a lower immediate impact of the noise of a given point on the iterates. However, a larger γ also means that the noise of a given gradient is kept longer, leading to little or no decrease of the total variance (dashed blue curve). Even in the case of stochastic gradient, the noise at a given timestep carries over to subsequent timesteps, even if the old gradients are not used for the update, as the iterate itself depends on the noise.

---

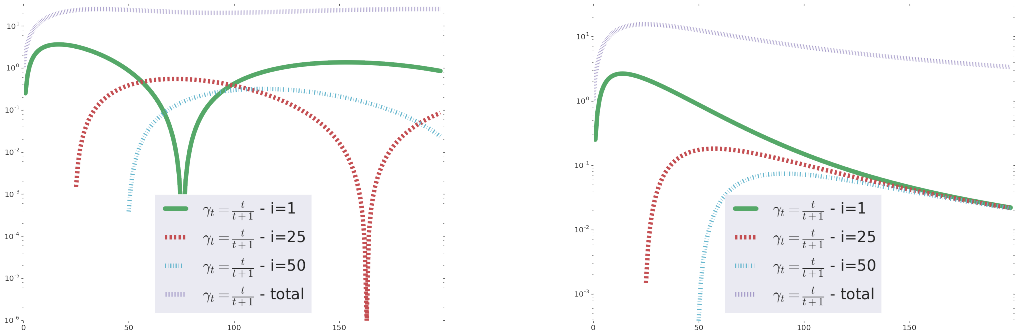
[2] Under some conditions on f.

[3] This is slightly different from the standard formulation but equivalent for constant γ_t.



(a) Stochastic gradient

(b) Momentum - $\gamma = 0.9$



(c) Momentum - $\gamma_t = 1 - \frac{1}{t}$

(d) Momentum - $\gamma_t = 1 - \frac{1}{t}$ with IGT.

Figure 1: Variance induced over time by the noise from three different datapoints ($i = 1$, $i = 25$ and $i = 50$) as well as the total variance for SG ($\gamma = 0$, *top left*), momentum with fixed $\gamma = 0.9$ (*top right*), momentum with increasing $\gamma_t = 1 - \frac{1}{t}$ without (*bottom left*) and with (*bottom right*) transport. The impact of the noise of each gradient ϵ_i increases for a few iterations then decreases. Although a larger γ reduces the maximum impact of a given datapoint, the total variance does not decrease. With transport, noises are now equal and total variance decreases. The y-axis is on a log scale.

At every timestep, the contribution to the noise of the 1st, the 25th and the 50th points in Fig. 1 is unequal. If we assume that the ϵ_i are i.i.d., then the total variance would be minimal if the contribution from each point was equal. Further, one can notice that the impact of datapoint i is only a function of $t - i$ and not of t. This guarantees that the total noise will not decrease over time.

To address these two points, one can increase the momentum parameter over time. In doing so, the noise of new datapoints will have a decreasing impact on the total variance as their gradient is multiplied by $1 - \gamma_t$. Figure 1c shows the impact $N_{i,t}^2$ of each noise ϵ_i for an increasing momentum $\gamma_t = 1 - \frac{1}{t}$. The peak of noise for $i = 25$ is indeed lower than that of $i = 1$. However, the variance still does not go to 0. This is because, as the momentum parameter increases, the update is an average of many gradients, including stale ones. Since these gradients were computed at iterates already influenced by the noise over previous datapoints, that past noise is amplified, as testified by the higher peak at $i = 1$ for the increasing momentum. Ultimately, increasing momentum does not lead to a convergent algorithm in the presence of noise when using a constant stepsize.

2.2 SAG and Hessian modelling

The impact of the staleness of the gradients on the convergence is not limited to momentum. In SAG, for instance, the excess error after k updates is proportional to $\left(1 - \min\left\{\frac{1}{16\widehat{\kappa}}, \frac{1}{8N}\right\}\right)^k$, compared to the excess error of the full gradient method which is $\left(1 - \frac{1}{\kappa}\right)^k$ where κ is the condition number of the problem. [4] The difference between the two rates is larger when the minimum in the SAG rate is the second term. This happens either when $\widehat{\kappa}$ is small, i.e. the problem is well conditioned and a lot

---

[4] The $\widehat{\kappa}$ in the convergence rate of SAG is generally larger than the κ in the full gradient algorithm.

of progress is made at each step, or when N is large, i.e. there are many points to the training set. Both cases imply that a large distance has been travalled between two draws of the same datapoint.

Recent works showed that correcting for that staleness by modelling the Hessian [46, 11] leads to improved convergence. As momentum uses stale gradients, the velocity is an average of current and past gradients and thus can be seen as an estimate of the true gradient at a point which is not the current one but rather a convex combination of past iterates. As past iterates depend on the noise of previous gradients, this bias in the gradients amplifies the noise and leads to a non-converging algorithm. We shall thus "transport" the old stochastic gradients $g(\theta_i, x_i)$ to make them closer to their corresponding value at the current iterate, $g(\theta_t, x_i)$. Past works did so using the Hessian or an explicit approximation thereof, which can be expensive and difficult to compute and maintain. We will resort to using *implicit transport*, a new method that aims at compensating the staleness of past gradients without making explicit use of the Hessian.

3 Converging optimization through implicit gradient tranport

Before showing how to combine the advantages of both increasing momentum and gradient transport, we demonstrate how to tranport gradients implicitly. This transport is only exact under a strong assumption that will not hold in practice. However, this result will serve to convey the intuition behind implicit gradient transport. We will show in Section 4 how to mitigate the effect of the unsatisfied assumption.

3.1 Implicit gradient transport

Let us assume that we received samples $x_0, \ldots, x_t$ in an online fashion. We wish to approach the full gradient $g_t(\theta_t) = \frac{1}{t+1} \sum_{i=0}^{t} g(\theta_t, x_i)$ as accurately as possible. We also assume here that a) We have a noisy estimate $\widehat{g}_{t-1}(\theta_{t-1})$ of $g_{t-1}(\theta_{t-1})$; b) We can compute the gradient $g(\theta, x_t)$ at any location θ. We shall seek a θ such that

$$\frac{t}{t+1}\widehat{g}_{t-1}(\theta_{t-1}) + \frac{1}{t+1}g(\theta, x_t) \approx g_t(\theta_t) \ .$$

To this end, we shall make the following assumption:

Assumption 3.1. *All individual functions $f(\cdot, x)$ are quadratics with the same Hessian H.*

This is the same assumption as [10, Section 4.1]. Although it is unlikely to hold in practice, we shall see that our method still performs well when that assumption is violated.

Under Assumption 3.1, we then have (see details in Appendix)

$$\begin{aligned} g_t(\theta_t) &= \frac{t}{t+1}g_{t-1}(\theta_t) + \frac{1}{t+1}g(\theta_t, x_t) \\ &\approx \frac{t}{t+1}\widehat{g}_{t-1}(\theta_{t-1}) + \frac{1}{t+1}g(\theta_t + t(\theta_t - \theta_{t-1}), x_t) \ . \end{aligned}$$

Thus, we can transport our current estimate of the gradient by computing the gradient on the new point at a shifted location $\theta = \theta_t + t(\theta_t - \theta_{t-1})$. This extrapolation step is reminiscent of Nesterov's acceleration with the difference that the factor in front of $\theta_t - \theta_{t-1}$, t, is not bounded.

3.2 Combining increasing momentum and implicit gradient transport

We now describe our main algorithm, Implicit Gradient Transport (IGT). IGT uses an increasing momentum $\gamma_t = \frac{t}{t+1}$. At each step, when updating the velocity, it computes the gradient of the new point at an extrapolated location so that the velocity v_t is a good estimate of the true gradient $g(\theta_t)$.

We can rewrite the updates to eliminate the velocity v_t, leading to the update:

$$\theta_{t+1} = \frac{2t+1}{t+1}\theta_t - \frac{t}{t+1}\theta_{t-1} - \frac{\alpha}{t+1}g\left(\theta_t + t(\theta_t - \theta_{t-1}), x_t\right) \ . \qquad \text{(IGT)}$$

We see in Fig. 1d that IGT allows a reduction in the total variance, thus leading to convergence with a constant stepsize. This is captured by the following proposition:

Proposition 3.1. *If f is a quadratic function with positive definite Hessian H with largest eigenvalue L and condition number κ and if the stochastic gradients satisfy: $g(\theta, x) = g(\theta) + \epsilon$ with ϵ a random i.i.d. noise with covariance bounded by BI, then Eq. IGT with stepsize $\alpha = 1/L$ leads to iterates θ_t satisfying*

$$E[\|\theta_t - \theta^*\|^2] \leq \left(1 - \frac{1}{\kappa}\right)^{2t} \|\theta_0 - \theta^*\|^2 + \frac{d\alpha^2 B \bar{\nu}_0^2}{t} \ ,$$

with $\nu = (2 + 2\log\kappa)\kappa$ for every $t > 2\kappa$.

The proof of Prop. 3.1 is provided in the appendix.

Despite this theoretical result, two limitations remain: First, Prop. 3.1 shows that IGT does not improve the dependency on the conditioning of the problem; Second, the assumption of equal Hessians is unlikely to be true in practice, leading to an underestimation of the bias. We address the conditioning issue in the next section and the assumption on the Hessians in Section 4.

3.3 IGT as a plug-in gradient estimator

We demonstrated that the IGT estimator has lower variance than the stochastic gradient estimator for quadratic objectives. IGT can also be used as a drop-in replacement for the stochastic gradient in an existing, popular first order method: the heavy ball (HB). This is captured by the following two propositions:

Proposition 3.2 (Non-stochastic)**.** *In the non-stochastic case, where $B = 0$, variance is equal to 0 and Heavyball-IGT achieves the accelerated linear rate $O\big(\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^t\big)$ using the known, optimal heavy ball tuning, $\mu = \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^2$, $\alpha = (1 + \sqrt{\mu})^2/L$.*

Proposition 3.3 (Online, stochastic)**.** *When $B > 0$, there exist constant hyperparameters $\alpha > 0$, $\mu > 0$ such that $\|E[\theta_t - \theta^*]\|^2$ converges to zero linearly, and the variance is $\tilde{O}(1/t)$.*

The pseudo-code can be found in Algorithm 1.

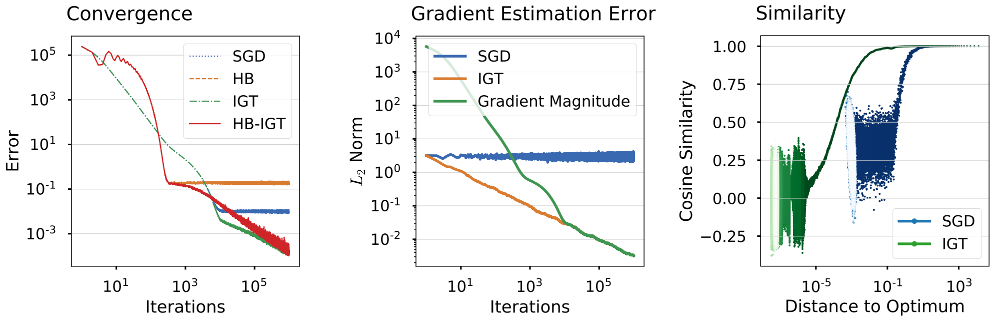
Algorithm 1 Heavyball-IGT

```
procedure HEAVYBALL-IGT(Stepsize $\alpha$, Momentum $\mu$, Initial parameters $\theta_0$)
    $v_0 \leftarrow g(\theta_0, x_0)$ ,  $w_0 \leftarrow -\alpha v_0$ ,  $\theta_1 \leftarrow \theta_0 + w_0$
    for $t = 1, \dots, T-1$ do
        $\gamma_t \leftarrow \frac{t}{t+1}$
        $v_t \leftarrow \gamma_t v_{t-1} + (1-\gamma_t) g\left(\theta_t + \frac{\gamma_t}{1-\gamma_t}(\theta_t - \theta_{t-1}), x_t\right)$
        $w_t \leftarrow \mu w_{t-1} - \alpha v_t$
        $\theta_{t+1} \leftarrow \theta_t + w_t$
    end for
    return $\theta_T$
end procedure
```

4 IGT and Anytime Tail Averaging

So far, IGT weighs all gradients equally. This is because, with equal Hessians, one can perfectly transport these gradients irrespective of the distance travelled since they were computed. In practice, the individual Hessians are not equal and might change over time. In that setting, the transport induces an error which grows with the distance travelled. We wish to average a linearly increasing number of gradients, to maintain the $O(1/t)$ rate on the variance, while forgetting about the oldest gradients to decrease the bias. To this end, we shall use *anytime tail averaging* [23], named in reference to the tail averaging technique used in optimization [16].

Tail averaging is an online averaging technique where only the last points, usually a constant fraction c of the total number of points seen, is kept. Maintaining the exact average at every timestep is memory inefficient and anytime tail averaging performs an approximate averaging using $\gamma_t = \frac{c(t-1)}{1+c(t-1)}\left(1 - \frac{1}{c}\sqrt{\frac{1-c}{t(t-1)}}\right)$. We refer the reader to [23] for additional details.



(a) (b) (c)

Figure 2: Analysis of IGT on quadratic loss functions. (a) Comparison of convergence curves for multiple algorithms. As expected, the IGT family of algorithms converges to the solution while stochastic gradient algorithms can not. (b) The blue and orange curves show the norm of the noise component in the SGD and IGT gradient estimates, respectively. The noise component of SGD remains constant, while it decreases at a rate $1/\sqrt{t}$ for IGT. The green curve shows the norm of the IGT gradient estimate. (c) Cosine similarity between the full gradient and the SGD/IGT estimates.

5 Impact of IGT on bias and variance in the ideal case

To understand the behaviour of IGT when Assumption 3.1 is verified, we minimize a strongly convex quadratic function with Hessian $Q \in \mathbb{R}^{100\times 100}$ with condition number 1000, and we have access to the gradient corrupted by noise ϵ_t, where $\epsilon_t \sim N(0, 0.3 \cdot I_{100})$. In that scenario where all Hessians are equal and implicit gradient transport is exact, Fig. 2a confirms the $O(1/t)$ rate of IGT with constant stepsize while SGD and HB only converge to a ball around the optimum.

To further understand the impact of IGT, we study the quality of the gradient estimate. Standard stochastic methods control the variance of the parameter update by scaling it with a decreasing stepsize, which slows the optimization down. With IGT, we hope to have a low variance while maintaining a norm of the update comparable to that obtained with gradient descent. To validate the quality of our estimator, we optimized a quadratic function using IGT, collecting iterates θ_t. For each iterate, we computed the squared error between the true gradient and either the stochastic or the IGT gradient. In this case where both estimators are unbiased, this is the trace of the noise covariance of our estimators. The results in Figure 2b show that, as expected, this noise decreases linearly for IGT and is constant for SGD.

We also analyse the direction and magnitude of the gradient of IGT on the same quadratic setup. Figure 2c displays the cosine similarity between the true gradient and either the stochastic or the IGT gradient, as a function of the distance to the optimum. We see that, for the same distance, the IGT gradient is much more aligned with the true gradient than the stochastic gradient is, confirming that variance reduction happens without the need for scaling the estimate.

6 Experiments

While Section 5 confirms the performance of IGT in the ideal case, the assumption of identical Hessians almost never holds in practice. In this section, we present results on more realistic and larger scale machine learning settings. All experiments are extensively described in the Appendix A and additional baselines compared in Appendix B.

6.1 Supervised learning

CIFAR10 image classification We first consider the task of training a ResNet-56 model [12] on the CIFAR-10 image classification dataset [19]. We use TF official models code and setup [1], varying only the optimizer: SGD, HB, Adam and our algorithm with anytime tail averaging both on its own (ITA) and combined with Heavy Ball (HB-ITA). We tuned the step size for each algorithm by running experiments using a logarithmic grid. To factor in ease of tuning [48], we used Adam's

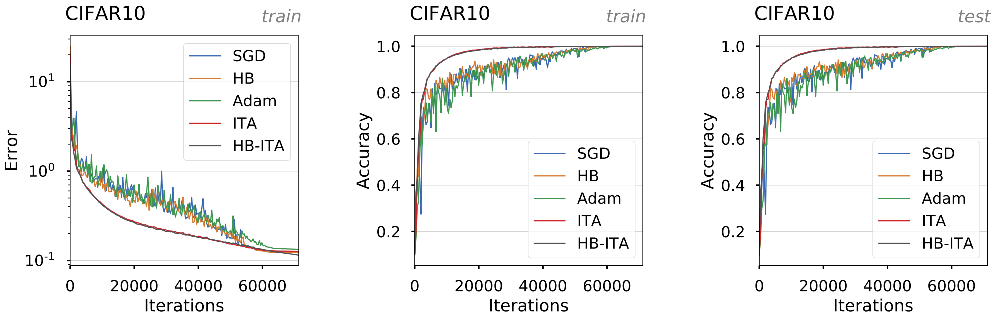


Figure 3: Resnet-56 on CIFAR10. **Left**: Train loss. **Center**: Train accuracy. **Right**: Test accuracy.

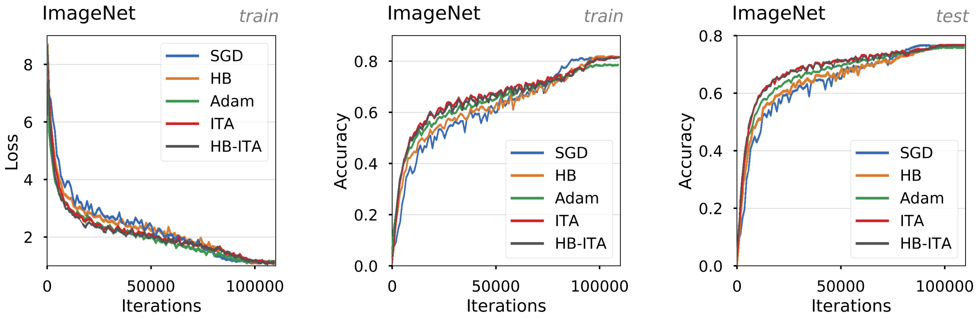


Figure 4: ResNet-50 on ImageNet. **Left**: Train loss. **Center**: Train accuracy. **Right**: Test accuracy.

default parameter values and a value of 0.9 for HB's parameter. We used a linearly decreasing stepsize as it was shown to be simple and perform well [43]. For each optimizer we selected the hyperparameter combination that is fastest to reach a consistently attainable target train loss [43]. Selecting the hyperparameter combination reaching the lowest training loss yields qualitatively identical curves. Figure 3 presents the results, showing that IGT with the exponential anytime tail average performs favourably, both on its own and combined with Heavy Ball: the learning curves show faster improvement and are much less noisy.

ImageNet image classification We also consider the task of training a ResNet-50 model[12] on the larger ImageNet dataset [36]. The setup is similar to the one used for CIFAR10 with the difference that we trained using larger minibatches (1024 instead of 128). In Figure 4, one can see that IGT is as fast as Adam for the train loss, faster for the train accuracy and reaches the same final performance, which Adam does not. We do not see the noise reduction we observed with CIFAR10, which could be explained by the larger batch size (see Appendix A.1).

IMDb sentiment analysis We train a bi-directional LSTM on the IMDb Large Movie Review Dataset for 200 epochs. [27] We observe that while the training convergence is comparable to HB, HB-ITA performs better in terms of validation and test accuracy. In addition to the baseline and IGT methods, we also train a variant of Adam using the ITA gradients, dubbed **Adam-ITA**, which performs similarly to Adam.

6.2 Reinforcement learning

Linear-quadratic regulator We cast the classical linear-quadratic regulator (LQR) [21] as a policy learning problem to be optimized via gradient descent. This setting is extensively described in Appendix A. Note that despite their simple linear dynamics and a quadratic cost functional, LQR systems are notoriously difficult to optimize due to the non-convexity of the loss landscape. [8]

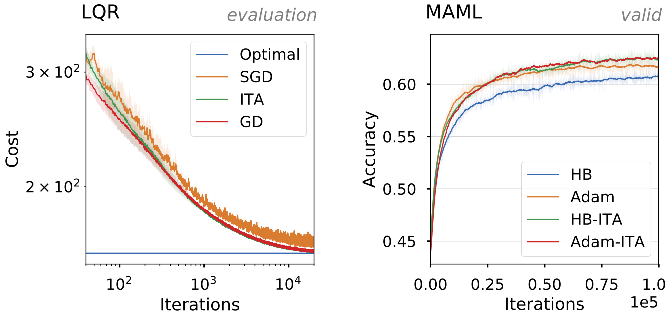


Figure 5: Validation curves for different large-scale machine learning settings. Shading indicates one standard deviation computed over three random seeds. **Left**: Reinforcement learning via policy gradient on a LQR system. **Right**: Meta-learning using MAML on Mini-Imagenet.

The left chart in Figure 5 displays the evaluation cost computed along training and averaged over three random seeds. The first method (**Optimal**) indicates the cost attained when solving the algebraic Riccati equation of the LQR – this is the optimal solution of the problem. **SGD** minimizes the costs using the REINFORCE [47] gradient estimator, averaged over 600 trajectories. **ITA** is similar to SGD but uses the ITA gradient computed from the REINFORCE estimates. Finally, **GD** uses the analytical gradient by taking the expectation over the policy.

We make two observations from the above chart. First, ITA initially suffers from the stochastic gradient estimate but rapidly matches the performance of GD. Notably, both of them converge to a solution significantly better than SGD, demonstrating the effectiveness of the variance reduction mechanism. Second, the convergence curve is smoother for ITA than for SGD, indicating that the ITA iterates are more likely to induce similar policies from one iteration to the next. This property is particularly desirable in reinforcement learning as demonstrated by the popularity of trust-region methods in large-scale applications. [41, 29]

6.3 Meta-learning

Model-agnostic meta-learning We now investigate the use of IGT in the *model-agnostic meta-learning* (MAML) setting. [9] We replicate the 5 ways classification setup with 5 adaptation steps on tasks from the Mini-Imagenet dataset [34]. This setting is interesting because of the many sources contributing to noise in the gradient estimates: the stochastic meta-gradient depends on the product of 5 stochastic Hessians computed over only 10 data samples, and is averaged over only 4 tasks. We substitute the meta-optimizer with each method, select the stepsize that maximizes the validation accuracy after 10K iterations, and use it to train the model for 100K iterations.

The right graph of Figure 5 compares validation accuracies for three random seeds. We observe that methods from the IGT family significantly outperform their stochastic meta-gradient counter-part, both in terms of convergence rate and final accuracy. Those results are also reflected in the final test accuracies where Adam-ITA (65.16%) performs best, followed by HB-ITA (64.57%), then Adam (63.70%), and finally HB (63.08%).

7 Conclusion and open questions

We proposed a simple optimizer which, by reusing past gradients and transporting them, offers excellent performance on a variety of problems. While it adds an additional parameter, the ratio of examples to be kept in the tail averaging, it remains competitive across a wide range of such values. Further, by providing a higher quality gradient estimate that can be plugged in any existing optimizer, we expect it to be applicable to a wide range of problems. As the IGT is similar to momentum, this further raises the question on the links between variance reduction and curvature adaptation. Whether there is a way to combine the two without using momentum on top of IGT remains to be seen.

References

[1] The TensorFlow Authors. Tensorflow official resnet model. 2018.

[2] Reza Babanezhad, Mohamed Osama Ahmed, Alim Virani, Mark Schmidt, Jakub Konečný, and Scott Sallinen. Stop wasting my gradients: Practical SVRG. In *Advances in Neural Information Processing Systems*, 2015.

[3] Francis Bach and Eric Moulines. Non-strongly-convex smooth stochastic approximation with convergence rate o (1/n). In *Advances in Neural Information Processing Systems*, pages 773–781, 2013.

[4] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[5] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pages 1646–1654, 2014.

[6] Aymeric Dieuleveut, Alain Durmus, and Francis Bach. Bridging the gap between constant step size stochastic gradient descent and markov chains. *arXiv preprint arXiv:1707.06386*, 2017.

[7] Aymeric Dieuleveut, Nicolas Flammarion, and Francis Bach. Harder, better, faster, stronger convergence rates for least-squares regression. *The Journal of Machine Learning Research*, 18(1):3520–3570, 2017.

[8] Maryam Fazel, Rong Ge, Sham M. Kakade, and Mehran Mesbahi. Global convergence of policy gradient methods for linearized control problems. *CoRR*, abs/1801.05039, 2018.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[10] Nicolas Flammarion and Francis Bach. From averaging to acceleration, there is only a step-size. In *Conference on Learning Theory*, pages 658–695, 2015.

[11] Robert M Gower, Nicolas Le Roux, and Francis Bach. Tracking the gradients using the hessian: A new look at variance reducing stochastic methods. *arXiv preprint arXiv:1710.07462*, 2017.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[14] Thomas Hofmann, Aurelien Lucchi, Simon Lacoste-Julien, and Brian McWilliams. Variance reduced stochastic gradient descent with neighbors. In *Advances in Neural Information Processing Systems*, pages 2305–2313, 2015.

[15] Prateek Jain, Sham M Kakade, Rahul Kidambi, Praneeth Netrapalli, and Aaron Sidford. Accelerating stochastic gradient descent for least squares regression. In *Conference On Learning Theory*, pages 545–604, 2018.

[16] Prateek Jain, Sham M. Kakade, Rahul Kidambi, Praneeth Netrapalli, and Aaron Sidford. Parallelizing stochastic gradient descent for least squares regression: Mini-batching, averaging, and model misspecification. *Journal of Machine Learning Research*, 18(223):1–42, 2018.

[17] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.

[18] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[20] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[21] Huibert Kwakernaak. *Linear optimal control systems*, volume 1.

[22] Simon Lacoste-Julien, Mark Schmidt, and Francis Bach. A simpler approach to obtaining an o (1/t) convergence rate for the projected stochastic subgradient method. *arXiv preprint arXiv:1212.2002*, 2012.

[23] Nicolas Le Roux. Anytime tail averaging. *arXiv preprint arXiv:1902.05083*, 2019.

[24] Nicolas Le Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in Neural Information Processing Systems*, pages 2663–2671, 2012.

[25] Yann LeCun and Corinna Cortes. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2010.

[26] Nicolas Loizou and Peter Richtárik. Momentum and stochastic momentum for stochastic gradient, newton, proximal point and subspace descent methods. *arXiv preprint arXiv:1712.09677*, 2017.

[27] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[28] Julien Mairal. Optimization with first-order surrogate functions. In *Proceedings of The 30th International Conference on Machine Learning*, pages 783–791, 2013.

[29] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub W. Pachocki, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *CoRR*, abs/1808.00177, 2018.

[30] Brendan O'Donoghue and Emmanuel Candes. Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics*, 15(3):715–732, 2015.

[31] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

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

[33] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[34] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

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

[36] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[37] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[38] Mark Schmidt. Convergence rate of stochastic gradient with constant step size. 2014.

[39] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Convergence rates of inexact proximal-gradient methods for convex optimization. In *Advances in Neural Information Processing Systems 24*, 2011.

[40] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

[41] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015.

[42] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss. *Journal of Machine Learning Research*, 14(1):567–599, February 2013.

[43] Christopher J Shallue, Jaehoon Lee, Joe Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E Dahl. Measuring the effects of data parallelism on neural network training. *arXiv preprint arXiv:1811.03600*, 2018.

[44] 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.

[45] Sharan Vaswani, Francis Bach, and Mark Schmidt. Fast and faster convergence of sgd for over-parameterized models and an accelerated perceptron. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, 2019.

[46] Hoi-To Wai, Wei Shi, Angelia Nedic, and Anna Scaglione. Curvature-aided incremental aggregated gradient method. *arXiv preprint arXiv:1710.08936*, 2017.

[47] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[48] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *arXiv preprint arXiv:1705.08292*, 2017.

[49] Jian Zhang and Ioannis Mitliagkas. Yellowfin and the art of momentum tuning. In *SysML*, 2019.

[50] Lijun Zhang, Mehrdad Mahdavi, and Rong Jin. Linear convergence with condition number independent access of full gradients. In *Advances in Neural Information Processing Systems*, pages 980–988, 2013.

A Experimental Details

This section provides additional information regarding the experiments included in the main text.

For each experimental setting we strive to follow the *reproducibility checklist*, and provide:

- a description and citation of the dataset,
- a description of pre-processing steps,
- training / validation / testing splits,
- a description of the hyper-parameter search process and chosen values for each method,
- the exact number of evaluation runs,
- a clear definition of statistics reported, and
- a description of the computing infrastructure.

A.1 CIFAR10 image classification

Dataset The CIFAR10 dataset [20] consists 50k training and 10k testing images, partitioned over 10 classes. We download and pre-process the images using the TensorFlow `models` package, available at the following URL: `https://github.com/tensorflow/models`

Model We use a residual convolutional network [12] with 56 layers as defined in the `models` package. Specifically, we use the second version whose blocks are built as a batch normalization, then a ReLU activation, and then a convolutional layer. [13]

Hyper-parameters We use the exact setup from `https://github.com/tensorflow/models/officials/resnet`. As such, training is carried out with minibatches of 128 examples for 182 epochs and the training data is augmented with random crops and horizontal flips. Also note this setup multiplies the step size by the size of the minibatch. One deviation from the setup is our use of a linearly decaying learning rate instead of an explicit schedule. The linearly decaying learning rate schedule is simple and was shown to perform well [43]. This schedule is specified using two parameters: the decay rate, a multiplier specifying the final step size (0.1 or 0.01), and the decay step, specifying the step at which the fully decayed rate is reached (always set to 90% of the training steps). To factor in ease of tuning[48] we used Adam's default parameter values and a value of 0.9 for HB's parameter. We used IGT with the exponential Anytime Tail Averaging approach. For the tail fraction, we tried two values: the number of epochs and a tenth of that number (180 and 18). We ran using the following learning rate: (1e0, 3e-1, 1e-1, 3e-2, 1e-2) for SGD, HB and the IGT variants and (1e-2, 3e-3, 1e-3, 3e-4, 1e-4) for Adam. We ran a grid search over the base learning rate and its decay rate with a single run per combination. For each optimizer we selected the hyperparameter combination that is fastest to reach a consistently attainable target train loss of 0.2 [43]. Note that selecting the hyperparameter combination reaching the lowest training loss yields qualitatively identical curves.

The resulting hyper-parameters are:

- SGD stepsize 0.3, decay 0.01
- HB stepsize 0.03, decay 0.01
- Adam stepsize 0.001, decay 0.01
- ITA stepsize 0.3, decay 0.01, tail fraction 18
- HB-ITA stepsize 0.03, decay 0.1, tail fraction 18

Infrastructure and Runs The experiments were run using P100 GPUs (single GPU).

Additional Results We provide all learning curves for the methods comparison presented in the main manuscript in figure 6.

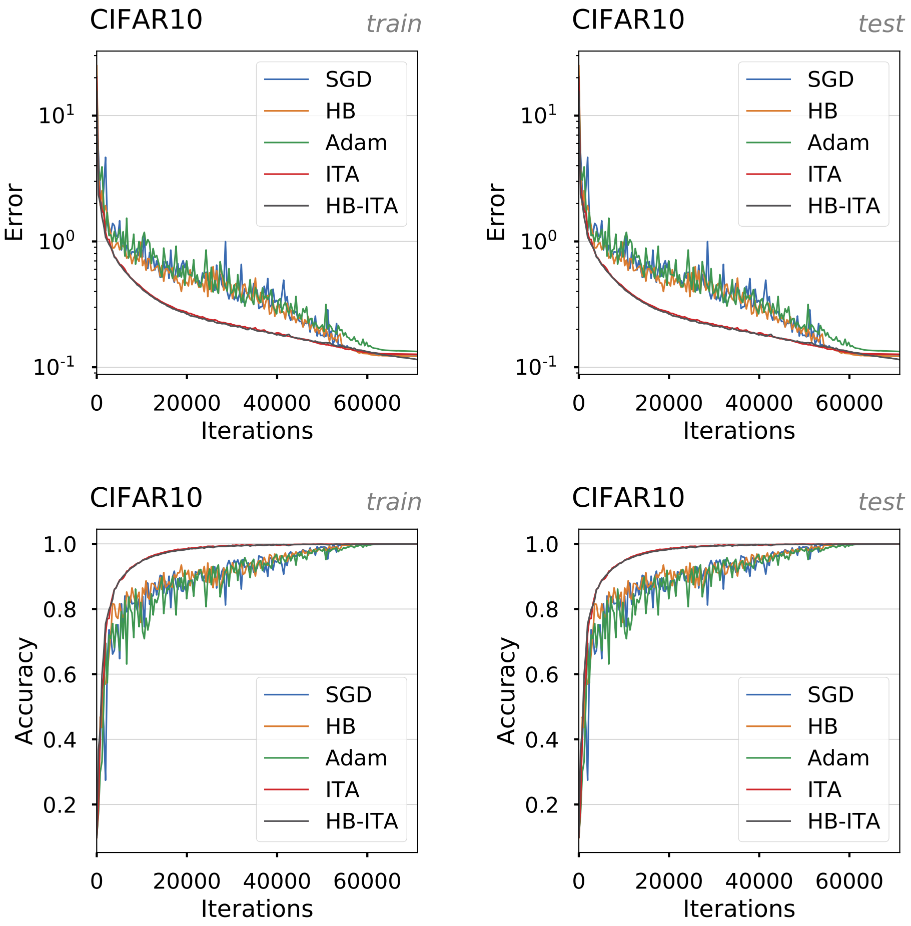


Figure 6: Convergence and accuracy curves along training for the CIFAR10 experiments comparing baseline methods to ours. **Left**: Training. **Right**: Testing.

Ablation study: importance of IGT correction We confirm the importance of the implicit gradient transport correction by running an experiment in which an increasing momentum is used without transport. The results appear in figure 7.

The resulting hyper-parameters are:

- ATA stepsize 0.3, decay 0.01, tail fraction 18

- HB-ATA stepsize 0.03, decay 0.01, tail fraction 18

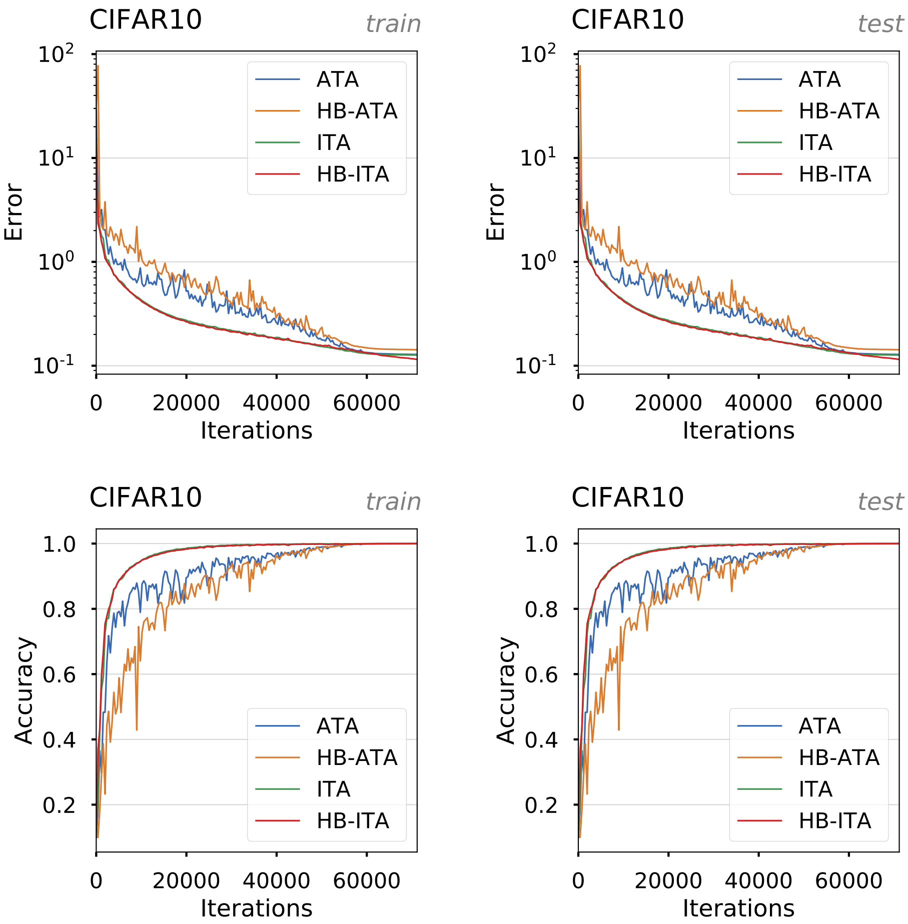


Figure 7: Convergence and accuracy curves along training for the CIFAR10 experiments comparing the use of ATA combined with our proposed implicit transport mechanism. **Left**: Training. **Right**: Testing.

Effect of the batch size We look into the effect of the batch size. To do so, we plot the number of steps required to reach a reliably attainable training loss of 0.4 as a function of the batch size. We ran using the following mini-batch sizes: 32, 128, 512 and 2048. Running with larger minibatches led to out of memory errors on our single GPU setup. The results presented in figure 8 show the benefit of IGT lowers as the batch size increases. Note that Adam's ability to keep benefiting from larger batch sizes is consistent with previous observations.

A.2 ImageNet image classification

Dataset We use the 2015 edition of the *ImageNet Large-Scale Visual Recognition Challenge* (ImageNet) [36] dataset. This dataset consists of 1.2M images partitioned into 1'000 classes. We use the pre-processing and loading utilities of the TensorFlow `models` package, available at the following URL: `https://github.com/tensorflow/models`

Model Our model is again a large residual network, consisting of 50 layers. Similar to our CIFAR10 experiments above, we use the implementation described in [13].

Hyper-parameters We used the same setup and approach as for the CIFAR-10 experiments. The setup trains for 90 epochs using mini-batches of 1024 examples. We used a larger grid for the learning rate schedule: decay (0.1, 0.01, 0.001) and decay step fraction (0.7, 0.8, 0.9).

The resulting hyper-parameters are:

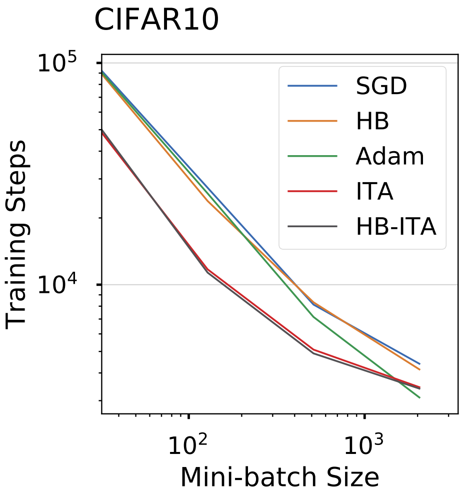


Figure 8: Effect of mini-batch size on the number of steps to reach a target training loss.

- SGD stepsize 0.3, decay 0.01, decay step 0.8
- HB stepsize 0.03, decay 0.001, decay step 0.9
- Adam stepsize 0.0001, decay 0.01, decay step 0.9
- ITA stepsize 0.3, decay 0.01, tail fraction 90, decay step 0.9
- HB-ITA stepsize 0.03, decay 0.01, tail fraction 90, decay step 0.9

Infrastructure and Runs We ran these experiments using Google TPUv2.

Additional Results We include error and accuracy curves for training and testing in Figure 9.

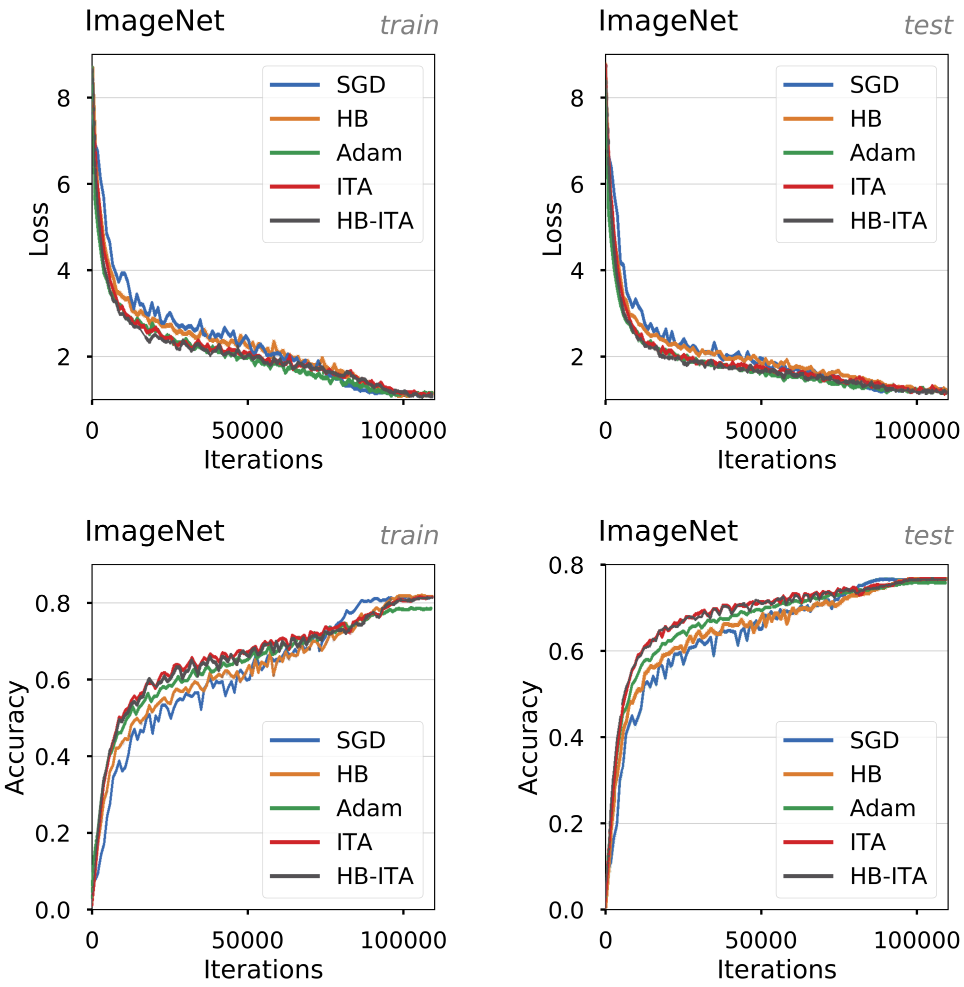


Figure 9: Convergence and accuracy curves along training for the ImageNet experiments. **Left**: Training. **Right**: Testing.

A.3 IMDb sentiment analysis

Dataset The *Internet Movie Database* (IMDb) [27] consists of 25'000 training and 25'000 test movie reviews. The objective is binary sentiment classification based on the review's text. We randomly split the training set in two folds of 17'536 and 7'552 reviews, the former being used for training and the latter for testing. The data is downloaded, splitted, and pre-processed with `torchtext` package, available at the following URL: `https://github.com/pytorch/text` More specifically, we tokenize the text at the word-level using the `spaCy` package, and embed the tokens using the 100-dimensional GloVe 6B [31] distributed representations.

Model The model consists of an embedding lookup-table, followed by a bi-directional LSTM with dropout, and then by a fully-connected layer. The LSTM uses 256 hidden units and the dropout rate is set to 0.5. The whole model consists of 3.2M trainable parameters, with the embedding lookup-table initilized with the GloVe vectors. The model is trained to minimize the `BCEWithLogitsLoss` with a mini-batch size of 64.

Hyper-parameters For each method, we used a grid-search to find the stepsize minimizing validation error after 15 epochs. The grid starts at 0.00025 and doubles until reaching 0.1024, so as to ensure that no chosen value lies on its boundaries. When applicable, the momentum factor is jointly optimized over values 0.1 to 0.95. The final hyper-parameters are displayed in the following table for each method.

Table 1: Hyperparameters for IMDb experiments.

	HB	Adam	ASGD	HB-IGT	HB-ITA
α	0.032	0.0005	0.064	0.128	0.064
μ	0.95	0.95	n/a	0.9	0.9
ξ	n/a	n/a	100	n/a	n/a
κ	n/a	n/a	10^5	n/a	n/a

Infrastructure and Runs All IMDb experiments use a single NVIDIA GTX 1080, with PyTorch v0.3.1.post2, CUDA 8.0, and cuDNN v7.0.5. We run each final configurations with 5 different random seeds and always report the mean tendency ± one standard deviation. Each run lasts approximately three hours and thirty minutes.

Additional Results In addition to the results reported in the main text, we include training, validation, and testing curves for each method in Figure 10. Shading indicates the one standard deviation interval. Note that our focus is explicitly on optimization: in the specific case of IMDb, training for 200 epochs is completely unnecessary from a generalization standpoint as performance degrades rapidly after 15-20 epochs.

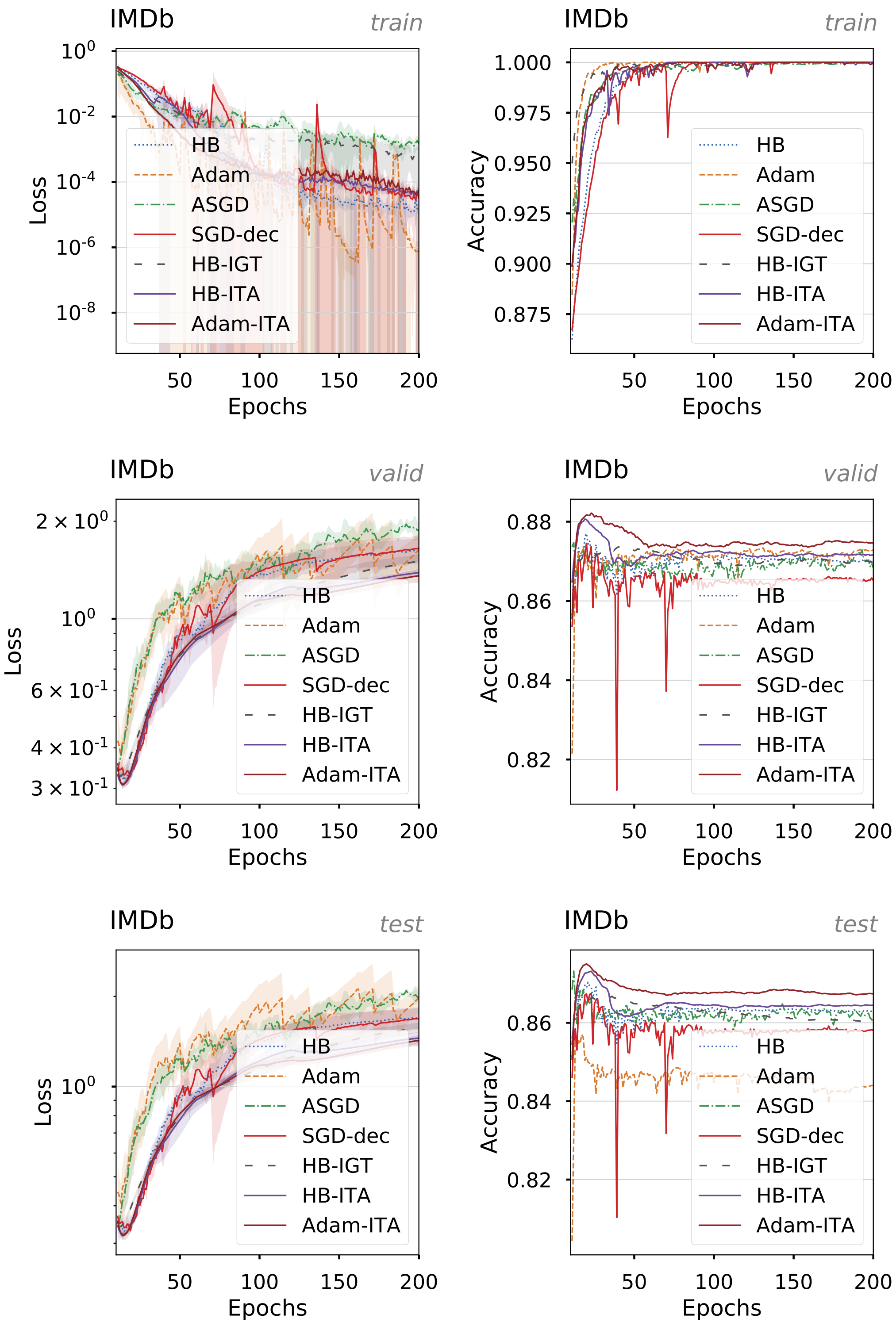


Figure 10: Convergence and accuracy curves along training for the IMDb experiments. **Left**: Convergence. **Right**: Accuracy.

A.4 Linear-quadratic regulator

Setup Our linear-quadratic regulator [21] implements the following equations. The cost functional is evaluated at every timestep h and is given by

$$C(s_h, a_h) = s_h^\top Q s_h + a_h^\top R a_h, \tag{2}$$

for random symmetric positive definite matrices $Q \in \mathbb{R}^{20\times 20}$ and $R \in \mathbb{R}^{12\times 12}$ each with condition number 3. The initial state $s_0 \sim \mathcal{N}(0, 3 \cdot I_{20})$ is sampled around the origin, and the subsequent states evolve according to

$$s_{h+1} = A s_h + B a_h, \tag{3}$$

where entries of $A \in \mathbb{R}^{20\times 20}, B \in \mathbb{R}^{20\times 12}$ are independently sampled from a Normal distribution and then scaled such that the matrix has unit Frobenius norm. The actions are given by the linear stochastic policy $a_h = K s_h + \epsilon_h^a$, where $\epsilon_h^a \sim \mathcal{N}(0, I)$ and K are the parameters to be optimized.

Gradient methods in this manuscript optimize the sum of costs using the REINFORCE estimate [47] given by

$$\nabla_K \mathbb{E} \sum_h^{10} C(s_h, a_h) = \mathbb{E} \left(\sum_h^{10} \nabla_K \log \pi_K(a_h | s_h) \right) \left(\sum_h^{10} C(s_h, a_h) \right). \tag{4}$$

In our experiments, the above expectation is approximated by the average of 600 trajectory rollouts. Due to the noisy dynamics of the system, it is possible for the gradient norm to explode leading to numerical unstabilities – especially when using larger stepsizes. To remedy this issue, we simply discard such problematic trajectories from the gradient estimator.

For each training iteration, we first gather 600 trajectories used for learning and then 600 more used to report evaluation metrics.

Hyper-parameters Due to the simplicity of the considered methods, the only hyper-parameter is the stepsize. For each method, we choose the stepsize from a logarithmically-spaced grid so as to minimize the evaluation cost after 600 iterations on a single seed. Incidentally, the optimal stepsize for GD, SGD, and ITA is 0.0002.

Infrastructure and Runs We use an Intel Core i7-5820K CPU to run the LQR experiments. All methods are implemented using `numpy` v1.15.4. We present results averaged over 3 random seeds, and also report the standard deviation. For stochastic gradient methods (SGD, ITA) training for 20K iterations takes about 3h, for full-gradient method (GD) about 10h, and computing the solution of the Riccati equation takes less than 5 seconds.

Additional Results In addition to the evaluation cost reported in the main text, we also include the cost witnessed during training (and used for optimization) in Figure 11.

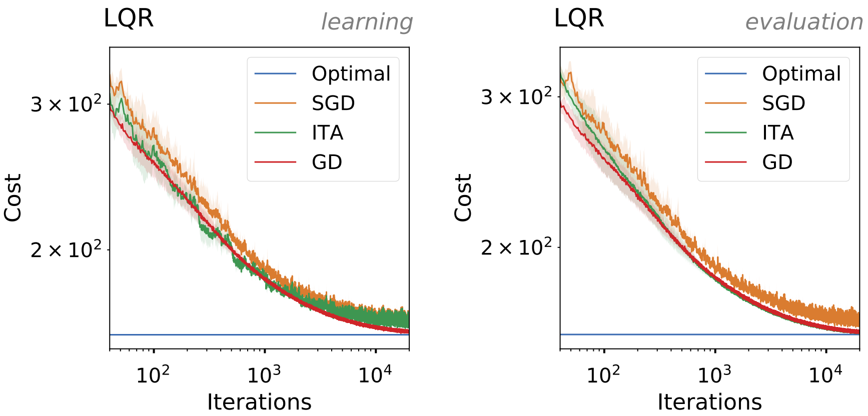


Figure 11: LQR costs along training iterations. **Left**: Costs used for learning. **Right**: Costs used for evaluation.

We notice that the training cost curve of ITA is not as smooth as the evaluation one. Similarly, the observed learning costs never reach as good a minima as the evaluation ones. This phenomena is easily clarified: during learning, ITA esimates the gradient using the shifted parameters $K_t + \frac{\gamma_t}{1-\gamma_t}(K_t - K_{t-1})$ as opposed to the true parameters K_t. Those shifted parameters are not subject to a reduced variance, hence explaining the observed noise in the cost as well as deteriorated convergence.

A.5 Model-agnostic meta-learning

Dataset We use the Mini-Imagenet dataset [34] in our model-agnostic meta-learning (MAML) [9] experiments. This dataset comprises 64 training, 12 validation, and 24 test classes. For each of train, validation, and test sets, tasks are constructed by sampling 5 classes from their respective split, and further sampling 5 images per class. Images are downsampled to 84x84x3 tensors of RGB values. For more details, please refer to the official code repository – which we carefully replicated – at the following URL: `https://github.com/cbfinn/maml`

Our implementation departs in two ways from the reference. First, we train our models for 100k iterations as opposed to 60k and only use 5 image samples to compute a meta-gradient whereas the reference implementation uses 15. Second, we only use 5 adaptation steps at evaluation time, when the reference uses 10.

Model The model closely replicates the convolutional neural network of MAML [9]. It consists of 4 layers, each with 32 3x3 kernels, followed by batch normalization and ReLU activations. For specific implementation details, we refer the reader to the above reference implementation.

Hyper-parameters We only tune the meta-stepsize for the MAML experiment. We set the momentum constant to 0.9, the adaptation-stepsize to 0.01, and average the meta-gradient of 4 tasks per iterations. Due to the reduced variance in the gradients, we found it necessary to increase the ϵ of Adam-ITA to 0.01.

For each method, we search over stepsize values on a logarithmically-spaced grid and select those values that maximize validation accuracy after 10k meta-iterations. These values are reported in Table 2.

	HB	Adam	HB-ITA	Adam-ITA
α	0.008	0.001	0.008	0.0005

Table 2: Stepsizes for MAML experiments.

Infrastructure and Runs Each MAML experiment is run on a single NVIDIA GTX TITAN X, with PyTorch v1.1.0, CUDA 8.0, and cuDNN v7.0.5. We run each configuration with 3 different random seeds and report the mean tendency ± one standard deviation. Each run takes approximately 36 hours, and we evaluate the validation and testing accuracy every 100 iteration.

Additional Results We complete the MAML validation curves from the main manuscript with training and testing accuracy curves in Figure 12. Moreover, we recall the final test accuracies for each method: Adam-ITA reaches 65.16%, HB-ITA 64.57%, Adam 63.70%, and HB 63.08%.

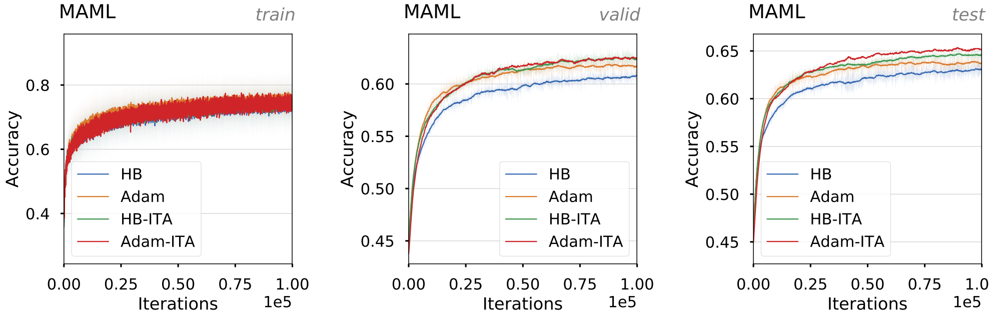


Figure 12: Training, validation, and testing accuracies for the MAML experiments along training. Shading indicates the 1 standard deviation interval. **Left**: Training. **Center**: Validation. **Right**: Testing.

B Additional Experiments

This section presents additional experiments to the ones included in the main text.

B.1 Baselines comparisons

While experiments in Section 5 highlighted properties of IGT and HB-IGT when the assumption of identical, constant Hessians was verified, we now turn to more realistic scenarios where individual functions are neither quadratic nor have the same Hessian to compare our proposed methods against popular baselines for the online stochastic optimization setting. We target optimization benchmarks and focus on training loss minimization.

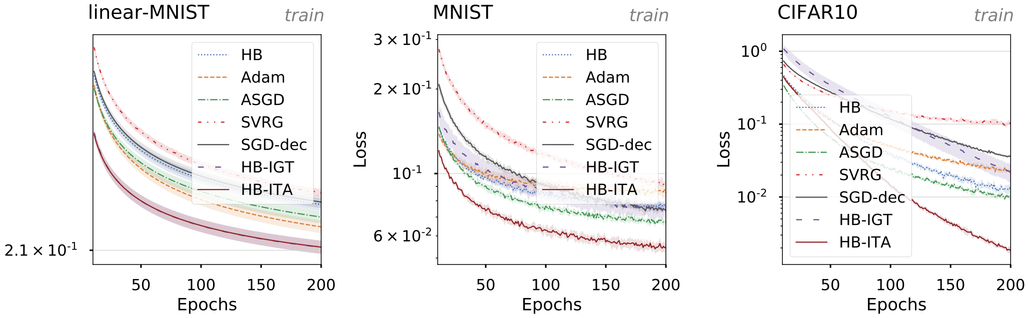


Figure 13: Training loss curves for different optimization algorithms on several popular benchmarks. For each method, the hyper-parameters are tuned to minimize the training error after 15 epochs. Algorithms using the IGT gradient estimates tend to outperform their stochastic gradient counter-parts. **Left**: Logistic regression on MNIST. **Center**: LeNet5 on MNIST. **Right**: MobileNetv2 on CIFAR10.

We investigate three different scenarios: (a) **linear-mnist**: a logistic regression model on MNIST, (b) **mnist**: a modified version of LeNet5 [25] on MNIST and (c) **cifar-small**: the MobileNetv2

	Linear-MNIST	MNIST	CIFAR10	IMDb
Heavyball	92.52 ± 0.04	99.08 ± 0.07	91.55 ± 0.25	86.90 ± 0.67
Adam	92.57 ± 0.10	98.99 ± 0.05	89.36 ± 0.75	85.62 ± 0.63
ASGD	92.47 ± 0.08	99.15 ± 0.07	91.45 ± 0.20	87.31 ± 0.21
SVRG	92.51 ± 0.04	99.06 ± 0.08	86.84 ± 0.17	n/a
SGD-dec	92.52 ± 0.06	99.11 ± 0.06	87.53 ± 0.23	86.73 ± 0.34
Heavyball-IGT	92.47 ± 0.10	99.00 ± 0.05	12.05 ± 0.21	86.61 ± 0.23
Heavyball-ITA	92.50 ± 0.10	99.19 ± 0.02	90.37 ± 0.31	87.26 ± 0.24

Table 3: Test accuracies from the best validation epoch.

architecture [37] consisting of 19 convolutional layers on CIFAR10. All models are trained with a mini-batch size of 64, while the remaining hyper-parameters are available in Tables 4, 5, and 6.

For each of the above tasks, models are trained for 200 epochs. We compare the following methods:

- **HB**: the heavy ball method [33],
- **Adam** [18],
- **ASGD** [15],
- **SVRG** [17],
- **SGD-dec**: stochastic gradient method with an exponential learning rate schedule and exponential constant 0.999,
- **HB-IGT**: the heavy ball using the IGT as a plug-in estimator, and
- **HB-ITA**: same as HB-IGT but using the anytime tail averaging to forget the oldest gradients.

The hyperparameters of each method, and in particular the stepsize, are tuned independently according to a logarithmic grid so as to minimize the mean training error after epoch 15 on one seed. We then use those parameters on 5 random seeds and report the mean and standard deviation of the performance.

Figure 13 shows the training curves for the five algorithms in the three settings.

First, we observe that, for the logistic regression, HB-IGT performs on par with HB-ITA and far better than all the other methods, even though the assumption on the Hessians is violated. When using a ConvNet, however, we see that HB-IGT is not competitive with state-of-the-art methods such as Adam or ASGD. HB-ITA, on the other hand, with its smaller reliance on the assumption, once again performs much better than all other methods. In fact, HB-ITA not only converges to a lower final train error but also has a faster initial rate.

While our focus is on optimization, we also report generalization metrics in Table 3. For each algorithm, we computed the best mean accuracy after each epoch on the test set and report this value together with its standard deviation. The importance of the Anytime Tail-Averaging mechanism is again apparent: without it, Heavyball-IGT is unable to improve for more than a few epochs on the CIFAR10 validation set, regardless of the stepsize choice. On the other hand, it is evident from those results that the solutions found by Heavyball-ITA are competitive with the ones discovered by other optimization algorithms.

	HB	Adam	ASGD	HB-IGT	HB-ITA
α	0.0128	0.0002	0.0032	0.0032	0.0016
μ	0.1	0.95	n/a	0.9	0.1
ξ	n/a	n/a	10	n/a	n/a
κ	n/a	n/a	10^4	n/a	n/a

Table 4: Hyperparameters for linear-MNIST experiments.

	HB	Adam	ASGD	HB-IGT	HB-ITA
α	0.0064	0.0016	0.0128	0.0032	0.0032
μ	0.9	0.95	n/a	0.95	0.95
ξ	n/a	n/a	10	n/a	n/a
κ	n/a	n/a	10^4	n/a	n/a

Table 5: Hyperparameters for MNIST experiments.

	HB	Adam	ASGD	HB-IGT	HB-ITA
α	0.0512	0.0512	0.1024	0.0128	0.0512
μ	0.95	0.9	n/a	0.9	0.1
ξ	n/a	n/a	100	n/a	n/a
κ	n/a	n/a	10^5	n/a	n/a

Table 6: Hyperparameters for MobileNetV2 on CIFAR10 experiments.

C Proofs

C.1 Transport formula

$$\begin{aligned}
g_t(\theta_t) &= \frac{t}{t+1} g_{t-1}(\theta_t) + \frac{1}{t+1} g(\theta_t, x_t) \\
&= \frac{t}{t+1}\left(g_{t-1}(\theta_{t-1}) + H(\theta_t - \theta_{t-1})\right) + \frac{1}{t+1} g(\theta_t, x_t) && \text{(Quadratic } f\text{)} \\
&= \frac{t}{t+1} g_{t-1}(\theta_{t-1}) + \frac{1}{t+1}\left(g(\theta_t, x_t) + tH(\theta_t - \theta_{t-1})\right) \\
&= \frac{t}{t+1} g_{t-1}(\theta_{t-1}) + \frac{1}{t+1} g(\theta_t + t(\theta_t - \theta_{t-1}), x_t) && \text{(Identical Hessians)} \\
&\approx \frac{t}{t+1} \widehat{g}_{t-1}(\theta_{t-1}) + \frac{1}{t+1} g(\theta_t + t(\theta_t - \theta_{t-1}), x_t)\ . && (\widehat{g}_{t-1} \text{ is an approximation})
\end{aligned}$$

D Proof of Prop. 3.1

In this proof, we assume that g is a strongly-convex quadratic function with hessian H.

At timestep t, we have access to a stochastic gradient $g(\theta, x_t) = g(\theta_t) + \epsilon_t$ where the ϵ_t are i.i.d. with variance $C \preceq \sigma^2 H$.

We first prove a simple lemma:

Lemma D.1. *If* $v_0 = g(\theta_0) + \epsilon_0$ *and, for* $t > 0$, *we have*

$$v_t = \frac{t}{t+1} v_{t-1} + \frac{1}{t+1} g\left(\theta_t + t(\theta_t - \theta_{t-1})\right) + \frac{1}{t+1}\epsilon_t\ ,$$

then

$$v_t = g(\theta_t) + \frac{1}{t+1}\sum_{i=0}^{t}\epsilon_i\ .$$

Proof. Per our assumption, this is true for $t = 0$. Now let us prove the result by induction. Assume this is true for $t - 1$. Then we have:

$$\begin{aligned}
v_t &= \frac{t}{t+1} v_{t-1} + \frac{1}{t+1} g(\theta_t + t(\theta_t - \theta_{t-1})) + \frac{1}{t+1}\epsilon_t \\
&= \frac{t}{t+1} g(\theta_{t-1}) + \frac{1}{t+1}\sum_{i=0}^{t-1}\epsilon_i \\
&\quad + \frac{1}{t+1} g(\theta_t + t(\theta_t - \theta_{t-1})) + \frac{1}{t+1}\epsilon_t && \text{(recurrence assumption)} \\
&= \frac{t}{t+1} g(\theta_{t-1}) + \frac{1}{t+1}\sum_{i=0}^{t-1}\epsilon_i \\
&\quad + g(\theta_t) - \frac{t}{t+1} g(\theta_{t-1}) + \frac{1}{t+1}\epsilon_t && \text{(g is quadratic)} \\
&= g(\theta_t) + \frac{1}{t+1}\sum_{i=0}^{t}\epsilon_i \ .
\end{aligned}$$

This concludes the proof. □

Lemma D.2. *Let us assume we perform the following iterative updates:*

$$\begin{aligned}
v_t &= \frac{t}{t+1} v_{t-1} + \frac{1}{t+1} g(\theta_t + t(\theta_t - \theta_{t-1})) + \frac{1}{t+1}\epsilon_t \\
\theta_{t+1} &= \theta_t - \alpha v_t \ ,
\end{aligned}$$

starting from $v_0 = g(\theta_0) + \epsilon_0$. Then, denoting $\Delta_t = \theta_t - \theta^*$, we have

$$\Delta_t = (I - \alpha H)^t \Delta_0 - \alpha \sum_{i=0}^{t-1} N_{i,t}\epsilon_i$$

with

$$\begin{aligned}
N_{i,0} &= 0 \\
N_{i,t} &= (I - \alpha H) N_{i,t-1} + 1_{i<t}\frac{1}{t} I \ .
\end{aligned}$$

Proof. The result is true for $t = 0$. We now prove the result for all t by induction. Let us assume this is true for $t - 1$. Using Lemma D.1, we have

$$v_{t-1} = g(\theta_{t-1}) + \frac{1}{t}\sum_{i=0}^{t-1}\epsilon_i$$

and thus, using $g(\theta_{t-1}) = H\Delta_{t-1}$,

$$\begin{aligned}
\Delta_t &= \Delta_{t-1} - \alpha v_{t-1} \\
&= \Delta_{t-1} - \alpha H \Delta_{t-1} - \frac{\alpha}{t}\sum_{i=0}^{t-1}\epsilon_i \\
&= (I - \alpha H)\Delta_{t-1} - \frac{\alpha}{t}\sum_{i=0}^{t-1}\epsilon_i \\
&= (I - \alpha H)^t \Delta_0 - \alpha \sum_{i=0}^{t-2} (I - \alpha H) N_{i,t-1}\epsilon_i - \frac{\alpha}{t}\sum_{i=0}^{t-1}\epsilon_i && \text{(recurrence assumption)} \\
&= (I - \alpha H)^t \Delta_0 - \alpha \sum_{i=0}^{t-1} N_{i,t}\epsilon_i
\end{aligned}$$

with

$$N_{i,t} = (I - \alpha H) N_{i,t-1} + 1_{i<t} \frac{1}{t} I \ .$$

This concludes the proof. □

For the following lemma, we will assume that the Hessian is diagonal and will focus on one dimension with eigenvalue h. Indeed, we know that there are no interactions between the eigenspaces and that we can analyze each of them independently [30].

Lemma D.3. *Denote $r_h = 1 - \alpha h$. We assume $\alpha \leq \frac{1}{L}$. Then, for any i and any t, we have*

$$\begin{aligned}
&N_{i,t} \geq 0 && \text{(Positivity)}\\
&N_{i,t} = 0 \quad \textit{if}\ t \leq i && \text{(Zero-start)}\\
&N_{i,t} \leq \log\left(\frac{2}{i(1-r_h)}\right) \qquad \textit{if}\ i < t \leq \frac{2}{1-r_h)} && \text{(Constant bound)}\\
&N_{i,t} \leq \frac{\max\left\{1 + r_h, 2\log\left(\frac{2}{i(1-r_h)}\right)\right\}}{t(1-r_h)} \qquad \textit{if}\ \frac{2}{1-r_h} \leq t \ . && \text{(Decreasing bound)}
\end{aligned}$$

Proof. The Zero-start case $i \geq t$ is immediate from the recursion of Lemma D.2. The Positivity property of $N_{i,t}$ is also immediate from the recursion since the stepsize α is such that $r_h = 1 - \alpha h$ is positive.

We now turn to the Constant bound property. We have, for $t > i$,

$$\begin{aligned}
N_{i,t} &= r_h N_{i,t-1} + \frac{1}{t}\\
&\leq N_{i,t-1} + \frac{1}{t} \ .
\end{aligned}$$

Thus, $N_{i,t} - N_{i,t-1} \leq \frac{1}{t}$. Summing these inequalities, we get a telescopic sum and, finally:

$$\begin{aligned}
N_{i,t} &\leq \sum_{j=i+1}^{t} \frac{1}{j}\\
&\leq \int_{x=i}^{t} \frac{dx}{x}\\
&= \log\left(\frac{t}{i}\right) \ .
\end{aligned}$$

This bound is trivial in the case $i = 0$. In that case, we keep the first term in the sum separate and get

$$N_{0,t} \leq 1 + \log t \ .$$

In the remainder, we shall keep the $\log\left(\frac{t}{i}\right)$ bound for simplicity. The upper bound on the right-hand size is increasing with t and its value for $t = \frac{2}{1-r_h}$ is thus an upper bound for all smaller values of t. Replacing t with $\frac{2}{1-r_h}$ leads to

$$\begin{aligned}
N_{i,\frac{2}{1-r_h}} &\leq \log\left(\frac{\frac{2}{1-r_h}}{i}\right)\\
&= \log\left(\frac{2}{i(1-r_h)}\right) \ .
\end{aligned}$$

This proves the third inequality.

We shall now prove the Decreasing bound by induction. This bound states that, for t large enough, each $N_{i,t}$ decreases as $O(1/t)$. Using the second and third inequalities, we have

$$\begin{aligned}
N_{i,\frac{2}{1-r_h}} &\leq \log\left(\frac{2}{i(1-r_h)}\right)\frac{\frac{2}{1-r_h}}{\frac{2}{1-r_h}} \\
&= \frac{\log\left(\frac{2}{i(1-r_h)}\right)}{1-r_h}\frac{2}{\frac{2}{1-r_h}} \\
&\leq \frac{\max\left\{1+r_h, 2\log\left(\frac{2}{i(1-r_h)}\right)\right\}}{\frac{2}{1-r_h}(1-r_h)} \; .
\end{aligned}$$

The maximum will help us prove the last property. Thus, for $t = \frac{2}{1-r_h}$, we have

$$\begin{aligned}
N_{i,t} &\leq \frac{\max\left\{1+r_h, 2\log\left(\frac{1}{i(1-r_h)}\right)\right\}}{t(1-r_h)} \\
&\leq \frac{\nu_i}{t} \; ,
\end{aligned}$$

with $\nu_i = \frac{\max\left\{1+r_h, 2\log\left(\frac{1}{i(1-r_h)}\right)\right\}}{(1-r_h)}$. The Decreasing bound is verified for $t = \frac{2}{1-r_h}$.

We now show that if, for any $t > \frac{2}{1-r_h}$, we have $N_{i,t-1} \leq \frac{\nu_i}{t-1}$, then $N_{i,t} \leq \frac{\nu_i}{t}$. Assume that there is such at t. Then

$$\begin{aligned}
N_{i,t} &= r_h N_{i,t-1} + \frac{1}{t} \\
&\leq \frac{r_h \nu_i}{t-1} + \frac{1}{t} \\
&= \frac{r_h t \nu_i + t - 1}{t(t-1)} \\
&= \frac{(t-1)\nu_i + (r_h - 1)t\nu_i + \nu_i + t - 1}{t(t-1)} \\
&= \frac{\nu_i}{t} + \frac{(r_h - 1)t\nu_i + \nu_i + t - 1}{t(t-1)} \; .
\end{aligned}$$

We now shall prove that $(r_h - 1)t\nu_i + \nu_i + t - 1 = [(r_h - 1)\nu_i + 1]t + \nu_i - 1$ is negative. First, we have that

$$\begin{aligned}
(r_h - 1)\nu_i + 1 &= 1 - \max\left\{1+r_h, 2\log\left(\frac{1}{i(1-r_h)}\right)\right\} \\
&\leq 0 \; .
\end{aligned}$$

Then,

$$[(r_h - 1)\nu_i + 1]t + \nu_i - 1 \leq 0 \Longleftrightarrow t \geq \frac{\nu_i - 1}{(1-r_h)\nu_i - 1}$$

since $(r_h - 1)\nu_i + 1 \leq 0$. Thus, the property is true for every $t \geq \frac{\nu_i - 1}{(1-r_h)\nu_i - 1}$. In addition, we have

$$\begin{aligned}
\nu_i &\geq \frac{1+r_h}{1-r_h} \\
\nu_i(1-r_h) &\geq 1 + r_h \\
2\nu_i(1-r_h) - 2 &\geq \nu_i(1-r_h) - 1 + r_h \\
\frac{2}{1-r_h} &\geq \frac{\nu_i - 1}{\nu_i(1-r_h) - 1} \; ,
\end{aligned}$$

and the property is also true for every $t \geq \frac{2}{1-r_h}$. This concludes the proof. □

Finally, we can prove the Proposition 3.1:

Proof. The expectation of Δ_t is immediate using Lemma D.2 and the fact that the ϵ_i are independent, zero-mean noises. The variance is equal to $V[\Delta_t] = \alpha^2 B \sum_{i=0}^t N_{i,t}^2$. While our analysis was only along one eigenspace of the Hessian with associated eigenvalue h, we must now sum over all dimensions. We will thus define

$$\begin{aligned}\bar{\nu}_i &= \frac{\max\left\{2 - \alpha\mu, 2\log\left(\frac{1}{i\alpha\mu}\right)\right\}}{\alpha\mu} \qquad \text{for } i > 0\\ \bar{\nu}_0 &= \frac{2 + 2\log\left(\frac{1}{\alpha\mu}\right)}{\alpha\mu}\ ,\end{aligned}$$

which is, for every i, the maximum ν_i across all dimensions. We get

$$\begin{aligned}V[\Delta_t] &\leq d\alpha^2 B \sum_{i=0}^t \frac{\bar{\nu}_i^2}{t^2}\\ &\leq d\alpha^2 B \sum_{i=0}^t \frac{\bar{\nu}_0^2}{t^2} \qquad \text{since } \nu_i \geq \nu_{i+1}\ \forall i\\ &\leq \frac{d\alpha^2 B \bar{\nu}_0^2}{t}\ .\end{aligned}$$

Since we have

$$E[\theta_t - \theta^*] = (I - \alpha H)^t(\theta_0 - \theta^*)\ ,$$

we get

$$\begin{aligned}E[\|\theta_t - \theta^*\|^2] &= \|E[\theta_t - \theta^*]\|^2 + V[\Delta_t]\\ &\leq (\theta_0 - \theta^*)^\top (I - \alpha H)^{2t}(\theta_0 - \theta^*) + \frac{d\alpha^2 B\bar{\nu}_0^2}{t}\\ &\leq \left(1 - \frac{1}{\kappa}\right)^{2t} \|\theta_0 - \theta^*\|^2 + \frac{d\alpha^2 B \bar{\nu}_0^2}{t}\ .\end{aligned}$$

This concludes the proof. ☐

E Proof of Proposition 3.2 and Proposition 3.3

In this section we list and prove all lemmas used in the proofs of Proposition 3.2 and Proposition 3.3; all lemmas are stated in the same conditions as the proposition.

We start the following proposition:

Proposition E.1. *Let f be a quadratic function with positive definite Hessian H with largest eigenvalue L and condition number κ and if the stochastic gradients satisfy $g(\theta, x) = g(\theta) + \epsilon$ with ϵ a random uncorrelated noise with covariance bounded by BI.*

Then, Algorithm 1 leads to iterates θ_t satisfying

$$E[\theta_t - \theta^*] = \begin{pmatrix} I \\ 0 \end{pmatrix} A^t \begin{pmatrix} E[\theta_1 - \theta^*] \\ E[\theta_0 - \theta^*] \end{pmatrix} \tag{5}$$

where

$$A = \begin{pmatrix} I - \alpha H + \mu I & -\mu I \\ I & 0 \end{pmatrix} \tag{6}$$

governs the dynamics of this bias. In particular, when its spectral radius, $\rho(A)$ is less than 1, the iterates converge linearly to θ^*.

In a similar fashion, the variance dynamics of Heavyball-IGT are governed by the matrix

$$D_i = \begin{pmatrix} (1-\alpha h_i+\mu)^2 + 2\alpha^2 h_i^2 & \mu^2 & -2\mu(1-\alpha h_i+\mu)^2 \\ 1 & 0 & 0 \\ 1-\alpha h_i+\mu & 0 & -\mu \end{pmatrix}$$

If the spectral radius of D_i, $\rho(D_i)$, is strictly less than 1 or all i, then there exist constants $t_0 > 0$ and $C > 0$ for which

$$\mathrm{Var}(\theta_t) \leq 2\alpha^2 dBC\frac{\log(t)}{t}, \quad \textit{for } t > t_0$$

where B is a bound on the variance of noise variables ϵ_i.

Lemma E.2 (IGT estimator as true gradient plus noise average)**.** *If $v_0 = g(\theta_0) + \epsilon_0$ and for $t > 0$ we have*

$$v_t = \frac{t}{t+1}v_{t-1} + \frac{1}{t+1}g(\theta_t + t(\theta_t - \theta_{t-1})) + \frac{1}{t+1}\epsilon_t,$$

then

$$v_t = g(\theta_t) + \frac{1}{t+1}\sum_{i=0}^{t}\epsilon_i.$$

This lemma is already proved in the previous section for the IGT estimator (Lemma D.1) and is just repeated here for completeness. We will use this result in the next few lemmas.

Lemma E.3 (The IGT gradient estimator is unbiased on quadratics)**.** *For the IGT gradient estimator, v_t, corresponding to parameters θ_t we have*

$$\mathbb{E}[v_t] = g(\mathbb{E}\theta_t),$$

where the expectation is over all gradient noise vectors $\epsilon_0, \epsilon_1, \ldots, \epsilon_t$.

Proof. The proof proceeds by induction. The base case holds as we have

$$\mathbb{E}[v_0] = \mathbb{E}[g_0 + \epsilon_0] = g(\theta_0).$$

For the inductive case, we can write

$$\begin{aligned}
\mathbb{E}[v_t] &= \mathbb{E}\left[\frac{t}{t+1}v_{t-1} + \frac{1}{t+1}\hat{g}(\theta_t + t(\theta_t - \theta_{t-1}))\right] \\
&= \mathbb{E}\left[\frac{t}{t+1}v_{t-1} + \frac{1}{t+1}g_t + \frac{t}{t+1}g_t - \frac{t}{t+1}g_{t-1} + \frac{1}{t+1}\epsilon_t\right] \\
&= \frac{t}{t+1}\mathbb{E}[v_{t-1} - g_{t-1}] + \mathbb{E}[g_t] + \frac{t}{t+1}\mathbb{E}[\epsilon_t] \\
&= \mathbb{E}[g_t] = g(\mathbb{E}[\theta_t]).
\end{aligned}$$

Where, in the third equality, $\mathbb{E}[v_{t-1} - g_{t-1}] = 0$ by the inductive assumption, and the last equality because the gradient of a quadratic function is linear. □

Lemma E.4 (Bounding the IGT gradient variance)**.** *Let v_t be the IGT gradient estimator. Then*

$$\mathrm{Var}[v_t] \leq 2h^2\mathrm{Var}[\theta_t - \theta^\star] + \frac{2B}{t},$$

where B is the variance of the homoscedastic noise ϵ_t.

Proof.

$$\begin{aligned}
\mathrm{Var}\left[v_t\right] &= \mathrm{Var}\left[g_t + \frac{1}{t+1}\sum_{i=0}^{t}\epsilon_i\right] \\
&= \mathrm{Var}\left[h\theta_t\right] + \mathrm{Var}\left[\frac{1}{t+1}\sum_{i=0}^{t}\epsilon_i\right] \\
&\quad + 2\mathrm{Cov}\left[h\theta_t, \frac{1}{t+1}\sum_{i=0}^{t}\epsilon_i\right] \\
&\leq 2\mathrm{Var}\left[h\theta_t\right] + 2\mathrm{Var}\left[\frac{1}{t+1}\sum_{i=0}^{t}\epsilon_i\right] \\
&= 2h^2\mathrm{Var}\left[\theta_t - \theta^\star\right] + 2\frac{B}{t}
\end{aligned}$$

□

Now that we have these basic results on the IGT estimator, we can analyze the evolution of the bias and variance of Heavyball-IGT. We use the quadratic assumption to decouple the vector dynamics of Heavyball-IGT into independent scalar dynamics. If the Hessian, H, has eigenvalues $L \geq h_1 \geq h_2 \geq \ldots \geq h_n = L/\kappa$, then we can assume without loss of generality that H is diagonal with $H_{ii} = h_i$.

Lemma E.5 (Evolution of bias for scalar quadratic)**.** *Assume that the Hessian, second derivative, is h.*

Starting with $v_0 = g(\theta_0) + \epsilon_0$ and $w_0 = 0$, performing the following iterative updates (Heavyball-IGT, Algorithm 1):

$$\begin{gathered}
v_t = \frac{t}{t+1}v_{t-1} + \frac{1}{t+1}g(\theta + t(\theta_t - \theta_{t-1})) + \frac{1}{t+1}\epsilon_t, \\
w_{t+1} = \mu w_t + \alpha v_t, \qquad \theta_{t+1} = \theta_t - w_{t+1}
\end{gathered}$$

results in

$$\Delta_t = A^t\Delta_0 - \alpha\sum_{i=0}^{t-1} N_{i,t}\begin{bmatrix}\epsilon_i \\ 0\end{bmatrix}$$

where $N_{j,0} = 0_{2\times 2}$, $\qquad N_{i,t} = AN_{i,t-1} + 1_{i<t}\frac{1}{t}I$,

$\Delta_t = \begin{bmatrix}\theta_t - \theta^* \\ \theta_{t-1} - \theta^*\end{bmatrix}$ *and* $A = \begin{pmatrix}1 - \alpha h + \mu & -\mu \\ 1 & 0\end{pmatrix}$.

Proof. The proof proceeds by induction. First notice that for $t = 0$ the equality naturally holds. We make the inductive assumption that it holds for $t - 1$, and start by using Lemma E.2:

$$\begin{aligned}
\Delta_t &= A\Delta_{t-1} - \frac{\alpha}{t}\sum_{i=0}^{t-1}\begin{bmatrix}\epsilon_i \\ 0\end{bmatrix} \\
&= A(A^{t-1}\Delta_0 - \alpha\sum_{i=0}^{t-2}N_{i,t}\begin{bmatrix}\epsilon_i \\ 0\end{bmatrix}) - \frac{\alpha}{t}\sum_{i=0}^{t-1}\begin{bmatrix}\epsilon_i \\ 0\end{bmatrix} && \text{(Inductive assumption)} \\
&= A^t\Delta_0 - \alpha(\sum_{i=0}^{t-2}AN_{i,t}\begin{bmatrix}\epsilon_i \\ 0\end{bmatrix} + \frac{1}{t}\sum_{i=0}^{t-1}\begin{bmatrix}\epsilon_i \\ 0\end{bmatrix}) \\
&= A^t\Delta_0 - \alpha\sum_{i=0}^{t-1}N_{i,t}\begin{bmatrix}\epsilon_i \\ 0\end{bmatrix} && \text{(Def. of } N_{i,t}\text{)}
\end{aligned}$$

□

Lemma E.6 (Evolution of variance)**.** *Let* $U_t = \operatorname{Var}\left[\theta_t\right]$ *and* $V_t = \operatorname{Cov}\left[\theta_t, \theta_{t-1}\right]$*, where* θ_t *is the* t*-th iterate of Heavyball-IGT on a 1-dimensional quadratic function with curvature* h*. The following matrix describes the variance dynamics of Heavyball-IGT.*

$$D = \begin{pmatrix} (1-\alpha h+\mu)^2 + 2\alpha^2 h^2 & \mu^2 & -2\mu(1-\alpha h+\mu)^2 \\ 1 & 0 & 0 \\ 1-\alpha h+\mu & 0 & -\mu \end{pmatrix} \tag{7}$$

If the spectral radius of D*,* $\rho(D)$*, is strictly less than 1, then there exist constants* $t_0 > 0$ *and* $C > 0$ *for which*

$$\operatorname{Var}(\theta_t) \leq 2\alpha^2 BC\frac{\log(t)}{t}$$

, where B *is a bound on the variance of the noise.*

Proof. The proof (and lemma) is similar to the proof of Lemma 9 in [49]. We start by expanding U_{t+1} as follows.

$$\begin{aligned}
U_{t+1} &= \mathbb{E}\left[(\theta_{t+1} - \bar{\theta}_{t+1})^2\right] \\
&= \mathbb{E}\left[(\theta_t - \alpha v_t + \mu(\theta_t - \theta_{t-1}) - \bar{\theta}_t + \alpha g_t - \mu(\bar{\theta}_t - \bar{\theta}_{t-1}))^2\right] \\
&= \mathbb{E}[(\theta_t - \alpha g_t + \mu(\theta_t - \theta_{t-1}) - \bar{\theta}_t + \alpha g_t \\
&\qquad - \mu(\bar{\theta}_t - \bar{\theta}_{t-1}) + \alpha(g_t - v_t))^2] \\
&= \mathbb{E}\left[((1-\alpha h+\mu)(\theta_t - \bar{\theta}_t) - \mu(\theta_{t-1} - \bar{\theta}_{t-1}))^2\right] \\
&\qquad + \alpha^2\mathbb{E}\left[(g_t - v_t)^2\right] \\
&\leq \mathbb{E}\left[((1-\alpha h+\mu)(\theta_t - \bar{\theta}_t) - \mu(\theta_{t-1} - \bar{\theta}_{t-1}))^2\right] \\
&\qquad + \alpha^2\left(2h^2\mathbb{E}\left[(\theta_t - \bar{\theta}_t)^2\right] + \frac{2B}{t+1}\right) \\
&\leq \left[(1-\alpha h+\mu)^2 + 2\alpha^2\mu^2\right]\mathbb{E}\left[(\theta_t - \bar{\theta}_t)^2\right] \\
&\qquad - 2\mu(1-\alpha h+\mu)\mathbb{E}\left[(\theta_t - \bar{\theta}_t)(\theta_{t-1} - \bar{\theta}_{t-1})\right] \\
&\qquad + \mu^2\mathbb{E}\left[(\theta_{t-1} - \bar{\theta}_{t-1})^2\right] + \alpha^2\frac{2B}{t+1}\ .
\end{aligned}$$

Where the fourth equality is obtained since we know that the IGT gradient estimator is unbiased, i.e. $\mathbb{E}\left[g_t - v_t\right] = 0$. The first inequality stems from Lemma E.4. We similarly expand V_t.

$$\begin{aligned}
V_t &= \mathbb{E}\left[(\theta_t - \bar{\theta}_t)(\theta_{t-1} - \bar{\theta}_{t-1})\right] \\
&= \mathbb{E}\big[\big((1-\alpha h+\mu)(\theta_{t-1} - \bar{\theta}_{t-1}) - \mu(\theta_{t-2} - \theta_{\bar{t}-2}) + \alpha(g_t - v_t)\big) \\
&\qquad (\theta_{t-1} - \bar{\theta}_{t-1})\big] \\
&= (1-\alpha h+\mu)\mathbb{E}\left[(\theta_{t-1} - \bar{\theta}_{t-1})^2\right] \\
&\qquad - \mu\mathbb{E}\left[(\theta_{t-1} - \bar{\theta}_{t-1})(\theta_{t-2} - \bar{\theta}_{t-2})\right]
\end{aligned}$$

From the above expressions, we obtain

$$
\begin{aligned}
\begin{pmatrix} U_{t+1} \\ U_t \\ V_{t+1} \end{pmatrix} &\leq D \begin{pmatrix} U_t \\ U_{t-1} \\ V_t \end{pmatrix} + \begin{pmatrix} \alpha^2 \frac{2B}{t+1} \\ 0 \\ 0 \end{pmatrix} \\
&\leq 2\alpha^2 B \sum_{i=0}^{t} D^i \begin{pmatrix} \frac{1}{t+1-i} \\ 0 \\ 0 \end{pmatrix} \\
&\leq 2\alpha^2 B \left(\sum_{i=0}^{s-1} D^i \begin{pmatrix} \frac{1}{t+1-i} \\ 0 \\ 0 \end{pmatrix} + \sum_{i=s}^{t} D^i \begin{pmatrix} \frac{1}{t+1-i} \\ 0 \\ 0 \end{pmatrix} \right)
\end{aligned}
$$

where an inequality of vectors implies the corresponding elementwise inequalities.

If the spectral radius of D, $\rho(D)$ is strictly less than 1, then there exists constant $C' > 0$ such that

$$
\begin{aligned}
\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}^T \sum_{i=0}^{s-1} D^i \begin{pmatrix} \frac{1}{t+1-i} \\ 0 \\ 0 \end{pmatrix} &\leq C' \sum_{i=0}^{s-1} \frac{1}{t+1-i} \\
&\leq \frac{C's}{t+2-s}
\end{aligned}
$$

If the spectral radius of D, $\rho(D)$, is strictly less than 1, then there exists constant $\zeta > 0$ and constant $C''(\zeta) > 0$ such that, $\rho(D) + \zeta < 1$ and

$$
\begin{aligned}
\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}^T \sum_{i=s}^{t} D^i \begin{pmatrix} \frac{1}{t+1-i} \\ 0 \\ 0 \end{pmatrix} &\leq \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}^T \sum_{i=s}^{t} D^i \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \\
&\leq C'' \sum_{i=s}^{t} (\rho(D) + \zeta)^s \\
&= C''(t - s + 1)(\rho(D) + \zeta)^s
\end{aligned}
$$

Let $\rho' = \rho(D) + \zeta$ and $s = \lceil 2 \log_{1/\rho'} t \rceil$. Then $(\rho(D) + \zeta)^s = 1/t^2$, and putting the above two bounds together,

$$
\begin{aligned}
U_{t+1} &\leq 2\alpha^2 B \left(\frac{2C' \log_{1/\rho'} t}{t + 2 - 2\log_{1/\rho'} t} + C'' \frac{t - 2\log_{1/\rho'} t + 1}{t^2} \right) \\
&\leq 2\alpha^2 BC \frac{\log(t+1)}{t+1}
\end{aligned}
$$

where the last inequality holds for $t > t_0$ for some t_0 and some constant $C > 0$.

□

We can now prove Proposition E.1.

Proof of Proposition E.1. The bias statement of the proposition follows directly from taking an expectation on the bound of Lemma B.4, and the variance statement from summing up the d different variance terms given for each scalar component by Lemma B.5. □

E.1 Proof of Proposition 3.2

This Proposition follows from the observation that, in the noiseless case, $\epsilon_t = 0$ in our model. In that case, Lemma E.3 shows that Heavyball-IGT reduces to the heavy ball, and the rest follows from the optimal tuning of the heavy ball [49].

E.2 Proof of Proposition 3.3

Proof. Like we did in previous proofs, we can assume without loss of generality that the Hessian, H, is diagonal with elements h_i. For a diagonal H, matrix A can be permuted to be block diagonal with blocks

$$A_i = \begin{pmatrix} 1 - \alpha h_i + \mu & \mu \\ 1 & 0 \end{pmatrix}.$$

To prove that $\rho(A) < 1$ it suffices to prove that $\rho(A_i) < 1$ for all i. For the rest of the proof we will focus on the dynamics along a single eigendirection with curvature h_i. The rest of this proof used D to denote D_i, A to denote A_i and h to denote h_i.

To make explicit the dependence of matrices A and D on hyperparameters and curvature, we write $A(\alpha, \mu, h)$ and $D(\alpha, \mu, h)$. Let $0 < \alpha < 2/(3h)$ and $\mu_0 = 0$. Using hyperparameters (α, μ_0) one gets the results for gradient descent without momentum. In particular $\rho(A(\alpha, \mu_0, h)) = |1 - \alpha h| < 1$, and the spectral radius of D is $\rho(D(\alpha, \mu_0, h)) = |(1 - \alpha h)^2 + 2\alpha^2 h^2| < 1$.

We will argue that there exists $\mu > 0$, such that $\rho(A(\alpha, \mu, h)) < 1$, and the spectral radius of D is $\rho(D(\alpha, \mu, h)) < 1$. Then the previous lemma implies that bias converges linearly, and variance is $O(\log(t)/t)$.

To argue the existence of $\mu > 0$, we will perform eigenvalue perturbation analysis using the Bauer-Fike theorem. Note that $A(\alpha, \mu, h) = A(\alpha, \mu_0, h) + \mu \Delta_A$ where

$$\Delta_A = \begin{pmatrix} 1 & -1 \\ 0 & 0 \end{pmatrix}.$$

Similarly, $D(\alpha, \mu, h) \approx D(\alpha, \mu_0, h) + \mu \Delta_D$ where

$$\Delta_D = \begin{pmatrix} 2(1 - \alpha h) & 0 & -2(1 - \alpha h) \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{pmatrix}.$$

This last approximate inequality is a first-order approximation, in the sense that we are working with arbitrarily small, positive values of μ, and we have kept terms linear in μ but ignored higher order terms, like μ^2.

We will apply the Bauer-Fike theorem to bound the eigenvalues of $D(\alpha, \mu, h)$. Consider the eigendecomposition $D(\alpha, \mu_0, h) = V \Lambda V^{-1}$. We can compute

$$V = \begin{pmatrix} 0 & 0 & \frac{1 - 2\alpha h + 3\alpha^2 h^2}{1 - \alpha h} \\ 0 & 1 & \frac{1}{1 - \alpha h} \\ 1 & 0 & 1 \end{pmatrix}$$

and

$$V^{-1} = \begin{pmatrix} \frac{1 - \alpha h}{1 - 2\alpha h + 3\alpha^2 h^2} & -\frac{1}{1 - 2\alpha h + 3\alpha^2 h^2} & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}.$$

Note that because we assume $\alpha < 2/(3h)$ we get $1 - \alpha h > 0$. Also, $1 - 2\alpha h + 3\alpha^2 h^2 > 0$ regardless of the choice of hyperparameters. This means that matrices V and V^{-1} are singular and of finite norm. The norm of Δ_D is also finite. The Bauer-Fike theorem states that, if ν is an eigenvalue of $D(\alpha, \mu_0, h)$, then there exists an eigenvalue λ of $D(\alpha, \mu, h)$ such that

$$|\lambda - \nu| \leq \|V\|_p \|V^{-1}\|_p \|\mu \Delta_D\|_p,$$

for any p-norm. Since by construction $|\nu| \leq \rho(D(\alpha, \mu_0, h)) < 1$, the above means that there exists a sufficiently small, but strictly positive value of μ, such that $\lambda < 1$. By repeating this argument for all pairs of eigenvalues, we get the stated result. The same argument can be repeated to prove the existence of a strictly positive μ such that $\rho(A(\alpha, \mu, h)) < 1$.

□