Journal of Machine Learning Research 25 (2024) 1-55 Submitted 9/21; Revised 5/23; Published 1/24

Iterate Averaging in the Quest for Best Test Error

Diego Granziol* DIEGO@PURESTRENGTH.AI
Machine Learning Research Group
University of Oxford, Oxford, UK

Nicholas P. Baskerville* N.P.BASKERVILLE@BRISTOL.AC.UK
School of Mathematics
University of Bristol, Bristol, UK

Xingchen Wan* XWAN@ROBOTS.OX.AC.UK
Machine Learning Research Group
University of Oxford, Oxford, UK

Samuel Albanie SAMUEL.ALBANIE.ACADEMIC@GMAIL.COM
Department of Engineering
University of Cambridge, Cambridge, UK

Stephen Roberts SJROB@ROBOTS.OX.AC.UK
Machine Learning Research Group
University of Oxford, Oxford, UK

Editor: Simon Lacoste-Julien

Abstract

We analyse and explain the increased generalisation performance of iterate averaging using a Gaussian process perturbation model between the true and batch risk surface on the high dimensional quadratic. We derive three phenomena from our theoretical results: (1) The importance of combining iterate averaging (IA) with large learning rates and regularisation for improved generalisation. (2) Justification for less frequent averaging. (3) That we expect adaptive gradient methods to work equally well, or better, with iterate averaging than their non-adaptive counterparts. Inspired by these results, together with empirical investigations of the importance of appropriate regularisation for the solution diversity of the iterates, we propose two adaptive algorithms with iterate averaging. These give significantly better results compared to stochastic gradient descent (SGD), require less tuning and do not require early stopping or validation set monitoring. We showcase the efficacy of our approach on the CIFAR-10/100, ImageNet and Penn Treebank datasets on a variety of modern and classical network architectures.

Keywords: iterate averaging, generalisation, deep learning theory, deep learning limit, adaptive gradient methods

*. These authors contributed equally to this work. XW discovered the AdamW + IA combination and led many experiments. DG developed the true gradient perturbation framework using i.i.d gaussian assumptions and conducted ImageNet experiments and paper writing. NPB expanded the i.i.d framework to a general GP and did the AdamW convergence proofs. SA helped in ImageNet experiments and SR helped with foundation concepts and did extensive proof reading; both provided overall supervision.

1 Introduction

Deep neural network (DNN) models achieve state of the art performance in a plethora of problems, such as speech recognition, visual object image recognition, object detection, drug discovery and genomics (LeCun et al., 2015). Of key interest is the ability of DNNs to "generalise" to unseen data, even when the parameter number greatly exceeds the dataset size (Zhang et al., 2016). DNNs are typically trained using stochastic gradient descent (SGD), in which model parameters at each optimisation step, $\boldsymbol{w}_{k+1}$, are updated using the gradient of the minibatch loss at the previous step, $L(\boldsymbol{w}_k)$:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \nabla L(\boldsymbol{w}_k), \tag{1}$$

where α_k denotes the learning rate at iteration k. Whilst careful monitoring of the validation metrics, along with weight decay (Krogh and Hertz, 1992), layer-wise normalisation (Ioffe and Szegedy, 2015) and data-augmentation (Shorten and Khoshgoftaar, 2019; Zhang et al., 2017) help protect against over-fitting to the training data, the initial value of α_k and its schedule throughout training has a large impact on generalisation (Jastrzebski et al., 2017; Li et al., 2019), making it a key hyperparameter to set correctly. Theoretical results for optimal asymptotic training set convergence prescribe a learning rate proportional to the inverse square root of the number of iterations (Nesterov, 2013) or a decay at this rate (Duchi, 2018). However, such schedules often result in poor test set performance for DNNs. Curiously, Merity et al. (2017) and Izmailov et al. (2018) demonstrate that combining *tail iterate averaging* (i.e. taking an average of the last iterates in training) with large learning rate SGD increases DNN generalisation at the expense of training accuracy. However, quite why and how this works is still something of a mystery, limiting its widespread adoption.

One proposal in the literature to limit sensitivity to the learning rate and its schedule has been the development of *adaptive gradient optimisers*, which invoke a per-parameter learning rate based on the history of gradients. Popular examples include Adam (Kingma and Ba, 2014), AdaDelta (Zeiler, 2012) and RMSprop (Tieleman and Hinton, 2012). Ignoring momentum and explicit regularisation, the k^{th} iteration of a general adaptive optimiser is given by:

$$\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{w}_k - \alpha_k \boldsymbol{B}^{-1} \nabla L_k(\boldsymbol{w}_k), \tag{2}$$

where the preconditioning matrix $\boldsymbol{B}$ typically approximates curvature information. Crucially, however, the generalisation of solutions found using adaptive methods, as measured in terms of test and validation error, significantly underperforms SGD (Wilson et al., 2017). Due to this, state-of-the-art convolutional neural networks, especially for image classification datasets such as CIFAR (Yun et al., 2019) and ImageNet (Xie et al., 2019; Cubuk et al., 2019) are still trained using SGD with momentum (Nesterov, 2013). Furthermore, despite Iterate Averaging (IA) being mentioned as a potential amendment in the original Adam paper (Kingma and Ba, 2014) and being required in the convergence proof (Reddi et al., 2019), it is not widely used for computer vision or other complex problems.

2 Contributions

The key contribution of this paper are:

- We investigate the impact of IA on generalisation by considering high-dimensional SGD on the quadratic model of the true risk perturbed by i.i.d. Gaussian noise, showing that the iterate average attains the global minimum, whereas the final point, despite multiple learning rate drops, or increases in batch size during training, does not.
- We extend the framework to a Gaussian process perturbation model between the true and batch gradients. We find that as long as certain technical conditions (well met in practice) are satisfied, the simplified result holds. Crucially, distance in weight space or relative weight space (depending on kernel choice) are pivotal to the effect, *justifying in practice the need for large learning rates in conjunction with iterate averaging.*
- We show that adaptive gradient methods have identical properties under the iterate average, but we expect them to converge faster than their non-adaptive counterparts.
- Motivated by these results, we consider *why adaptive methods are not typically used in conjunction with iterate averaging*? We find that ineffective regularisation, which limits the effective distance and prediction diversity between the iterates, is the main culprit and propose a simple, yet effective solution. We propose two Adam-based algorithms: **Gadam** and **GadamX**. Both outperform baselines tested for all networks and datasets we consider. GadamX achieves a Top-1 error of 22.69% on ImageNet using ResNet-50, outperforming a well-tuned SGD baseline of 23.85% (Chintala et al., 2017). To put this into perspective, the gain attributed to widely-adopted cosine schedules increases accuracy by 0.3% (Bello et al., 2021). We thus show that adaptive methods can outperform SGD and SGD with IA provides a practical framework that can be used, even for large-scale problems.
- We show explicitly that for a small eigenvalue error in the implied Adam curvature matrix, we expect Adam with decoupled weight decay to converge faster to the same low test error as SGD. This result implies that practioners should use Adam with decoupled weight decay and iterate averaging over SGD (with or without iterate averaging).

Related work & motivation. To the best of our knowledge there has been no explicit theoretical work analysing the generalisation benefit of iterate averaging. Whilst Izmailov et al. (2018) propose that iterate averaging leads to "flatter minima which generalise better", flatness metrics are known to have limitations as a proxy for generalisation (Dinh et al., 2017), in addition to which we show in the Appendix Section E.1 that adaptive methods can find very sharp minima with good generalisation properties. Martens (2014) show that the IA convergence rate for both SGD and second-order methods are identical, but argue that second-order methods have an optimal pre-asymptotic convergence rate on a quadratic loss surface. Here, pre-asymptotic means before taking the number of iterations $t \to \infty$ and quadratic means that the Hessian is constant at all points in weight-space. The analysis does not extend to *generalisation* and no connection is made to adaptive gradient methods, nor to the importance of the high parameter-space dimensionality of the problem, two major contributions of our work. Further related theoretical work includes Bach and Moulines (2013) who analyse the convergence in true risk using unbiased gradient estimators using averaged SGD and specifically for least squares regression, that averaged SGD with a constant step size leading to minimax optimal finite time prediction guarantees after a single pass

on the data. Neu and Rosasco (2018) show that averaging geometrically is equivalent to considering regularised SGD and give prescriptions based on the condition number of the problem and dataset size as to when to use geometric or simple averaging. Amendments to improve the generalisation of adaptive methods include switching between Adam and SGD (Keskar and Socher, 2017) and decoupled weight decay (Loshchilov and Hutter, 2019), limiting the extent of adaptivity (Chen and Gu, 2018; Zhuang et al., 2020). We incorporate these insights into our algorithms but significantly outperform them experimentally. The closest algorithmic contribution to our work is Izmailov et al. (2018) however, they only look at SGD, which is not the focus of our work. Specifically for adaptive optimisers the closest work is *Lookahead* (Zhang et al., 2019b), which combines adaptive methods with an exponentially moving average scheme. We analyse this algorithm both theoretically (see Appendix Section B.1) and experimentally.

3 Iterate Averaging: A New Theory for Generalisation

The iterate average (Polyak and Juditsky, 1992) is the arithmetic mean of the model parameters over the optimisation trajectory $\boldsymbol{w}_{\text{avg}} = \frac{1}{n}\sum_i^n \boldsymbol{w}_i$. It is a classical variance reducing technique in optimisation and offers optimal asymptotic convergence rates and greater robustness to the choice of learning rate (Kushner and Yin, 2003). Indeed, popular regret bounds that form the basis of gradient-based convergence proofs (Duchi et al., 2011; Reddi et al., 2019) often consider convergence for the iterate average (Duchi, 2018). Further, theoretical extensions have shown that the rate of convergence can be improved by a factor of $\log T$ (where T is the iteration number) by *suffix averaging* (Rakhlin et al., 2011), which considers a fraction of the last iterates, *polynomial decay averaging* (Shamir and Zhang, 2013) which decays the influence of the previous iterates, or *weighted averaging* (Lacoste-Julien et al., 2012) which weights the iterate by its iteration number. That the final iterate of SGD is sub-optimal in terms of its convergence rate, by this logarithmic factor, has been proved by Harvey et al. (2019). However, under an alternative decay schedule it can be shown to be equal to that of averaged schemes (Jain et al., 2019).

For networks with batch normalisation (Ioffe and Szegedy, 2015), a naïve application of IA (in which we simply average the batch normalisation statistics) is known to lead to poor results (Defazio and Bottou, 2019). However, by computing the batch normalisation statistics for the iterate average using a forward pass of the data at the IA point, Izmailov et al. (2018) show that the performance of small-scale image experiments such as CIFAR-10/100 and pretrained ImageNet can be significantly improved. Even for small experiments this computation is expensive, so they further approximate IA by taking the average at the end of each epoch instead of each iteration, referred to as *stochastic weight averaging* (SWA). We show experimentally in Section 7.2 that the two approaches produce almost identical results, with SWA slightly outperforming IA. Since SWA can be seen as IA with a lower averaging frequency, we retain the terminology IA - however, in our theoretical analysis we also include analysis for reduced frequency iterate averaging.

Notations:

- With some variable $n \to \infty$, and scalar-valued functions f, g, $f(n) = o(g(n))$ is shorthand for $f(n)/g(n) \to 0$ as $n \to \infty$. Similarly, $f(n) = \mathcal{O}(g(n))$ is shorthand for

$f(n)/g(n) \to c$, for some constant $c > 0$. In particular $\mathcal{O}(1)$ can be read as shorthand for any fixed non-zero constant, and $o(1)$ for any term which decays to 0. For example $f(n) = 3n + 2 + 1/n$ can be abbreviated as $f(n) = \mathcal{O}(n)$, or $f(n) = 3n + \mathcal{O}(1)$, or $f(n) = 3n + 2 + o(1)$, depending on the level of precision required. We will also employ the asymptotic equivalence notation[1] $f(n) \sim g(n)$ to denote the special case $f(n)/g(n) \to 1$.

- For matrices B, $\|B\|_F = \sqrt{\sum_{ij} B_{ij}^{1/2}}$ denotes the Frobenius norm and $\|B\| = \sup_{\|q\|=1} \|B\boldsymbol{q}\|$ denotes the operator norm.
- For random vectors X, define $\|X\|_{\psi_2} = \inf\left\{t > 0 : \mathbb{E}\exp(X^2/t^2) \leq 2\right\}$.
- For a set of P positive eigenvalues $\lambda_1, \ldots, \lambda_P$ and any rational function of f, $\langle f(\lambda)\rangle = \frac{1}{P}\sum_{i=1}^{P} f(\lambda_i)$.

3.1 A High-Dimensional Geometry Perspective

We examine the variance reducing effect of IA in the context of a quadratic approximation to the true loss combined with additive perturbation models for the batch training loss.

The theory we present is high-dimensional (i.e. large number of parameters, P) and considers the small batch size (small B) regime, which we term the "deep learning limit".

Intuitively, any given example from the training set $j \in \mathcal{D}$, will contain *general features*, which hold over the data generating distribution and *instance specific features* (which are relevant only to the training sample in question). For example, for a training image of a dog, we may have that:

$$\overbrace{\underbrace{\nabla L_{\text{sample}}(\boldsymbol{w})}_{\text{training set example}}}^{\text{dog } j} = \overbrace{\underbrace{\nabla L_{\text{true}}(\boldsymbol{w})}_{\text{general features}}}^{\text{4 legs, snout}} + \overbrace{\underbrace{\boldsymbol{\epsilon}(\boldsymbol{w}).}_{\text{instance-specific features}}}^{\text{black pixel in top corner, green grass}} \tag{3}$$

Note that mathematically, it is trivial, to write the batch loss as in Granziol (2020a) to write

$$\nabla L_{\text{batch}} = \nabla L_{\text{true}} + \left(\nabla L_{\text{batch}} - \nabla L_{\text{true}}\right) \tag{4}$$

Under such a writing the difference between the sample or mini-batch gradient and that of the true gradient can be considered as instance or batch specific features. The basic insight of our paper is that even by simply considering the instance specific or batch specific perturbation to be a Gaussian perturbation (with no bias) and the loss to be quadratic we get interesting results in the high dimensional regime (shown below). We take this idea further by considering a fully general Gaussian process noise model (which allows for bias and more interesting noise distributions) but the key concept remains unchanged. Under a quadratic approximation to the *true loss*[2] $L_{\text{true}}(\boldsymbol{w}) = \boldsymbol{w}^T\boldsymbol{H}\boldsymbol{w}$, where $\boldsymbol{H} = \nabla^2 L$ is the Hessian of the true loss with respect to the weights. We sample a mini-batch gradient of

1. Note that $\sim$ is also often used to denote equivalence in the sense of asymptotic expansions; this is not the definition we use.
2. The loss under the expectation of the data generating distribution, rather than the loss over the dataset $L_{\text{emp}}(\boldsymbol{w}_k)$.

size B at point $\boldsymbol{w} \in \mathbb{R}^{P\times 1}$. The observed mini-batch gradient is perturbed by $\boldsymbol{\epsilon}(\boldsymbol{w})$ from the true loss gradient (due to instance specific features as shown in Equation equation 3). Under this model the component of the $\boldsymbol{w}_t$'th iterate along the j'th eigenvector $\boldsymbol{\phi}_j$ of the true loss when running SGD with learning rate α can be written:

$$\boldsymbol{w}_t^T \boldsymbol{\phi}_j = (1-\alpha\lambda_j)^t \boldsymbol{w}_0^T \boldsymbol{\phi}_j - \alpha(1-\alpha\lambda_j)^{t-1} \boldsymbol{\epsilon}(\boldsymbol{w}_1)^T \boldsymbol{\phi}_j \cdots , \tag{5}$$

in which $\lambda_j, \boldsymbol{\phi}_j$ are the eigenvalue/eigenvector pairs of $\boldsymbol{H}$ where the dependence on the iterate $\boldsymbol{w}$ has been dropped. The simplest tractable model for the gradient noise $\boldsymbol{\epsilon}(\boldsymbol{w}_t)$ is to assume the data is sampled from the data generating distribution with some additive isotropic, multivariate Normal noise. In particular, this assumption removes any dependence on $\boldsymbol{w}_t$ and precludes the existence of any distinguished directions in the gradient noise. Using this assumption, we obtain Theorem 2 below, which relies on an intermediate result, found in Vershynin (2018).

Lemma 1 (Vershynin (2018) Theorem 6.3.2) *Let R be an $m \times n$ matrix, and let $X = (X_1, \ldots, X_n) \in \mathbb{R}^n$ be a random vector with independent mean-zero unit-variance sub-Gaussian coordinates. Then for any $\xi \in \mathbb{R}$*

$$\mathbb{P}\left(|\|RX\|_2 - \|R\|_F| > \xi\right) \leq 2\exp\left(-\frac{c\xi^2}{K^4\|R\|^2}\right)$$

where $K = \max_i \|X_i\|_{\psi_2}$ and $c > 0$ is a constant.

Theorem 2 *Assume the quadratic loss model $L_{true}(\boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{H} \boldsymbol{w}$, where $\boldsymbol{H}$ has eigenvalues $\{\lambda_i\}_{i=1}^P$ and assume the $\{\epsilon_t\}_{t=0}^n$ are all i.i.d. Gaussian vectors in $\mathbb{R}^P$ with distribution $\mathcal{N}(0, \sigma^2 B^{-1} I)$ where B is the batch size. Assume the weights are updated according to the rule from (5)*

$$\boldsymbol{w}_t^T \boldsymbol{\phi}_j = (1-\alpha\lambda_j)^t \boldsymbol{w}_0^T \boldsymbol{\phi}_j - \alpha(1-\alpha\lambda_j)^{t-1} \boldsymbol{\epsilon}(\boldsymbol{w}_1)^T \boldsymbol{\phi}_j. \tag{6}$$

Assume further that $\alpha\lambda_i < 1$ for all i and $\lambda_i > 0$ for all i. Then there exists a constant $c > 0$ such that for all $\xi > 0$, as $n \to \infty$

$$\begin{aligned}
&\mathbb{P}\left(\left|\sqrt{\sum_i^P \left(w_{n,i} - w_{0,i}\mathrm{e}^{-n\alpha\lambda_i}(1+o(1))\right)^2} - \sqrt{P\frac{\alpha\sigma^2}{B}\left\langle \frac{1}{\lambda(2-\alpha\lambda)}\right\rangle}\right| \geq \xi\right) \leq \nu(\xi), \\
&\mathbb{P}\left(\left|\sqrt{\sum_i^P \left(w_{\mathrm{avg},i} - \frac{w_{0,i}}{\lambda_i n\alpha}(1+o(1))\right)^2} - \sqrt{\frac{P\sigma^2}{Bn}\left\langle \frac{1}{\lambda}\right\rangle}\right| \geq \xi\right) \leq \nu(\xi),
\end{aligned} \tag{7}$$

where $\nu(\xi) = 2\exp(-c\xi^2)$ and where $w_{n,i}$ are the components of $\boldsymbol{w}_n$ in the basis $\{\boldsymbol{\phi}_i\}_i$.

We note that several works (Wu et al., 2018b; Martens, 2014; Polyak and Juditsky, 1992; Kushner and Yin, 2003) investigate the convergence on noisy quadratics of the last and average iterates and the maths is straightforward. What we bring forward to the reader and what we expand upon greatly in the remainder of this manuscript is the dependence on dimensionality. To the best of our knowledge this has not been considered in previous works.

Proof Let $Y = (Y_1, \ldots, Y_P)$ be a random sub-Gaussian vector with independent components. Let

$$X_i = \frac{Y_i - \mathbb{E}Y_i}{\sqrt{\mathrm{Var}Y_i}}, \quad R = \mathrm{diag}(\sqrt{\mathrm{Var}Y_1}, \ldots, \sqrt{\mathrm{Var}Y_P}).$$

Lemma 1 then applies, to give

$$\mathbb{P}\left(\left|\|Y - \mathbb{E}Y\|_2 - \sqrt{\sum_{i=1}^{P} \mathrm{Var}Y_i}\right| > \xi\right) \leq 2\exp\left(-\frac{c\xi^2}{K^4\|R\|^2}\right).$$

We have $K \leq C \max_i \mathrm{Var}Y_i$ for some constant $C > 0$ (Vershynin (2018), exercise 2.5.8), and $\|R\|^2 = (\max_i \sqrt{\mathrm{Var}Y_i})^2 = \max_i \mathrm{Var}Y_i$. Hence we obtain

$$\mathbb{P}\left(\left|\|Y - \mathbb{E}Y\|_2 - \sqrt{\sum_{i=1}^{P} \mathrm{Var}Y_i}\right| > \xi\right) \leq 2\exp\left(-\frac{c\xi^2}{(\max_i \mathrm{Var}Y_i)^2}\right) \tag{8}$$

for some new constant $c > 0$. The proof is then completed if we compute the means and variances of $\boldsymbol{w}_n$ and $\boldsymbol{w}_{\mathrm{avg}}$. To that end, with $\boldsymbol{\Lambda} = \mathrm{diag}\,(\lambda_1, \ldots, \lambda_P)$, the update rule (6) gives

$$\boldsymbol{w}_t = (1 - \alpha\boldsymbol{\Lambda})^n \boldsymbol{w}_0 + \alpha \sum_{i=0}^{t-1} (1 - \alpha\boldsymbol{\Lambda})^{t-i-1} \boldsymbol{\epsilon}_i, \tag{9}$$

for any $1 \leq t \geq n$. Since $\boldsymbol{\Lambda}$ is diagonal, each component of $\vec{w}_n$ can be treated independently when we sum to obtain $\vec{w}_{avg}$, so for any vector $\vec{v}$

$$\sum_{t=1}^{n} (1 - \alpha\boldsymbol{\Lambda})^t \boldsymbol{v} = \frac{1 - (1 - \alpha\boldsymbol{\Lambda})^t}{\alpha} \boldsymbol{\Lambda}^{-1}(1 - \alpha\boldsymbol{\Lambda})\boldsymbol{v} \tag{10}$$

So averaging (9) over t gives

$$\boldsymbol{w}_{avg} = \frac{1 - (1 - \alpha\boldsymbol{\Lambda})^n}{\alpha n} \boldsymbol{\Lambda}^{-1}(1 - \alpha\boldsymbol{\Lambda})\boldsymbol{w}_0 + \sum_{t=0}^{n-1} \frac{1 - (1 - \alpha\boldsymbol{\Lambda})^{n-t}}{n} \boldsymbol{\Lambda}^{-1}\boldsymbol{\epsilon}_t. \tag{11}$$

Since the $\boldsymbol{\epsilon}_i$ are all i.i.d. centred Gaussians, obtaining the distributions of $\boldsymbol{w}_n$ and $\boldsymbol{w}_{avg}$ amounts to computing the covariances

$$\begin{aligned} \mathrm{Cov}\left(\alpha \sum_{i=0}^{n-1} (1 - \alpha\boldsymbol{\Lambda})^{n-i-1} \boldsymbol{\epsilon}_i\right) =& \sigma^2 B^{-1} I \sum_{i=1}^{n-1} \alpha^2 (1 - \alpha\boldsymbol{\Lambda})^{2(n-i-1)} \\ =& \sigma^2 B^{-1} I \alpha^2 (1 - (1 - \alpha\boldsymbol{\Lambda})^{2n}) \left(1 - (1 - \alpha\boldsymbol{\Lambda})^2\right)^{-1} \end{aligned} \tag{12}$$

and similarly

$$\begin{aligned} &\mathrm{Cov}\left(\sum_{t=0}^{n-1} \frac{1 - (1 - \alpha\boldsymbol{\Lambda})^{n-t}}{n} \boldsymbol{\Lambda}^{-1}\boldsymbol{\epsilon}_t\right) \\ =& \sum_{t=0}^{n-1} \left(\frac{1 - (1 - \alpha\boldsymbol{\Lambda})^{n-t}}{n} \boldsymbol{\Lambda}^{-1}\right)^2 \\ =& \frac{\boldsymbol{\Lambda}^{-2}}{n^2} \left(n - \frac{2(1 - (1 - \alpha\boldsymbol{\Lambda})^n)}{\alpha} \boldsymbol{\Lambda}^{-1} + \left(1 - (1 - \alpha\boldsymbol{\Lambda})^{2n}\right) \left(1 - (1 - \alpha\boldsymbol{\Lambda})^2\right)^{-1}\right). \end{aligned} \tag{13}$$

Now using $\alpha\lambda_i < 1$ for all $i = 1, 2 \ldots, P$, and taking $n \to \infty$, (9) and (12) give

$$\mathrm{Cov}(\boldsymbol{w}_n) \sim \sigma^2\alpha^2 B^{-1}\left(1-(1-\alpha\boldsymbol{\Lambda})^2\right)^{-1} = \sigma^2\alpha B^{-1}\left(2\boldsymbol{\Lambda}-\alpha\boldsymbol{\Lambda}^2\right)^{-1} \tag{14}$$

and similarly (11) and (13) give

$$\mathrm{Cov}(\boldsymbol{w}_{avg}) \sim \frac{1}{n}\boldsymbol{\Lambda}^{-2}. \tag{15}$$

Thus it follows from (9) and (14) that

$$\mathbb{E}w_{n,i} = (1-\alpha\lambda_i)^n w_{0,i} \sim e^{-n\alpha\lambda_i} w_{0,i}, \;\; \mathrm{Var}(w_{n,i}) \sim \frac{\sigma^2}{B}\frac{\alpha}{2\lambda_i(1-\alpha\lambda_i)} \tag{16}$$

and from (11) and (15) it follows

$$\mathbb{E}w_{avg,i} \sim \frac{w_{0,i}}{\lambda_i\alpha n}, \quad \mathrm{Var}(w_{avg,i}) = \frac{\sigma^2}{B}\frac{1}{n\lambda_i^2} \tag{17}$$

where in both cases we have used $\alpha\lambda_i \ll 1$ to simplify the expected values for large n. To complete the proof for $\boldsymbol{w}_n$, we apply (8) using (16) and noting that

$$\sqrt{\sum_{i=1}^{P}\mathrm{Var}(w_{n,i})} \sim \frac{\sigma^2 P\alpha}{2B}\left\langle\frac{1}{\lambda(1-\alpha\lambda)}\right\rangle \tag{18}$$

and $0 < \max_i w_{n,i} < \infty$ since $\lambda_i > 0$ and $\alpha\lambda_i < 1$. The results for $\boldsymbol{w}_{avg}$ follows similarly by using (8) with (17). This produces two different constants $c > 0$ in the statement of (7), but we can simply take the smaller of the two constants to produce the desired statement.

■

The final iterate attains exponential convergence in the mean of $\boldsymbol{w}_n$, but does not control the variance term. Whereas for $\boldsymbol{w}_{\mathrm{avg}}$, although the convergence in the mean is worse (linear), the variance vanishes asymptotically – this motivates *tail averaging*, to get the best of both worlds. Another key implication of Theorem 2 lies in its dependence on P. P is a gauge of the model size and appears as a simple linear multiplier of the variances of $\boldsymbol{w}_n$ and $\boldsymbol{w}_{avg}$, so increasing over-parametrisation implies increasing variance of the final iterate and the IA, however IA provides a counterbalancing variance reduction effect that is entirely absent from the final iterate. This implies that in more complex, over-parameterised models, we expect the benefit of IA over the final iterate to be greater, as IA provides a mechanism to control the weight variance even as it grows with P. We show this explicitly in our experiments in Figure 4c. Note the limited extra improvement possible by simply increasing the batch size, compared to IA asymptotically.

3.2 A Dependent Model for the Perturbation

We proceed now to propose a relaxation of the gradient perturbation independence assumption. (3) can be written equivalently as

$$L_{\mathrm{batch}}(\boldsymbol{w}) = L_{\mathrm{true}}(\boldsymbol{w}) + \eta(\boldsymbol{w}) \tag{19}$$

where η is a scalar field with $\nabla\eta = \boldsymbol{\epsilon}$. Note that we have neglected an irrelevant arbitrary constant in Equation (19) and also that we have L_{batch} rather than L_{sample}, but this amounts to scaling the per-sample noise variance σ^2 by the inverse batch size B^{-1}. We model η as a Gaussian process $\mathcal{GP}(m, k)$, where k is some kernel function $\mathbb{R}^P \times \mathbb{R}^P \to \mathbb{R}$ and m is some mean function[3] $\mathbb{R}^P \to \mathbb{R}$. As an example, taking $k(\boldsymbol{w}, \boldsymbol{w}') \propto (\boldsymbol{w}^T\boldsymbol{w}')^p$ and restricting $\boldsymbol{w}$ to a hypersphere results in $\boldsymbol{\epsilon}$ taking the exact form of a spherical p-spin glass, studied previously for DNNs (Choromanska et al., 2015; Gardner and Derrida, 1988; Mezard et al., 1987; Ros et al., 2019; Mannelli et al., 2019; Baskerville et al., 2021a,b). *We are not* proposing to model the loss surface (batch or true) as a spin glass (or more generally, a Gaussian process), rather we are modelling the perturbation between the loss surfaces in this way. We emphasise that this model is a strict generalisation of the i.i.d. assumption above, and presents a rich, but tractable, model of isotropic Gaussian gradient perturbations in which the noise for different iterates is neither independent nor identically distributed.

Following from our Gaussian process definition, the gradient perturbations are jointly Gaussian and their covariance can be computed using a well-known result (see Adler and Taylor (2009) equation 5.5.4):

$$\text{Cov}(\epsilon_i(\boldsymbol{w}), \epsilon_j(\boldsymbol{w}')) = \partial_{w_i}\partial_{w'_j}k(\boldsymbol{w}, \boldsymbol{w}'). \tag{20}$$

Further assuming a stationary kernel $k(\boldsymbol{w}, \boldsymbol{w}') = k\left(-\frac{1}{2}||\boldsymbol{w} - \boldsymbol{w}'||_2^2\right)$

$$\text{Cov}(\epsilon_i(\boldsymbol{w}), \epsilon_j(\boldsymbol{w}')) = (w_i - w'_i)(w'_j - w_j)k''\left(-\frac{1}{2}||\boldsymbol{w} - \boldsymbol{w}'||_2^2\right) + \delta_{ij}k'\left(-\frac{1}{2}||\boldsymbol{w} - \boldsymbol{w}'||_2^2\right). \tag{21}$$

Thus we have a non-trivial covariance between gradient perturbation at different points in weight-space. This covariance structure can be used to prove the upcoming variance reduction result. Its proof relies on some technical Lemmas (proved in the Appendix, Section A.1) which we now state.

Lemma 3 *Let $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ be a sequence of jointly multivariate Gaussian random variables in $\mathbb{R}^P$ such that*

$$\boldsymbol{X}_i \;\mid\; \{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{i-1}\} \sim \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$$

where there exists a $\sigma > 0$ and a constant $A > 0$ such that $\det\Sigma_i \geq A\sigma^P$ for all P and i. Let also $\boldsymbol{X}_0$ be any deterministic element of $\mathbb{R}^P$. For $1 \leq m \leq n$, define the events

$$A_m(\delta) = \{||\boldsymbol{X}_i - \boldsymbol{X}_j||_2 > \delta \mid 0 \leq i < j \leq m\}.$$

Consider $P \to \infty$ with $P \gg \log n$ and let $\delta > 0$ be $o(P^{\frac{1}{2}})$ (note that δ and n need not diverge with P, but they can). Then $\mathbb{P}(A_n(\delta)) \to 1$ as $P \to \infty$.

3. It is natural to take $m = 0$ in a model for the sample perturbation, however retaining fully general m does not affect our arguments.

Lemma 4 *Assume the covariance structure (21). Take any $a_i \in \mathbb{R}$ and define $\bar{\boldsymbol{\epsilon}} = \sum_{i=1}^n a_i \boldsymbol{\epsilon}_i$, where $\boldsymbol{\epsilon}_i = \boldsymbol{\epsilon}(\boldsymbol{w}_i)$. Then*

$$\text{Tr } \text{Cov}(\bar{\boldsymbol{\epsilon}}) = k'(0)P\sum_{i=1}^n a_i^2 + 2P\sum_{1\leq i<j\leq n} a_i a_j \left[k'\Big(-\frac{d_{ij}^2}{2}\Big) + P^{-1}k''\Big(-\frac{d_{ij}^2}{2}\Big)d_{ij}^2\right] \tag{22}$$

where we define $d_{ij} = ||\boldsymbol{w}_i - \boldsymbol{w}_j||_2$.

Theorem 5 *Let $\boldsymbol{w}_n$ and $\boldsymbol{w}_{avg}$ be defined as in Theorem 2 and let the gradient perturbation be given by the covariance structure in (20). Assume that the kernel function k is such that $k(-x)$ and its derivatives decay at least as fast as $|x|^M e^{-x}$, for some $M > 0$, as $x \to \infty$ and define $\sigma^2 B^{-1} = k'(0)$. Assume further that $P^{1-\theta} \gg \log n$ for some $\theta \in (0,1)$. Let $\delta > 0$, then $\boldsymbol{w}_n$ and $\boldsymbol{w}_{avg}$ are multivariate Gaussian random variables and, with probability which approaches unity as $P, n \to \infty$ the iterates $\boldsymbol{w}_t$ are all mutually at least δ apart and*

$$\mathbb{E}w_{n,i} \sim e^{-\alpha\lambda_i n} w_{0,i}, \quad \frac{1}{P}\,\text{Tr}\,\text{Cov}(\boldsymbol{w}_n) \sim \frac{\alpha\sigma^2}{B}\left\langle \frac{1}{\lambda(2-\alpha\lambda)}\right\rangle, \tag{23}$$

$$\mathbb{E}w_{avg,i} \sim \frac{1-\alpha\lambda_i}{\alpha\lambda_i n} w_{0,i}, \frac{1}{P}\,\text{Tr}\,\text{Cov}(\boldsymbol{w}_{avg}) \leq \frac{\sigma^2}{Bn}\left\langle \frac{1}{\lambda}\right\rangle + \mathcal{O}(1)\left(k'\left(-\frac{\delta^2}{2}\right) + P^{-1}\delta^2 k''\left(-\frac{\delta^2}{2}\right)\right). \tag{24}$$

δ need not diverge with P (e.g. if could just be a fixed constant), but this result holds provided that $\delta = \mathcal{O}(P^{1/2})$, i.e. there exists a constant c independent of P such that $\delta/P^{1/2} \to c$ as $P \to \infty$. Clearly, the larger δ is, the smaller the δ-dependent terms above are.

Proof We will prove the result in the case $\lambda_i = \lambda\ \forall i$ for the sake of clarity. The same reasoning can be repeated in the more general case; where one gets $P^{-1}f(\lambda)\,\text{Tr}\,I$ below, one need only replace it with $\langle f(\lambda)\rangle$, exploiting linearity of the trace. We will also vacuously replace $\sigma^2 B^{-1}$ with σ^2 to save on notation. For weight iterates $\boldsymbol{w}_i$, we have the recurrence

$$\boldsymbol{w}_i = (1-\alpha\lambda)\boldsymbol{w}_{i-1} + \alpha\boldsymbol{\epsilon}(\boldsymbol{w}_{i-1})$$

which leads to

$$\boldsymbol{w}_n = (1-\alpha\lambda)^n \boldsymbol{w}_0 + \alpha\sum_{i=0}^{n-1}(1-\alpha\lambda)^{n-i-1}\boldsymbol{\epsilon}(\boldsymbol{w}_i) \tag{25}$$

and then

$$\boldsymbol{w}_{avg} = \frac{1-(1-\alpha\lambda)^n}{\alpha\lambda n}(1-\alpha\lambda)\boldsymbol{w}_0 + \sum_{i=0}^{n-1}\boldsymbol{\epsilon}(\boldsymbol{w}_i)\frac{1-(1-\alpha\lambda)^{n-i}}{\lambda n}. \tag{26}$$

As above, define $\boldsymbol{\epsilon}_i = \boldsymbol{\epsilon}(\boldsymbol{w}_i)$, for convenience. Now define

$$a_i = \alpha(1-\alpha\lambda)^{n-1-i}, \quad \bar{a}_i = \frac{1-(1-\alpha\lambda)^{n-i}}{\lambda n}.$$

Next we will apply Lemma 4 and utilise Lemma 3 to bound the variance of $\boldsymbol{w}_{avg}$ and $\boldsymbol{w}_n$. We first gather the following facts, which were also computed and used in the proof of Theorem 1:

$$\sum_{i=1}^{n-1} a_i^2 = \frac{\alpha^2(1-(1-\alpha\lambda)^{2n})}{1-(1-\alpha\lambda)^2} \tag{27}$$

$$\sum_{i<j} a_i a_j = \frac{\alpha}{\lambda}\left(\frac{1-(1-\alpha\lambda)^n}{\alpha\lambda} - \frac{1-(1-\alpha\lambda)^{2n}}{1-(1-\alpha\lambda)^2}\right). \tag{28}$$

The sum of squares for the $\bar{a}_i$ is simple to obtain similarly

$$\sum_{i=0}^{n-1} \bar{a}_i^2 = \frac{1}{\lambda^2 n^2}\left(n - \frac{2(1-(1-\alpha\lambda)^n)}{\alpha\lambda} + \frac{1-(1-\alpha\lambda)^{2n}}{1-(1-\alpha\lambda)^2}\right). \tag{29}$$

We now use the assumption that $0 < \alpha\lambda < 1$ (required for the convergence of gradient descent) which gives, as $n \to \infty$,

$$\sum_{i=1}^{n-1} a_i^2 \sim \frac{\alpha^2}{1-(1-\alpha\lambda)^2} \tag{30}$$

$$\sum_{i<j} a_i a_j \sim \frac{\alpha}{\lambda}\left(\frac{1}{\alpha\lambda} - \frac{1}{1-(1-\alpha\lambda)^2}\right) \tag{31}$$

$$\sum_{i=1}^{n-1} \bar{a}_i^2 \sim \frac{1}{\lambda^2 n} \tag{32}$$

Summing $\sum_{i<j} \bar{a}_i \bar{a}_j$ explicitly is possible but unhelpfully complicated. Instead, some elementary bounds give

$$\sum_{i<j} \bar{a}_i \bar{a}_j \le \left(\sum_{i=0}^{n-1} \bar{a}_i\right)^2 = \frac{1}{\lambda^2 n^2}\left(n - \frac{1-(1-\alpha\lambda)^n}{\alpha\lambda}\right)^2 \sim \frac{1}{\lambda^2}$$

and

$$\sum_{i<j} \bar{a}_i \bar{a}_j \ge \sum_{i<j}\left(\frac{1-(1-\alpha\lambda)^{n-1}}{\lambda n}\right)^2 \sim \frac{1}{2\lambda^2}$$

so in particular $\sum_{i<j} \bar{a}_i \bar{a}_j = \mathcal{O}(1)$. Now define the events $A_n(\delta)$ as in Lemma 3 using $\boldsymbol{\epsilon}_i$ in place of $\boldsymbol{X}_i$. Further, choose δ large enough so that $k'(-\frac{x^2}{2})$ and $x^2 k''(-\frac{x^2}{2})$ are decreasing for $x > \delta$. Define $k'(0) = \sigma^2$. Lemma 4 gives

$$\frac{1}{P}\operatorname{Tr}\operatorname{Cov}(\boldsymbol{w}_n) \mid A_n(\delta) \le \sigma^2 \sum_{i=1}^{n} a_i^2 + 2\sum_{i<j} a_i a_j \left(k'\Big(-\frac{\delta^2}{2}\Big) + P^{-1}\delta^2 k''\Big(-\frac{\delta^2}{2}\Big)\right) \tag{33}$$

where we note that we have only upper-bounded the second term in (33), so using (30) and (31) and taking δ large enough we obtain

$$\frac{1}{P}\operatorname{Tr}\operatorname{Cov}(\boldsymbol{w}_n) \mid A_n(\delta) = \frac{\sigma^2\alpha^2}{1-(1-\alpha\lambda)^2} + o(1). \tag{34}$$

Turning now to $\boldsymbol{w}_{avg}$ we similarly obtain

$$\frac{1}{P}\operatorname{Tr}\operatorname{Cov}(\boldsymbol{w}_{avg}) \mid A_n(\delta) \leq \frac{\sigma^2}{n}\frac{1}{\lambda^2} + \mathcal{O}(1)\left(k'\Big(-\frac{\delta^2}{2}\Big) + P^{-1}\delta^2 k''\Big(-\frac{\delta^2}{2}\Big)\right) \tag{35}$$

and, as before, taking δ large enough we can obtain

$$\frac{1}{P}\operatorname{Tr}\operatorname{Cov}(\boldsymbol{w}_{avg}) \mid A_n(\delta) = o(1). \tag{36}$$

Finally recalling (25) and (26) and writing $(1-\alpha\lambda)^n = \mathrm{e}^{-\alpha\lambda n} + o(1)$ for large n, we obtain the results in the statement of the theorem but conditional on the event $A_n(\delta)$. To complete the proof, we need only to establish that $\mathbb{P}(A_n(\delta)) \to 1$ $P, n \to \infty$, which we will do with an application of Lemma 3. Since the loss noise term is a Gaussian process, the $\boldsymbol{\epsilon}(\vec{w}_i)$ are all jointly Gaussian with the covariance structure (21), but to apply Lemma 3 we must further establish a lower bound on the covariance of the conditional $\boldsymbol{\epsilon}_i$. Let Σ_n be the $P \times P$ covariance matrix of $\boldsymbol{\epsilon}_n \mid \{\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_{n-1}\}$, then we are required to show that there exists some n-independent $A, \sigma > 0$ such that $\det \Sigma_n > A\sigma^{2P}$ for all n (subject to $\log n \ll P^{(1-\theta)}$). Define S_n to be the $nP \times nP$ covariance matrix of all of the $\{\boldsymbol{\epsilon}_i\}_{i=1}^n$, i.e.

$$(S_n)_{iP+j,kP+l} = \operatorname{Cov}\left(\epsilon_j(\boldsymbol{w}_j), \epsilon_l(\boldsymbol{w}_k)\right), \quad 0 \leq i,k < n,\ 1 \leq j,l \leq P,$$

and for convenience define $k'(0) = s^2$. The rules of standard Gaussian conditioning give

$$\Sigma_n = s^2 I - X_n S_{n-1}^{-1} X_n^T,$$

where X_n is the $P \times (n-1)P$ matrix such that S_n has the following block structure

$$S_n = \left(\begin{array}{c|c} S_{n-1} & X_n^T \\ \hline X_n & s^2 I_P \end{array}\right), \tag{37}$$

so, concretely, from (21)

$$(X_n)_{i,Pj+l} = \left((\boldsymbol{w}_n)_i - (\boldsymbol{w}_j)_i\right)\left((\boldsymbol{w}_j)_l - (\boldsymbol{w}_n)_l\right) k''\left(-\frac{1}{2}d_{jn}^2\right) + \delta_{il}k'\left(-\frac{1}{2}d_{jn}^2\right), \tag{38}$$

for $1 \leq i,l \leq P,\ 0 \leq j < n-1$. We can now Taylor expand the determinant

$$\begin{aligned}\det \Sigma_n &= s^{2P}\det\left(1 - s^{-2}X_n S_{n-1}^{-1}X_n^T\right) \\ &= s^{2P}\left(1 - s^{-2}\operatorname{Tr} X_n S_{n-1}^{-1}X_n^T\right) + \ldots\end{aligned}$$

which is valid provided that the trace term is small compared with 1. We have

$$|\operatorname{Tr} X_n S_{n-1}^{-1} X_n^T| \leq \operatorname{Tr} X_n X_n^T \|S_{n-1}^{-1}\|_{op} = \|X_n\|_F \|S_{n-1}^{-1}\|_{op}$$

where $\|\cdot\|_F, \|\cdot\|_{op}$ are the Frobenius and operator matrix norms respectively. Hence, it suffices to prove n, P-independent bounds $\|S_{n-1}^{-1}\|_{op} < q$ for some $q > 0$ and $\|X_n\|_F < c$ for some $0 < c < s^2/10$, say, valid for all n large enough, to thence obtain $\det \Sigma_n \geq c' s^{2P}$ for some constant $c' > 0$. Strictly speaking, one must use a bounded form of the remainder in Taylor's theorem to make precise all of these constants, but in reality we will see that we can make c as small as necessary, so that certainly $c' > 0$ exists and the bound $\det \Sigma_n \geq c' s^{2P}$ holds. Proceeding directly:

$$\begin{aligned}
\|X_n\|_F = \operatorname{Tr} X_n X_n^T &= \sum_{i,l=1}^{P} \sum_{j=0}^{n-2} (X_n)_{i,Pj+l}^2 \\
&= \sum_{j=0}^{n-2} \left\{ Pk'\left(-\frac{d_{jn}^2}{2}\right) - 2d_{jn}^2 k'\left(-\frac{d_{jn}^2}{2}\right) k''\left(-\frac{d_{jn}^2}{2}\right) + \left[d_{jn}^2 k''\left(-\frac{d_{jn}^2}{2}\right)\right]^2 \right\} \\
&\leq (n-1)\left(Pk'\left(-\frac{\delta^2}{2}\right) - 2\delta^2 k'\left(-\frac{\delta^2}{2}\right) k''\left(-\frac{\delta^2}{2}\right) + \left[\delta^2 k''\left(-\frac{\delta^2}{2}\right)\right]^2 \right),
\end{aligned}$$

but recall that we require $\delta = o(P^{1/2})$, so take for example $\delta = aP^{1/2-\varphi/2}$ for some $0 < \varphi < 1$, so

$$\|X_n\|_F \leq (n-1)\left(Pk'\left(-\frac{P^{1-\varphi}}{2}\right) - 2P^{1-\varphi} k'\left(-\frac{P^{1-\varphi}}{2}\right) k''\left(-\frac{P^{1-\varphi}}{2}\right) + \left[P^{1-\varphi} k''\left(-\frac{P^{1-\varphi}}{2}\right)\right]^2 \right).$$

Now recall that $xk'(-x)$ and $xk''(-x)$ are decaying for large enough x, and $\log n \ll P^{1-\theta}$, hence

$$\|X_n\|_F \leq (n-1)\left(2\log^{\frac{1}{1-\theta}} n k'\left(-\frac{\log^{\frac{1-\varphi}{1-\theta}} n}{2}\right) + \left[\log^{\frac{1-\varphi}{1-\theta}} n \;\; k''\left(-\frac{\log^{\frac{1-\varphi}{1-\theta}} n}{2}\right)\right]^2 \right).$$

Since $\theta > 0$, we can take some $0 < \varphi < \theta$ so that there exists $\chi \in (0,1)$ such that

$$\log^{\frac{1-\varphi}{1-\theta}} n > \log^{1+\chi} n \tag{39}$$

for large enough n, and so

$$\|X_n\|_F \leq (n-1)\left(2\log^{\frac{1}{1-\theta}} n k'\left(-\frac{\log^{1+\chi} n}{2}\right) + \left[\log^{1+\chi} n \;\; k''\left(-\frac{\log^{1+\chi} n}{2}\right)\right]^2 \right).$$

We assume that $k'(x), k''(x)$ decay at least as fast as $x^M e^{-x}$ for some $M > 0$ as $x \to \infty$, i.e. $k'(x)x^{-M}e^x \to 0$ (and similarly $k''(x)$). Writing $n - 1 \leq n = e^{\log n}$, we have

$$\|X_n\|_F \leq 2\log^{\frac{1}{1-\theta}} n k'\left(\log n - \frac{\log^{1+\chi} n}{2}\right) + \left[\log^{1+\chi} n \;\; k''\left(\log n - \frac{\log^{1+\chi} n}{2}\right)\right]^2,$$

but for large n, $\log^{1+\chi} n \gg \log n$ and so this last expression clearly converges to 0 as $n \to \infty$. Indeed, $e^{-\log^{1+\chi} n/2}$ decays faster than any fixed power of n, so the same is true of $\|X_n\|_F$. Hence we can find the constant $c > 0$ such that, for large enough $n > n_0$, say, $\|X_n\|_F < c$, as required. Now we turn to bounding $\|S_{n-1}^{-1}\|_{op}$, which is done by induction on n. Define the upper bounds $\|S_n^{-1}\|_{op} \leq q_n$ for all n. Recalling the block structure (37), we get the inverse

$$S_n^{-1} = \begin{pmatrix} (S_{n-1} - s^{-2}X_n^T X_n)^{-1} & 0 \\ 0 & \Sigma_n^{-1} \end{pmatrix} \begin{pmatrix} I & -s^{-2}X_n^T \\ -s^{-2}X_n & I \end{pmatrix} \equiv YZ.$$

$\|S_n^{-1}\|_{op}$ is bounded above by $\|X\|_{op}, \|Y\|_{op}$ and so we now bound these norms in turn. Since the off diagonals are zero, we have

$$\|Y\|_{op} \leq \max\{\|\Sigma_n^{-1}\|_{op}, \|(S_{n-1} - s^{-2}X_n^T X_n)^{-1}\|_{op}\}.$$

Recalling the expression for Σ_n above and expanding the matrix inverse

$$\begin{aligned}
\|\Sigma_n^{-1}\|_{op} &= s^{-2}\|(I - s^{-2}X_n S_{n-1}^{-1} X_n^T)^{-1}\|_{op} \\
&= s^{-2}\|(I + s^{-2}X_n S_{n-1}^{-1} X_n^T + s^{-4}(X_n S_{n-1}^{-1} X_n^T)^2 + \dots\|_{op} \\
&\leq s^{-2}\left(1 + s^{-2}\|X_n S_{n-1}^{-1} X_n^T\|_{op} + s^{-4}\|_{op}(X_n S_{n-1}^{-1} X_n^T)^2\|_{op} + \dots\right) \\
&\leq s^{-2}\left(1 + s^{-2}\|X_n\|_F\|S_{n-1}^{-1}\|_{op} + s^{-4}\|X_n\|_F^2\|S_{n-1}^{-1}\|_{op}^2 + \dots\right) \\
&\leq s^{-2}\left(1 + s^{-2}\|X_n\|_F\|q_{n-1} + s^{-4}\|X_n\|_F^2\|q_{n-1}^2 + \dots\right) \\
&\leq s^{-2}(1 + \alpha s^{-2} q_{n-1}\|X_n\|_F)
\end{aligned}$$

for some constant $\alpha > 0$, since we have already demonstrated that $\|X_n\|_F \to 0$ as $n \to \infty$. For the other term

$$\|(S_{n-1} - s^{-2}X_n^T X_n)^{-1}\|_{op} \leq \|S_{n-1}^{-1}\|_{op}\|(I - s^{-2}S_{n-1}^{-1}X_n^T X_n)^{-1}\|_{op}$$

from which point, one proceeds just as for $\|\Sigma_n^{-1}\|_{op}$ to obtain

$$\|(S_{n-1} - s^{-2}X_n^T X_n)^{-1}\|_{op} \leq q_{n-1}(1 + \alpha s^{-2} q\|X_n\|_F),$$

hence overall

$$\|Y\|_{op} \leq \max\{s^{-2}(1 + \alpha s^{-2} q_{n-1}\|X_n\|_F), q_{n-1}(1 + \alpha s^{-2} q_{n-1}\|X_n\|_F)\}.$$

We can always relax the bound on $\|S_{n-1}^{-1}\|_{op}$ so that $q_{n-1} > s^{-2}$, so we simply have $\|Y\|_{op} \leq q_{n-1}(1 + \alpha s^{-2} q_{n-1}\|X_n\|_F)$. To bound $\|Z\|_{op}$, we split it into a sum of two matrices

$$\|Z\|_{op} = \left\|\begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} + \begin{pmatrix} 0 & -s^{-2}X_n^T \\ -s^{-2}X_n & 0 \end{pmatrix}\right\|_{op} \leq 1 + 2s^{-2}\|X\|_{op} \leq 1 + 2s^{-2}\|X_n\|_F,$$

but $\|X_n\|_F \to 0$ as $n \to \infty$, so overall we can say

$$\|S_n^{-1}\|_{op} \leq q_{n-1}(1 + r_n), \quad r_n \equiv s^{-2}\|X_n\|_F(\alpha q_{n-1} + 2 + 2\alpha q_{n-1}\|X_n\|_F),$$

which we can simplify to

$$\|S_n^{-1}\|_{op} \leq q_{n-1}(1 + r'_n), \quad r'_n \equiv s^{-2}\|X_n\|_F(\alpha' q_{n-1} + 2)$$

and so can say

$$q_n = q_{n-1} + 2s^{-2}\|X_n\|_F q_{n-1} + s^{-2}\alpha'\|X_n\|_F q_{n-1}^2.$$

For large enough n, we seek a stability solution to this recurrence, i.e. using the ansatz $q_n = q + h_n$ for h_n small

$$q + h_n = q + h_{n-1} + 2s^{-2}\|X_n\|_F q + 2s^{-2}\|X_n\|_F h_{n-1} + s^{-2}\alpha'\|X_n\|_F(q^2 + 2qh_{n-1} + h_{n-1}^2). \tag{40}$$

Gathering the leading order terms gives

$$\begin{aligned} h_n &= h_{n-1} + 2s^{-2}q\|X_n\|_F + s^{-2}\alpha'\|X_n\|_F q^2 \\ \implies h_n &= h_{n_0} + s^{-2}q(2 + q\alpha') \sum_{j=n_0+1}^{n} \|X_j\|_F. \end{aligned}$$

Recall that $\|X_n\|_F$ decays faster than any fixed power of n, so the sum $\sum_{j\geq 2}\|X_j\|_F$ converges, hence for $\varepsilon > 0$ we can take some fixed n_0 large enough so that $\sum_{j=n_0+1}^{n}\|X_j\|_F < \varepsilon$ for all $n > n_0$. We are free to choose $h_{n_0} = 0$ and then for large enough n_0, we can guarantee $|h_n| < 1$, say, thus

$$q_n \leq \max\left\{\max_{1\leq m\leq n_0} q_m, q_{n_0} + 1\right\} \equiv q^*.$$

Hence we have succeeded in bounding $\|S_n^{-1}\|_{op} \leq q^*$ for all n. Combined with the earlier bound on $\|X_n\|_F$, we have now established the bound $\det\Sigma_n \geq c' s^{2P}$, so we have satisfied the conditions of Lemma 3 and completed the proof. ■

Note that Theorem 5 is a generalisation of Theorem 2 to the context of our dependent perturbation model. Let us make some clarifying remarks about the theorem and its proof:

1. The bound (24) in the statement of the theorem relies on *all* iterates being separated by a distance at least δ. Moreover, the bound is only useful if δ is large enough to ensure the k' and k'' terms are small.

2. Just as in the independent case of Theorem 2, the first term in the bound in (24) decays only in the case that the number of iterates $n \to \infty$.

3. The remaining conditions on P, n, δ are required for the high-dimensional probability argument which we use to ensure that all iterates are separated by at least δ.

4. $P^{1-\theta} \gg \log n$ is a perfectly reasonable condition in the context of deep learning. E.g. for a ResNet-50 with $P \approx 25 \times 10^6$, violation of this condition would require $n > 10^{10^7}$. A typical ResNet schedule on ImageNet has $< 10^6$ total steps.

Consequently, our result points to the importance of good separation between weight iterates in IA to retain the independence benefit and variance reduction in a non-independent noise setting, hence one would expect large learning rates to play a crucial role in successful IA. At the same time, our result is particularly adapted to the *deep learning limit* of very many model parameters ($P \to \infty$), since this is the only regime in which we can argue probabilistically for good separation of weight iterates (otherwise one may simply have to assume such separation). We note experimentally in Figure 4c that we do notice an increased improvement in using average as opposed to not doing so as a function of P. Furthermore, the importance of $P \gg \log n$ indicates that perhaps averaging less frequently than every iteration could be beneficial to generalisation. The following corollary makes this intuition precise.

Corollary 6 *Let $\boldsymbol{w}_{avg}$ now be a strided iterate average with stride κ, i.e.*

$$\boldsymbol{w}_{avg} = \frac{\kappa}{n} \sum_{i=1}^{\lfloor n/\kappa \rfloor} \boldsymbol{w}_i. \tag{41}$$

Then, under the same conditions as Theorem 5

$$\mathbb{E}w_{avg,i} = \frac{\kappa(1-\alpha\lambda_i)^{\kappa}}{n(1-(1-\alpha\lambda_i)^{\kappa})}(1+o(1))w_{0,i}, \tag{42}$$

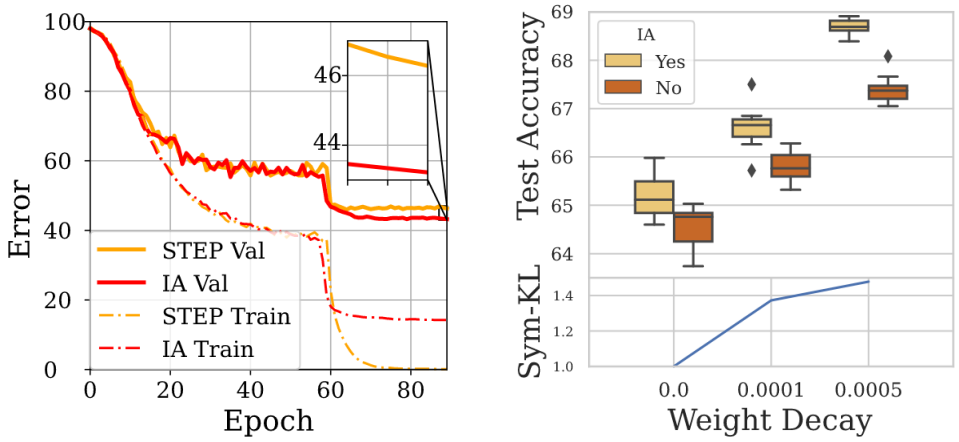
$$\frac{1}{P}\operatorname{Tr}\operatorname{Cov}(\boldsymbol{w}_{avg}) \leq \frac{\sigma^2\alpha^2\kappa}{Bn}\left\langle \frac{1}{(1-(1-\alpha\lambda)^{\kappa})^2}\frac{1-(1-\alpha\lambda)^{2\kappa}}{1-(1-\alpha\lambda)^2}\right\rangle + \mathcal{O}(1)\left(k'\Big(-\frac{\delta^2}{2}\Big) + P^{-1}\delta^2 k''\Big(-\frac{\delta^2}{2}\Big)\right) \tag{43}$$

where the constant $\mathcal{O}(1)$ coefficient of the second term in (43) is independent of κ.

Proof Very similar to that of Theorem 5. See Appendix Section A.2. ■

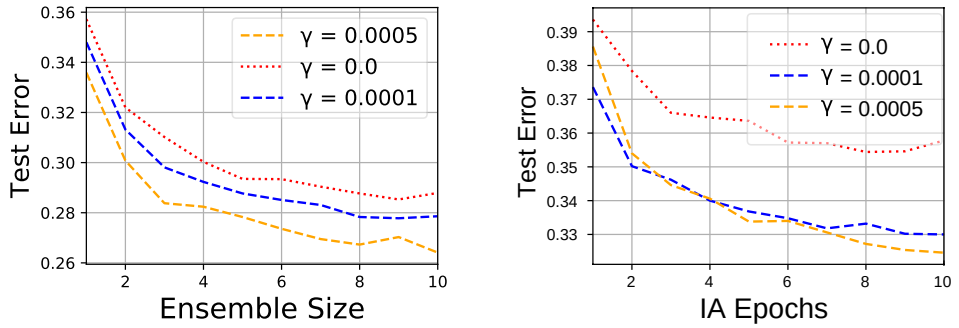
Intuitively, the first term in the covariance in (24) is an "independence term", i.e. it is common between Theorems 2 and 5 and represents the simple variance reducing effect of averaging. The second variance term in (24) comes from dependence between the iterate gradient perturbations. We see from the corollary that an independent model for gradient perturbation would predict an unambiguous inflationary effect of strided IA on variance (the first term in (43)). However introducing dependence in the manner that we have predicts a more nuanced picture, where increased distance between weight iterates can counteract the simple "independent term" inflationary effect of striding, leaving open the possibility for striding to improve on standard IA for the purposes of generalisation. We investigate and experimentally confirm this hypothesis in Section 7.1.

Validation of theory. To better understand the effect of the large learning learning rate on generalisation, we train a VGG-16 network (Simonyan and Zisserman, 2014) with no data augmentation/batch normalisation (to isolate the overfitting effect from reducing the learning rate) with a learning rate of $\alpha = 0.05$. Replacing the learning rate drop (performed at epoch 60 by a factor of 10 with weight decay $\gamma = 0.0005$) with IA at the same point, we find that the test error is reduced by a greater margin ($\approx 2\%$), shown in Figure 1a. We note that IA improves over the SGD learning rate equivalent for all values of weight decay,



(a) IA vs. STEP schedule (b) Impact of weight decay

Figure 1: (a) STEP (learning rate decay) and IA Train/Val error. Both approaches reduce Val error, but IA by a greater margin. (b) Effect of weight decay on held out test error for IA/sharp learning rate decay solutions. Greater weight decay increases the margin of IA improvement. The lower subplot shows the average symmetric KL-divergence between IA solutions.



(a) Network Ensemble (b) Iterate Averaging

Figure 2: Test error improvement with differing degrees of regularisation γ for (a) network ensembling and (b) IA.

with results for 10 seeds shown in Fig 1b and hence this argument is independent of explicit regularisation as indicated by Theorem 5. For our Deep Neural Network experiments, we find that the best IA optimiser improvement over its base optimiser is proportional to the number of parameters P as shown in Fig 4c and predicted by Theorem 2. This theorem, and experimental validation thereof, translates into the following advice for practitioners: *In the deep learning limit (large P and small relative B) one should keep the learning rate high and use iterate averaging instead of sharply dropping the learning rate!*

3.3 A Closer Look at IA and the Importance of Regularisation

Izmailov et al. (2018) argue, under a linearisation assumption, that IA can be seen as approximate model ensembling. Since averaging only improves test performance for sufficiently uncorrelated models (through a reduction in variance of the ensemble), we must ensure sufficiently diverse models at each epoch through our training procedure. We note from Fig 2b, that unlike model ensembling (shown in Fig 2a), the IA improvement is strongly dependent on the use of weight decay. We show the difference between IA and sharply decaying learning rate schedules (which mirror conventional setups) over 10 seeds as a function of weight decay coefficient in Fig 1b. The margin of improvement from IA, over sharply decaying schedules, steadily increases with regularisation - with $\gamma = 0.0005$ (where γ denotes the amount of weight decay or the coefficient of L_2 regularisation in the loss) delivering a greater final validation accuracy at the final IA point, despite starting from a lower accuracy compared to $\gamma = 0.0001$. To explicitly show that such weight decay regularisation encourages greater diversity in the iterates we calculate the symmetrised KL-divergence, $\frac{1}{2}\Big(\sum p(x)\log\frac{p(x)}{q(x)} + q(x)\log\frac{q(x)}{p(x)}\Big)$, over the entire test set between the softmax outputs of the IA iterates. We take an average for each weight decay value (normalising the 0 weight decay value to 1), as shown in the lower subfigure of Fig 1b. As expected, greater weight decay gives greater solution diversity.

Distance in weight space or relative distance? Theorem 5 relies on sufficiently large distances in weight-space between iterates to achieve variance reduction with IA. It is possible to drive the second term in (24) to zero by allowing the distance between iterates to diverge. Of course, this requires the L_2 norm of most of the iterates to diverge also, which is not necessarily plausible in a real network. We have observed above that weight decay combines advantageously with iterate averaging and a natural interpretation of weight decay's effect is that it places some constraint on the norm of the weight iterates, which suggests a tension between the our observations about weight decay and the explanation of iterate averaging in Theorem 5. Indeed, consider n iterates $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n$ and assume there exists some $R > 0$ such that $\|\boldsymbol{w}_i\|_2 \leq R$ for all i. Define $\delta > 0$ such that $\|\boldsymbol{w}_i - \boldsymbol{w}_j\|_2 \geq \delta$ for all i, j. The balls of radius δ centred at each of the $\boldsymbol{w}_i$ must lie within the sphere of radius $R + \delta$ in order that the bounds on $\|\boldsymbol{w}_i\|_2$ and on $\|\boldsymbol{w}_i - \boldsymbol{w}_j\|_2$ all be satisfied. Hence we have the bound

$$n^P \delta \leq (R + \delta)^P \implies \delta \leq \frac{R}{n^{1/P} - 1}. \tag{44}$$

Let us consider the case of sufficiently large but still finite n and P, rather than attempting an asymptotic analysis as in Theorem 5, since this is more closely related to practical training. Recalling (24), for large enough n, P we have the bound

$$\begin{aligned}
&\frac{1}{P}\operatorname{Tr}\operatorname{Cov}(\boldsymbol{w}_{avg}) \\
\leq &\frac{\sigma^2}{Bn}\left\langle\frac{1}{\lambda}\right\rangle + \mathcal{O}(1)\left(k'\left(-\frac{\delta^2}{2}\right) + P^{-1}\delta^2 k''\left(-\frac{\delta^2}{2}\right)\right) \\
\leq &\frac{\sigma^2}{Bn}\left\langle\frac{1}{\lambda}\right\rangle + \mathcal{O}(1)\left(k'\left(-\frac{R^2}{2(n^{1/P}-1)^2}\right) + P^{-1}\frac{R^2}{(n^{1/P}-1)^2}k''\left(-\frac{R^2}{2(n^{1/P}-1)^2}\right)\right).
\end{aligned} \tag{45}$$

This bound immediately demonstrates the competition between the two variance terms for IA. The first term strictly decreases as the number of iterates averaged over increases, while the second term grows, since larger n means that the largest possible δ is decreased which leads to an increased error due to non-independence of the averaged iterates. If we fix P and R, this expression implies that the variance of the averaged iterates is not minimised by taking n as large as possible but at some intermediate optimal value, since the second term in (45) increases to some finite non-zero limit as n increases. It is reasonable to suppose that some bound $\|\vec{w}_i\|_2 \leq R \ \forall i$ exists for practical networks. In that case, we can see immediately that smaller R directly increases the variance of the IA weights. Given the standard interpretation of weight decay as essentially further constraining the weight norms and decreasing R, it is natural to question why weight decay should help at all when combined with IA. For learning rate α and weight decay γ, we move a distance $\alpha\nabla L(\boldsymbol{w}) - \alpha\gamma\boldsymbol{w}$ in weight space. Intuitively, since random vectors in high dimensions are nearly orthogonal with high probability (Vershynin, 2018), we expect the distance in weight space to move a distance $\alpha\sqrt{(\nabla L(\boldsymbol{w}))^2 + \gamma^2\boldsymbol{w}^2}$, which is larger for $\gamma > 0$. Conversely, we expect $||\boldsymbol{w}||^2$ to be smaller for smaller γ and the gradients also to be smaller (Granziol, 2020b). For the experiment (with equal learning rates) shown in Fig 1b, the average distances between the IA epochs for weight decay values $\gamma = \{0, 0.0001, 0.0005\}$ are $17.7, 14.9, 13.9$, respectively. We note, however, that the relative distance, when normalised by the average weight norm, increases successively as $0.11, 0.13, 0.22$. This begs the question whether the above considerations and Theorem 5 can be extended in any way to use a notion of relative distance in weight space. For a large enough number of weight iterates, the mean L_2 norm will converge with high probability on some deterministic value $r \leq R$. Let us define relative distance by $\Delta(\boldsymbol{w}, \boldsymbol{w}') = \frac{\|\boldsymbol{w}-\boldsymbol{w}'\|_2}{r}$, and so $\Delta(\boldsymbol{w}_i, \boldsymbol{w}_j) \geq \delta$ for all i, j holds if $\|\boldsymbol{w}_i - \boldsymbol{w}_j\|_2 \geq \delta r$ for all i, j. Hence we have the following bound on δ, as above,

$$\delta \leq \frac{1}{n^{1/P} - 1}\frac{r}{R}. \tag{46}$$

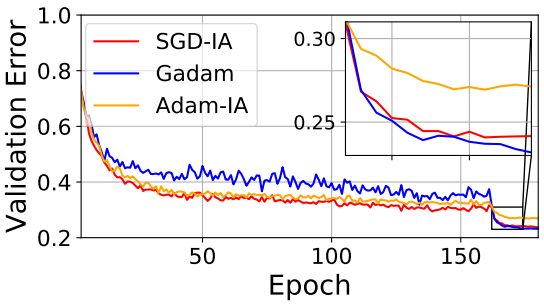
Now, as weight decay is varied, it is reasonable to suppose that r and R vary similarly. Indeed R can be interpreted as the length scale of the weight iterates and one expects r to be of the same scale, hence the ratio r/R is independent of the length scale of the weight iterates. With high probability, we can say that for large n, r/R has approximately the value c and this constant is independent of the overall weight scale R. So there is an important distinction between distance and relative distance, namely that the maximal possible pairwise distance δ scales linearly with the typical length scale of the weight iterates, but this is not the case for the *relative* distance.

$$\begin{aligned}&\frac{1}{P}\operatorname{Tr}\operatorname{Cov}(\boldsymbol{w}_{avg})\\ \leq&\frac{\sigma^2}{Bn}\left\langle\frac{1}{\lambda}\right\rangle + \mathcal{O}(1)\left(k'\left(-\frac{c^2}{2(n^{1/P}-1)^2}\right) + P^{-1}\frac{c^2}{(n^{1/P}-1)^2}k''\left(-\frac{c^2}{2(n^{1/P}-1)^2}\right)\right).\end{aligned} \tag{47}$$

Under this framework it is clear that we can extend the notion of relating independence of the iterates to distance in relative (scaled by the constant r), not absolute weight space. This generalises the theoretical framework to include weight decay, which reduces the movement in absolute weight space. As shown experimentally, increase in relative distance between the

iterates, increases the generalisation impact of iterate averaging, in line with our theoretical predictions.

4 Adaptive Gradient Methods with Iterate Averaging



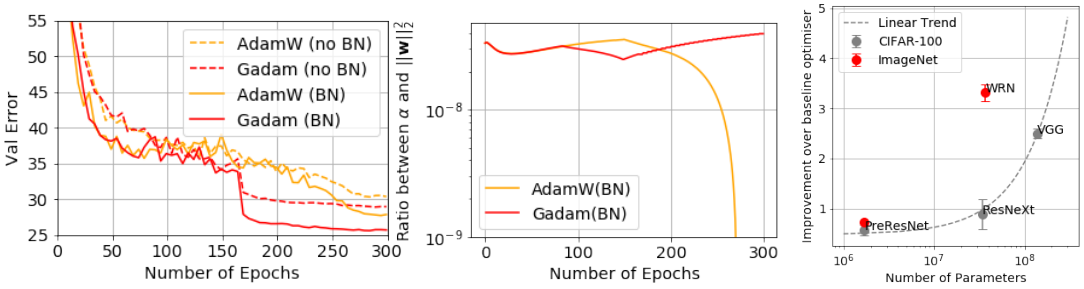
(a) PRN-110 Val. Error

Optim	SGD-IA	Adam-IA	Gadam
$\mathcal{D}(p\|\|q)$	20.2	19.1	22.8
$\|\|p-q\|\|^2$	18.2	18.0	18.6
ΔE	9.0	5.8	11.9

Values except ΔE in units of 1000.

(b) Solution diversity

Figure 3: (a) Validation error for the PRN-110 on CIFAR-100 for various optimisers using IA and (b) the solution diversity given as the symmetrised KL $\mathcal{D}$ or total variation distance calculated on the test set and the change in validation error ΔE for the final IA point.



(a) Val Error

(b) $\alpha/||\boldsymbol{w}||^2$

(c) Improvement vs. P

Figure 4: (a) Val. error and (b) effective learning rate $\frac{\alpha}{||\boldsymbol{w}||^2}$ of VGG-16 on CIFAR-100 with and without BN. (c) Improvement in using IA over the base optimiser for networks over different parameter counts.

Naïvely combining IA with Adam is not effective, as shown in Figure 3a. Despite the same L_2 regularisation (0.0001), the error drop is significantly less than for SGD-IA. Following our intuition from Sec 3.3, we consider whether the problem could be that overly correlated solutions form the IA due to ineffective regularisation. As shown in Tab. (b) of Figure 3, both the symmetrised KL divergence $\mathcal{D}(p||q)$ and total variation distance $||p-q||^2$ (calculated between all epochs using IA at the end of training and then averaged) are lower for Adam-IA than for SGD-IA. For adaptive optimisers, L_2 regularisation is not equivalent to weight decay (Zhang et al., 2018; Loshchilov and Hutter, 2019), with weight decay generalising better - known as *AdamW*. For AdamW with a decoupled weight decay of 0.25, the solution diversity increases beyond that of SGD-IA. This is accompanied by a greater drop in validation error,

even outperforming SGD-IA. We term this combination of AdamW + IA *Gadam* to denote a variant of Adam that generalises. Previous work has shown that limiting the belief in the Adam curvature matrix improves generalisation (Zhuang et al., 2020; Chen and Gu, 2018), hence we also incorporate such a partially adaptive Adam into our framework and term the resulting Algorithm *GadamX*.

For convolutional neural networks using batch normalisation, the effective learning rate is proportional to $\alpha_{\text{eff}} \propto \frac{\alpha}{||\boldsymbol{w}||^2}$ (Hoffer et al., 2018). With batch normalisation, the output is invariant to the channel weight norm, hence weight changes are only with respect to the direction of the vector. Since the effective weight decay depends on the (effective) learning rate, we expect this to lead to more regularised solutions and better validation error. To test this hypothesis, we train a VGG-16 network on CIFAR-100 with and without BN (see Figs 4a,4b): for an identical setup, the margin of improvement of Gadam over AdamW is much larger with BN. We note that while Gadam keeps the effective learning rate $\frac{\alpha}{||\boldsymbol{w}||^2}$ high, in scheduled AdamW it quickly vanishes once we start learning rate decay. We were unable to compensate with learning rate scheduling, underscoring the importance of appropriate weight decay.

Algorithm 1 Gadam/GadamX

Require: initial weights θ_0; learning rate scheduler $\alpha_t = \alpha(t)$; momentum parameters $\{\beta_1, \beta_2\}$; partially adaptive parameter $p \in [0, 0.5]$ Default to $\{0.125, 0.5\}$ for {GadamX, Gadam}; decoupled weight decay γ; averaging starting point T_{avg}; tolerance ϵ (10^{-8})

Ensure: Optimised weights $\tilde{\theta}$

Set $\boldsymbol{m}_0 = 0, \boldsymbol{v}_0 = 0, \hat{\boldsymbol{v}}_0 = 0, n_{\text{models}} = 0$.

for t = 1, ... T **do**

$\alpha_t = \alpha(t)$

$\boldsymbol{g}_t = \nabla f_t(\theta_t - 1)$

$\boldsymbol{m}_t = \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t / (1 - \beta_1^t)$

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

$\hat{\boldsymbol{v}}_t = \max(\hat{\boldsymbol{v}}_{t-1}, \hat{\boldsymbol{v}}_t)$ (If using Amsgrad (Reddi et al., 2019))

$\theta_t = (1 - \alpha_t \gamma)\theta_{t-1} - \alpha_t \frac{\hat{\boldsymbol{m}}_t}{(\hat{\boldsymbol{v}}_t + \epsilon)^p}$

if $T \geq T_{\text{avg}}$ **then**

$n_{\text{models}} = n_{\text{models}} + 1$

$\theta_{\text{avg}} = \frac{\theta_{\text{avg}} \cdot n_{\text{models}} + \theta_t}{n_{\text{models}} + 1}$

else

$\theta_{\text{avg}} = \theta_t$

end if

end for

return $\tilde{\theta} = \theta_{\text{avg}}$

Here we present the full Gadam/GadamX algorithm. Note that for simplicity, in Algorithm 1, we present a Polyak-style averaging of every iteration. In practice we find both practical and theoretical results suggesting that averaging *less* frequently is almost equally good, if not better. We discuss this in Corollary 6 and conduct experiments on the averaging frequency in Section 7.1.

5 Extension of the Theoretical Framework to Weight Decay and Adaptive Methods

To make a closer connection with the new optimisation algorithms proposed in this work we consider decoupled weight decay (strength γ) and gradient preconditioning:

$$\boldsymbol{w}_t = (1 - \alpha\gamma)\boldsymbol{w}_{t-1} - \alpha \tilde{\boldsymbol{H}}_t^{-1} \nabla L_{batch}(\boldsymbol{w}_{t-1}) \tag{48}$$

where $\tilde{\boldsymbol{H}}_t^{-1}$ is some approximation to the true loss Hessian used at iteration t. In the presence of weight decay, we move the true loss minimum away from the origin for the analysis, i.e. $L_{\text{true}}(\boldsymbol{w}) = (\boldsymbol{w} - \boldsymbol{w}^*)^T \boldsymbol{H} (\boldsymbol{w} - \boldsymbol{w}^*)$. The update rule is then

$$\boldsymbol{w}_t = \left(1 - \alpha\gamma - \alpha \tilde{\boldsymbol{H}}_t^{-1} \boldsymbol{H}\right) \boldsymbol{w}_{t-1} + \alpha \boldsymbol{H} \boldsymbol{w}^* + \alpha \boldsymbol{\epsilon}(\boldsymbol{w}_{t-1}). \tag{49}$$

We take $\tilde{\boldsymbol{H}}_t^{-1}$ to be diagonal in the eigenbasis of $\boldsymbol{H}$, with eigenvalues $\tilde{\lambda}_i^{(t)} + \varepsilon$, where ε is the standard tolerance parameter (Kingma and Ba, 2014). One could try to construct the $\tilde{\boldsymbol{H}}_t^{-1}$ from the Gaussian process loss model, so making them stochastic and covarying with the gradient noise, however we do not believe this is tractable. Instead, let us heuristically assume that, with high probability, $\tilde{\lambda}_i^{(t)}$ is close to λ_i, say within a distance ζ, for large enough t and all i. If we take a large enough ζ this is true even for SGD and we expect Adam to better approximate the local curvature matrix than SGD (Granziol et al., 2020a). This results in the following theorem.

Theorem 7 *Fix some $\zeta > 0$ and assume that $|\tilde{\lambda}_i^{(t)} - \lambda_i| < \zeta$ for all $t \geq n_0$, for some fixed $n_0(\zeta)$. Use the update rule (49). Assume also that $\langle 1/\lambda \rangle > 1$. Let everything else be as in Theorem 5. Then*

$$\mathbb{E}w_{n,i} \sim e^{-\alpha(1+O(1)(\epsilon+\zeta+\gamma)\lambda_i^{-1})n} w_{0,i} + w_i^* \tag{50}$$

$$\left| \frac{1}{P} \operatorname{Tr} \operatorname{Cov}(\boldsymbol{w}_n) - \frac{\alpha\sigma^2}{B} \left\langle \frac{1}{(2-\alpha)} \right\rangle \right| \leq \mathcal{O}(1)(\epsilon+\zeta+\gamma) \left\langle \frac{1}{\lambda} \right\rangle, \tag{51}$$

$$\mathbb{E}w_{avg,i} \sim \frac{1-\alpha(1+O(1)(\epsilon+\zeta+\gamma))\lambda_i^{-1}}{\alpha(1+O(1)(\epsilon+\zeta+\gamma)\lambda_i^{-1})n} w_{0,i} + w_*^i., \tag{52}$$

$$\left| \frac{1}{P} \operatorname{Tr} \operatorname{Cov}(\boldsymbol{w}_{avg}) - \frac{\sigma^2}{Bn} \right| \leq \mathcal{O}(1) \left(k'\Big(-\frac{\delta^2}{2}\Big) + P^{-1}\delta^2 k''\Big(-\frac{\delta^2}{2}\Big) \right) + \mathcal{O}(1)(\epsilon+\zeta+\gamma) \left\langle \frac{1}{\lambda} \right\rangle. \tag{53}$$

Proof The results follows almost immediately from Theorem 5 by making some replacements due to the modified update rule. Recall the basic update rule used in Theorem 5:

$$\boldsymbol{w}_t = (1 - \alpha\boldsymbol{H})\boldsymbol{w}_{t-1} + \alpha\boldsymbol{\epsilon}(\boldsymbol{w}_{t-1}).$$

Comparing with (49), one sees that we must replace $\boldsymbol{H}$ with $\tilde{\boldsymbol{H}}_t^{-1}\boldsymbol{H} + \gamma I$ and $\boldsymbol{\epsilon}(\boldsymbol{w}_{t-1})$ with $\boldsymbol{\epsilon}(\boldsymbol{w}_{t-1}) + \boldsymbol{H}\boldsymbol{w}^*$. Based on the assumption about the accuracy of the eigenvalues of $\tilde{\boldsymbol{H}}$, we find for all $t \geq n_0$

$$\frac{\lambda_i}{\tilde{\lambda}_i^{(t)} + \varepsilon} = \frac{\lambda_i}{\lambda_i + \tilde{\lambda}_i^{(t)} - \lambda_i + \varepsilon} < \frac{\lambda_i}{\lambda_i + \varepsilon - \zeta} < 1 + |\varepsilon - \zeta|\lambda_i^{-1}$$

and

$$\frac{\lambda_i}{\tilde{\lambda}_i^{(t)} + \varepsilon} = \frac{\lambda_i}{\lambda_i + \tilde{\lambda}_i^{(t)} - \lambda_i + \varepsilon} > \frac{\lambda_i}{\lambda_i + \varepsilon + \zeta} > 1 - (\varepsilon + \zeta)\lambda_i^{-1}$$

where the final inequality in each case can be derived from Taylor's theorem with Lagrange's form of the remainder (Shirali and Vasudeva, 2014). Since the λ_i are bounded away from zero, we have established

$$\begin{aligned} &\left| \frac{\lambda_i}{\tilde{\lambda}_i^{(t)} + \varepsilon} - 1 \right| < (\varepsilon + \zeta)\lambda_i^{-1}, \\ \implies &\left| \frac{\lambda_i}{\tilde{\lambda}_i^{(t)} + \varepsilon} + \gamma - 1 \right| < (\varepsilon + \zeta + \gamma)\lambda_i^{-1}, \end{aligned} \tag{54}$$

where the second step follows from the assumption $\left\langle \frac{1}{\lambda} \right\rangle > 1$. Using this result alone we can immediately write down from Theorem 5 the covariance results

$$\left| \frac{1}{P} \operatorname{Tr} \operatorname{Cov}(\boldsymbol{w}_n) - \frac{\alpha\sigma^2}{B} \left\langle \frac{1}{(2-\alpha)} \right\rangle \right| \leq \mathcal{O}(1)(\epsilon + \zeta + \gamma) \left\langle \frac{1}{\lambda} \right\rangle, \tag{55}$$

$$\left| \frac{1}{P} \operatorname{Tr} \operatorname{Cov}(\boldsymbol{w}_{avg}) - \frac{\sigma^2}{Bn} \right| \leq \mathcal{O}(1) \left(k'\Big(-\frac{\delta^2}{2} \Big) + P^{-1}\delta^2 k''\Big(-\frac{\delta^2}{2} \Big) \right) + \mathcal{O}(1)(\epsilon + \zeta + \gamma) \left\langle \frac{1}{\lambda} \right\rangle. \tag{56}$$

Consider now the mean results from Theorem 5. Again, we can write down the results with the modified update rule:

$$\mathbb{E}w_{n,i} \sim e^{-\alpha(1+O(1)(\epsilon+\zeta+\gamma)\lambda_i^{-1})n} w_{0,i} + w_i^*, \quad \mathbb{E}w_{avg,i} \sim \frac{1 - \alpha(1 + O(1)(\epsilon + \zeta + \gamma))\lambda_i^{-1}}{\alpha(1 + O(1)(\epsilon + \zeta + \gamma)\lambda_i^{-1})n} w_{0,i} + w_*^i.$$

■

Theorem 7 demonstrates the same IA variance reduction as seen previously, but in the more general context of weight decay and adaptive optimisation. As expected, improved estimation of the true Hessian eigenvalues (i.e. smaller ζ) reduces the error in recovery of $\boldsymbol{w}^*$. Moreover, increasing the weight decay strength γ decreases the leading order error bounds in (50) and (52), but only up to a point, as the other error terms are valid and small only if γ is not too large.

Note also that Theorem 7 can be applied equally well to the case of IA with plain SGD (as was addressed in Theorem 5. In that case, the pre-conditioning matrix $\tilde{\boldsymbol{H}}$ is simply the identity and $\gamma = 0$. It follows that $\zeta > \max_i |\lambda_i - 1|$ and so the error terms in (50-53) are larger compared to the case of a non-identity $\tilde{\boldsymbol{H}}$ provided that the estimation of the λ_i by $\tilde{\lambda}_i$ is at least good enough to reduce ζ below $\max_i |\lambda_i - 1|$.

Now consider the more general case where the eigenbases of $\tilde{\boldsymbol{H}}$ and $\boldsymbol{H}$ do not align. The eigenbasis of $\tilde{\boldsymbol{H}}$ is just the coordinate basis $\{\boldsymbol{e}_i\}$ and we denote the eigenbasis of $\boldsymbol{H}$ by $\{\boldsymbol{\phi}_i\}_i$. Consider the term $\left\langle \frac{1}{\lambda} \right\rangle$ which appears in Theorem 5 and is replaced by $P^{-1} \operatorname{Tr} \tilde{\boldsymbol{H}}^{-1}\boldsymbol{H} + \gamma I$ in Theorem 7. We have

$$\begin{aligned} P^{-1} \operatorname{Tr} \tilde{\boldsymbol{H}}^{-1}\boldsymbol{H} &= P^{-1} \operatorname{Tr} \sum_{i,j=1}^{P} (\tilde{\lambda}_i + \epsilon)^{-1} \boldsymbol{e}_i \boldsymbol{e}_i^T \lambda_j \boldsymbol{\phi}_j \boldsymbol{\phi}_j^T \\ &= P^{-1} \sum_{i,j=1}^{P} (\tilde{\lambda}_i + \epsilon)^{-1} \lambda_j (\boldsymbol{e}_i^T \boldsymbol{\phi}_j)^2, \end{aligned}$$

but $P^{-1} \sum_{i,j} (\boldsymbol{e}_i^T \boldsymbol{\phi}_j)^2 = 1$, so

$$\left| P^{-1} \operatorname{Tr}(\tilde{\boldsymbol{H}}^{-1}\boldsymbol{H} + \gamma I) - 1 \right| = P^{-1} \left| \sum_{i,j=1}^{P} \left[(\tilde{\lambda}_i + \epsilon)^{-1} \lambda_j + \gamma - 1 \right] (\boldsymbol{e}_i^T \boldsymbol{\phi}_j)^2 \right|.$$

Now define $\varphi_{ij} = \boldsymbol{e}_i^T \boldsymbol{\phi}_j$ and then

$$\left|P^{-1}\operatorname{Tr}(\tilde{\boldsymbol{H}}^{-1}\boldsymbol{H}+\gamma I)-1\right| = P^{-1}\left|\sum_{i,j=1}^{P}\left[(\tilde{\lambda}_i+\epsilon)^{-1}\lambda_j+\gamma-1\right]\varphi_{ij}^2\right|$$
$$\leq P^{-1}\left|\sum_{i\neq j}^{P}\left[(\tilde{\lambda}_i+\epsilon)^{-1}\lambda_j+\gamma-1\right]\varphi_{ij}^2\right| + P^{-1}(\epsilon+\zeta+\gamma)\sum_{i=1}^{P}\frac{\varphi_{ii}^2}{\lambda_i}.$$

The second of these terms is obviously decreasing in ζ, as in Theorem 7. In the worst case, that the eigenspaces are in general (random) position relative to each other, the values φ_{ij} are those of a $P \times P$ Haar distributed orthogonal random matrix Anderson et al. (2010) and it is known that the rows (or columns) of such a matrix are "delocalised", i.e. all entries are of size $O(P^{-1/2})$ with high probability. In that case, we have the rough equivalence

$$\left|P^{-1}\operatorname{Tr}(\tilde{\boldsymbol{H}}^{-1}\boldsymbol{H}+\gamma I)-1\right| \sim P^{-2}\left|\sum_{i,j=1}^{P}\left[(\tilde{\lambda}_i+\epsilon)^{-1}\lambda_j+\gamma-1\right]\right|$$
$$\sim \left|\langle\lambda\rangle\left\langle(\tilde{\lambda}+\epsilon)^{-1}\right\rangle+\gamma-1\right|$$
$$= \left|\operatorname{Tr}\boldsymbol{H}\operatorname{Tr}(\tilde{\boldsymbol{H}}+\epsilon)^{-1}+\gamma-1\right|$$

which is independent of the accuracy of the eigenvalue approximation ζ. Overall, we see that some better than random estimation of the Hessian $\boldsymbol{H}$ can be expected to yield superior error bounds in the style of Theorem 7 than obtained with SGD alone. In the typical worst case of randomly aligned eigenspaces, the bounds are not expected to be any worse than for SGD.

6 Experiments

6.1 Image Classification on CIFAR and Down-sampled 32x32 ImageNet Datasets

Here we consider VGG-16, Preactivated ResNet (PRN) and ResNeXt (Simonyan and Zisserman, 2014; He et al., 2016b; Xie et al., 2017) on CIFAR datasets (Krizhevsky et al., 2009). We also considered the down-sampled ImageNet dataset (Russakovsky et al., 2015) on Wide Residual Networks.

Learning rate schedule. For all experiments without IA, we use the following learning rate schedule for the learning rate at the t-th epoch, similar to Izmailov et al. (2018), which we find to perform better than the conventionally employed step scheduling (refer to the experimental details in Appendix Section D.4):

$$\alpha_t = \begin{cases} \alpha_0, & \text{if } \frac{t}{T} \leq 0.5 \\ \alpha_0\left[1-\frac{(1-r)(\frac{t}{T}-0.5)}{0.4}\right] & \text{if } 0.5 < \frac{t}{T} \leq 0.9 \\ \alpha_0 r, & \text{otherwise} \end{cases} \tag{57}$$

where α_0 is the initial learning rate. In the motivating logistic regression experiments on MNIST, we used $T = 50$. $T = 300$ is the total number of epochs budgeted for all CIFAR

experiments, whereas we used $T = 200$ and 50 respectively for PRN-110 and WideResNet (WRN) 28×10 in ImageNet. We set $r = 0.01$ for all experiments. For experiments with iterate averaging, we use the following learning rate schedule instead:

$$\alpha_t = \begin{cases} \alpha_0, & \text{if } \frac{t}{T_{\text{avg}}} \leq 0.5 \\ \alpha_0 \Big[1 - \frac{(1-\frac{\alpha_{\text{avg}}}{\alpha_0})(\frac{t}{T}-0.5)}{0.4}\Big] & \text{if } 0.5 < \frac{t}{T_{\text{avg}}} \leq 0.9 \\ \alpha_{\text{avg}}, & \text{otherwise} \end{cases} \tag{58}$$

where α_{avg} refers to the (constant) learning rate after iterate averaging activation, and in this paper we set $\alpha_{\text{avg}} = \frac{1}{2}\alpha_0$. T_{avg} is the epoch after which iterate averaging is activated, and the methods to determine T_{avg} was described in the main text. This schedule allows us to adjust learning rate smoothly in the epochs leading up to iterate averaging activation through a similar linear decay mechanism in the experiments without iterate averaging, as described above.

The only exception is the WRN experiments on ImageNet 32×32, where we only run 50 epochs of training and start averaging from 30th epoch. We found that when using the schedule described above for the IA schedules (SWA/Gadam/GadamX), we start decay the learning rate too early and the final result is not satisfactory. Therefore, for this particular set of experiments, we use the same learning rate schedule for both averaged and normal optimisers. The only difference is that for IA experiments, we decay the learning rate until the 30th epoch and keep it fixed for the rest of the training.

Hyperparameter tuning. In CIFAR experiments, we tune the base optimisers (i.e. SGD, Adam(W), Padam(W)) only, and assuming that the ideal hyperparameters in base optimisers apply to IA, and apply the same hyperparameter setting for the corresponding IA optimisers (i.e. SWA, Gadam, GadamX). For SGD, we use a base learning rate of 0.1 and use a grid searched initial learning rates in the range of $\{0.001, 0.01, 0.1\}$ and use the same learning rate for Padam, similar to the procedures suggested in Chen and Gu (2018). For Adam(W), we simply use the default initial learning rate of 0.001 except in VGG-16, where we use initial learning rate of 0.0005. After the best learning rate has been identified, we conduct a further search on the weight decay, which we find often leads to a trade-off between the convergence speed and final performance; again we search on the base optimisers only and use the same value for the IA optimisers. For CIFAR experiments, we search in the range of $[10^{-4}, 10^{-3}]$, from the suggestions of Loshchilov and Hutter (2019). For decoupled weight decay, we search the same range for the weight decay scaled by initial learning rate.

On ImageNet (Russakovsky et al., 2015) experiments, we conduct the following process. On WRN we use the settings recommended by Chrabaszcz et al. (2017), who conducted a thorough hyperparameter search: we set the learning rate at 0.03 and weight decay at 0.0001 for SGD/SWA and Padam, based on their searched optimal values. for AdamW/Gadam, we set decoupled weight decay at 0.01 and initial learning rate to be 0.001 (default Adam learning rate). For GadamX, we again use the same learning rate of 0.03, but since the weight decay in GadamX is partially decoupled, we set the decoupled weight decay to 0.0003. On PRN-110, we follow the recommendations of the authors of He et al. (2016b) to set the initial learning rate for SGD, Padam and GadamX to be 0.1. For AdamW and Gadam, we again use the default learning rate of 0.001. Following the observation by Loshchilov and

Hutter (2019) that smaller weight decay should be used for longer training (in PRN-110 we train for 200 epochs), we set weight decay at 10^{-5} and decoupled weight decay at 0.0003 (GadamX)/0.001 (others) respectively, where applicable.

Overall, we do **not** tune adaptive methods (Adam and Gadam) as much (most noticeably, we usually fix their learning rate to 0.001), and therefore in particular the AdamW results we obtain may or may not be at their optimal performance. Nonetheless, the rationale is that by design, one of the key advantage claimed is that adaptive optimisers should be less sensitive to hyperparameter choice, and in this paper, the key message is that Gadam performs well, *even though its base optimiser's parameters (AdamW) are rather crudely tuned.*

In all experiments, the momentum parameter ($\beta = 0.9$) for SGD and $\{\beta_1, \beta_2\} = \{0.9, 0.999\}$, $\epsilon = 10^{-8}$ for Adam and its variants, are left at their respective default values. For all experiments, unless otherwise stated, we average once per epoch. We also apply standard data augmentation (e.g. flip, random crops) and use a batch size of 128 for all experiments conducted.

Results. We show the results for the ResNext on CIFAR-100 in Table 1, ImageNet-32 in Table 2 and further CIFAR-100/10 results in Table 4. We also show the training curves for CIFAR-100 and ImageNet-32 in Figure 5. As AdamW always outperforms Adam in our experiments, the curves for the latter are omitted in the main text; we detail these results in the supplementary. The results show that optimisers with IA (SWA, Gadam and GadamX) invariably improve over their counterparts without, and GadamX always delivers the strongest performance. Without compromising convergence speed, Gadam outperforms tuned SGD and Padam - suggesting that solutions found by adaptive optimisers do not necessarily generalise more poorly, as suggested in the literature (Wilson et al., 2017). Indeed, any generalisation gap seems to be closed by the using IA and an appropriately implemented weight decay. We emphasise that results here are achieved **without** tuning the point at which we start averaging T_{avg}; if we allow crude tuning of T_{avg}, on CIFAR-100 GadamX achieves 77.22% (VGG-16) and 79.41% [4] (PRN-110) test accuracy respectively, which to our knowledge are the best reported performances on these architectures. We show results on ImageNet 32×32 (Chrabaszcz et al., 2017) in Figure 5c. While Gadam does not outperform our strong SGD baseline, it nevertheless improves upon AdamW greatly and posts a performance stronger than the SGD baseline in literature with identical (Chrabaszcz et al., 2017) and improved (McDonnell, 2018) setups. Finally, GadamX performs strongly, outperforming more than 3% compared to the baseline (Chrabaszcz et al., 2017) in Top-5 accuracy. We run each experiment three times with mean and standard deviation reported. In this section, all non-IA baselines are tuned rigorously with proper schedules for fair comparisons[5], and we also include the results reported in the previous works in Table 8 of Appendix Section C.

6.2 Image Classification on the Full ImageNet Dataset

We compare against step learning rate decay (factor of 10 every 30 epochs) and linear schedule for SGD and AdamW for 90 epochs (He et al., 2016a), with respective initial learning rates $\alpha = 0.1, 0.001$ and weight decays $10^{-4}, 10^{-2}$ on ImageNet (Russakovsky et al., 2015). We

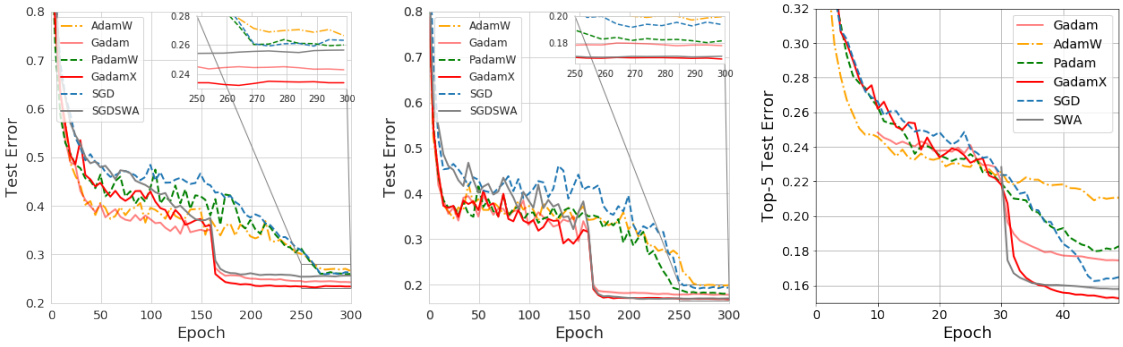
4. As opposed to 77.90% without tuning.
5. In image classification, we use the *linear schedule*, which both performs better than usual step schedule (see supplementary) and is consistent with (Izmailov et al., 2018).

Table 1: ResNeXt on CIFAR-100.

Architecture	Optimiser	Test Accuracy
ResNeXt-29	SGD	81.47±0.17
	SWA	82.95±0.28
	Adam(W)	80.16±0.16
	Padam(W)	82.37±0.35
	Gadam	82.13±0.20
	GadamX	**83.27**±0.11

Table 2: Test accuracy on ImageNet 32×32.

Architecture	Optimiser	Top-1	Top-5
WRN-28-10	SGD	61.33±0.11	83.52±0.14
	SWA	62.32±0.13	84.23±0.05
	AdamW	55.51±0.19	79.09±0.33
	Padam	59.65±0.17	81.74±0.16
	Gadam	60.50±0.19	82.56±0.13
	GadamX	**63.04**±0.06	**84.75**±0.03



(a) VGG-16 (b) ResNeXt-29 (c) WRN-28-10

Figure 5: (a-b) Top 1 Test error on CIFAR-100, (c) Top-5 Test Error on ImageNet-32 and (d) IA test improvement over its base optimiser against number of parameters.

Table 3: Accuracy on ImageNet.

Architecture	Optimiser	Top-1	Top-5
ResNet-50	SGD(step)	75.63	92.67
	SWA	76.32	93.15
	AdamW(lin)	74.04	91.57
	Ranger	75.64	92.53
	Gadam	76.79	93.21
	GadamX	**77.31**	**93.47**
ResNet-101	SGD (step)	77.37	93.78
	SWA	78.08	93.92
	AdamW(lin)	74.48	91.82
	Ranger	75.62	92.42
	Gadam	78.53	**94.29**
	GadamX	**78.72**	94.18

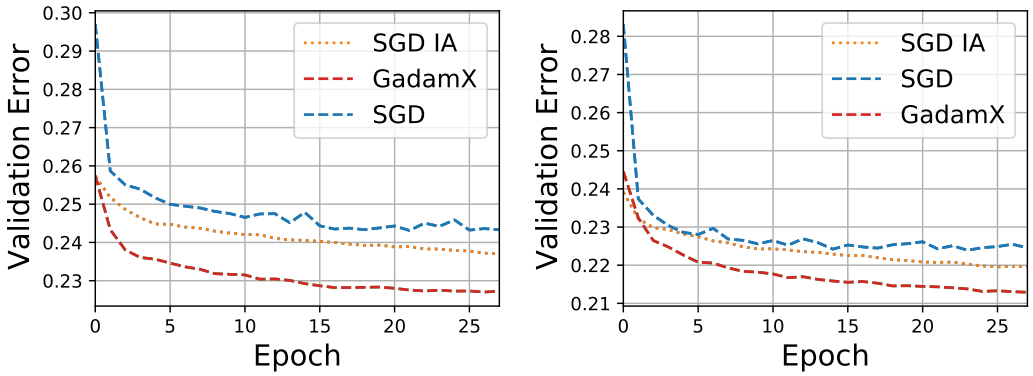
Table 4: Test accuracy on CIFAR-10/100.

Architecture	Optimiser	C10	C100
VGG-16	SGD	94.14±0.37	74.15±0.06
	SWA	94.69±0.36	74.57±0.27
	Adam(W)	93.90 ±0.11	73.26±0.30
	Padam(W)	94.13 ±0.06	74.56±0.19
	Gadam	94.62±0.15	75.73±0.29
	GadamX	**94.88**±0.03	**76.85**±0.08
PRN-110	SGD	95.40±0.25	77.22±0.05
	SWA	95.55±0.12	**77.92**±0.36
	Adam(W)	94.69±0.14	75.47±0.21
	Padam(W)	95.28±0.13	77.30 ±0.11
	Gadam	95.27±0.02	77.37±0.09
	GadamX	**95.95**±0.06	77.90 ±0.21

combine LookAhead (Zhang et al., 2019b) with gradient centralisation (Yong et al., 2020) as a high performance adaptive baseline *Ranger* (Wright, 2019), also using step decay. We search for the best performing initial learning rates for SGD, AdamW, SWA, GadamX and Ranger by factors of 3 i.e $0.001, 0.003$ in either direction (increase/decrease) until we find a

local maximum in performance, otherwise leaving settings as in Section 6.1. We show the results in Table 3.

Experimenting with partial adaptivity for the best image classification results. Following Granziol et al. (2020a); Choi et al. (2019), we experiment with setting the numerical stability coefficient to $\delta = 10^{-4}$ instead of 10^{-8} for GadamX and attempt an SGD like procedure for Gadam where we train with $\alpha = 0.5, \delta = 1, \gamma = 10^{-4}$. Note that such a large numerical stability coefficient has a similar effect to reducing the effect of the preconditioning matrix as GadamX hence also allowing for a larger global learning rate. We find that whilst the generalisation benefit of using Gadam alone is significant (without decreasing partial adaptivity) it is not competitive with SGD on this dataset, wheras leaving the numerical stability coefficient unchanged for GadamX only results in a very minor decrease in performance. We detail both of these effects in experimental finding 3.

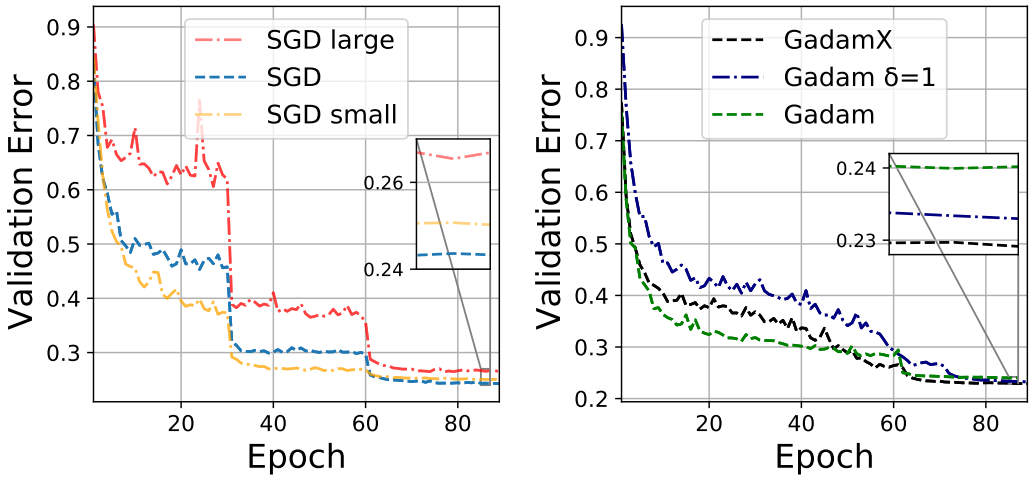


(a) ResNet-50 (b) ResNet-101

Figure 6: Final ImageNet epochs, showing the improvement of both SGD with Iterate Averaging (SGD IA) and our proposed GadamX optimiser over the SGD step-schedule in Top-1 validation error.

Due to poor "out of the box" performance of SWA, we repeat the logarithmic grid search procedure on the IA learning rate for SWA. We report results in Table 3, where we see Gadam(X) strongly out-performing all baselines. We do not include the ResNet-18 as AdamW outperforms SGD with 69.92% top-1 accuracy over 69.72%, hence not a useful test-case for analysing the *adaptive generalisation gap*, prevalent in deeper models. Gadam nonetheless improves on this attaining 70.11%. Whilst we find that step/linear scheduling is less effective for AdamW/SGD, attaining 73.68/75.52% respectively on the ResNet-50. Since these are small difference we don't consider scheduling to be a major factor in our outstanding results. We detail our major experimental findings from these experiments which could be of use to the community.

1. *Adaptive IA makes use of huge initial learning rates*: unlike SGD and SWA, which have a strong performance degradation when large initial learning rates are used, shown in Fig 7a, we find that large initial learning rates improve the generalisation performance of Gadam/GadamX, with the largest initial learning rates giving the best results.



(a) SGD $\alpha = [0.03, 0.1, 0.3]$ (b) Speed/Error trade-off

Figure 7: (a) Unlike IA adaptive methods, SGD does not benefit from larger initial learning rates. (b) To attain the greatest generalisation with adaptive methods, the fast convergence is sacrificed. Gadam $\delta = 1$, has a correspondingly large learning rate 0.5.

2. *Convergence speed comes at a cost:* Combining a large numerical stability coefficient and large learning rates allows Gadam to give significantly superior performance to SGD. However, the price paid is in convergence speed, shown in Fig 7b. For these settings the convergence speed is often as slow or slower than GadamX. Using the same settings as in the small scale experiments (shown as Gadam in the graph) we achieve a top-1 accuracy of 75.52 for ResNet-50. Whilst this significantly improves upon the base optimiser AdamW, these results are not as strong as those of SGD. Whilst increasing the base learning rate to 0.003 increases the ResNet-50 Gadam generalisation performance to 76.53, much of the convergence speed is already lost. We note that the effective weight decay is given by $(1 - \alpha\gamma)$ so we expect higher regularisation from higher learning rates. We do not find that increasing the weight decay whilst keeping the same base learning rates produces as strong results in our experiments and hence this learning rate and weight decay interplay could form the basis for interesting future work.

3. *Partially adaptive optimisation generalises best:* We find that for all experiments GadamX delivers the strongest performance. We do not find a strong dependence on the choice of the IA starting point (we try epoch $61, 71, 81$). We find that altering the numerical stability constant gives a small boost in Top-1 error, from 77.19 to 77.31 for the ResNet-50, but that results remain strong for the traditional setting.

Comparison to results reported in the literature. We specifically report the final (as opposed to best) validation error for all our runs. We find the best SGD ResNet-50/101 results to be 75.75/77.62%, which are slightly worse/better than the official repository results. All of these results are still significantly lower than results achieved by Gadam/GadamX. We

note that iterate averaged methods seem to continually decrease error in the final epochs of training, unlike SGD, which can sometimes overfit slightly in the final epochs of training.

6.3 Language Modelling with LSTM

We run word-level language modelling using a 3-layer Long-short Term Memory (LSTM) model (Gers et al., 2000) on the Penn Treebank (PTB) dataset (Marcus et al., 1993) and the results are shown in Table 5 and Figure 8. Remarkably, Gadam achieves a test perplexity of 58.77 (58.61 if we tune T_{avg}. See Table 6 in Section 7.2), better than the baseline NT-ASGD in Merity et al. (2017) that *runs an additional 300 epochs* on an identical network. Note that since, by default, the ASGD uses a constant learning rate, we do *not* schedule the learning rate except Padam which requires scheduling to converge. Also, for consistency, we use a manual trigger to start averaging at the 100th epoch for ASGD (which actually outperforms the NT-ASGD variant). We additionally conduct experiments *with* scheduling and NT-ASGD (Appendix Section D) and Gadam still outperforms. It is worth mentioning that for state of the art results in language modelling (Melis et al., 2017; Brown et al., 2020; Shoeybi et al., 2019), Adam is the typical optimiser of choice. Hence these results are both encouraging and significant for wider use in the community.

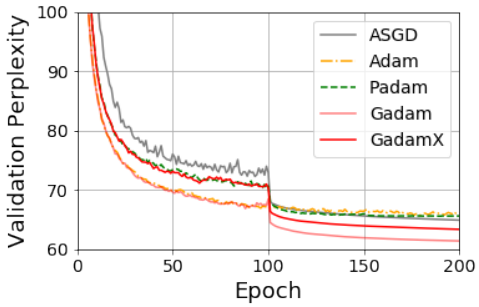


Figure 8: Validation perplexity of 3-layer LSTM on PTB word-level modelling

Data	Optimiser	Perplexity	
		Validation	Test
PTB	ASGD	64.88±0.07	61.98±0.19
	Adam	65.96±0.08	63.16±0.24
	Gadam	**61.35**±0.05	**58.77**±0.08
	GadamX	63.49±0.19	60.45±0.04

Table 5: Validation and test perplexities on the Penn Treebank (PTB) language modelling task using a LSTM.

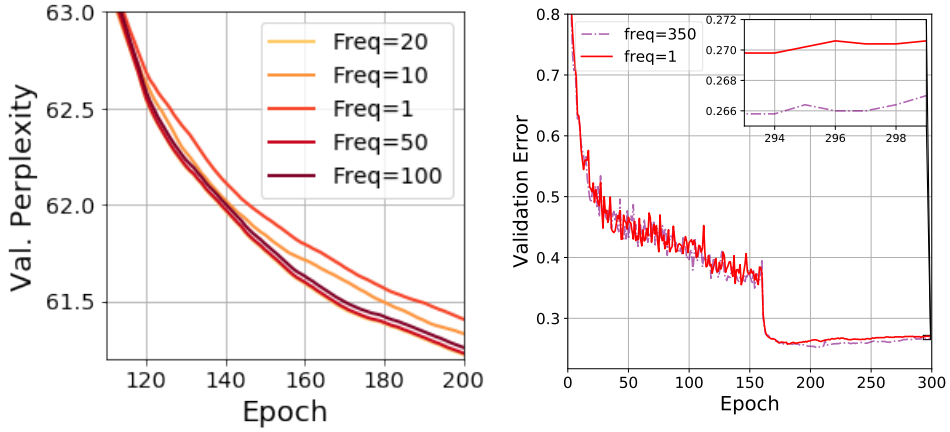
6.4 Transformers with Iterate Averaging

We note that since the initial draft of this paper, that the transformer architecture has become extremely prevalent both in natural language processing and computer vision. We note that "checkpoint" averaging, which is simply strided averaging either per epoch or per X epochs ion ubiqitous in this field with the original transformer paper (Vaswani et al., 2017) uses checkpoint averaging of the last 5 checkpoints for their base model and the last 20 checkpoints for their large model, which was found as an optimal amount on a development set. We note that this is in line with our theoretical predictions. Iterate averaging improves performance, strided averaging works similarly well and we expect increases in model size to bring greater improvements in averaging. We note that other works in neural machine translation (Gao et al., 2022; Shao and Feng, 2022) also investigate the impact of checkpoint averaging and find positive results, including for the finetuning case.

7 Ablation Studies and Additional Experiments

7.1 Effect of the Frequency of Averaging

While we derive the theoretical bounds for both Polyak-style averaging on every *iteration* and strided averaging, in practice we use strided averaging to save on computation. We either average once per *epoch* similar to Izmailov et al. (2018), or select a rather arbitrary value such as averaging once per 100 iterations. The reason is both practical and theoretical: averaging much less frequently leads to significant computational savings and for more independent iterates the benefit from averaging is greater. In this case, averaging less causes the iterates to be further apart and more independent, and thus fewer number of iterates is required to achieve the similar level of performance if less independent iterates are used. We verify this both on the language and the vision experiments using an identical setup. With reference to Figure 9(a), not only is the final perplexity very insensitive to averaging frequency (note that the y-axis scale is very small), it is also interesting that averaging *less* actually leads to a slightly better validation perplexity compared to schemes that, say, average every iteration. We see a similar picture emerge in Figure 9(b), where in spite of following very close trajectories, averaging every iteration gives a slightly worse testing performance compared to once an epoch and is also significantly more expensive (with a NVIDIA GeForce RTX 2080 Ti GPU, each epoch of training takes around 10s if we average once per epoch but averaging every iteration takes around 20s).



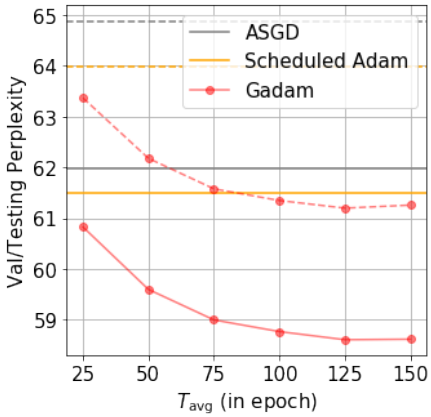
(a) LSTM on PTB (b) VGG-16 on CIFAR-100

Figure 9: Effect of different averaging frequencies on validation perplexity of Gadam on representative (a) Language and (b) Image classification tasks. `Freq=`n suggests averaging once per n iterations. `freq=350` in (b) is equivalently averaging once per *epoch*.

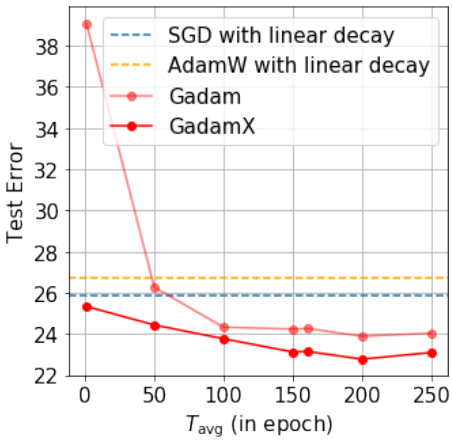
7.2 Effect of the Starting Point of Averaging and GadamAuto

In Gadam(X), we need to determine when to start averaging (T_{avg} in Algorithm 1), and here we investigate the sensitivity of Gadam(X) to this hyperparameter. We use a range of T_{avg} for a number of different tasks and architectures (Figure 10 and Table 6), including extreme choices such as $T_{\text{avg}} = 0$ (start averaging at the beginning). We observe that for any

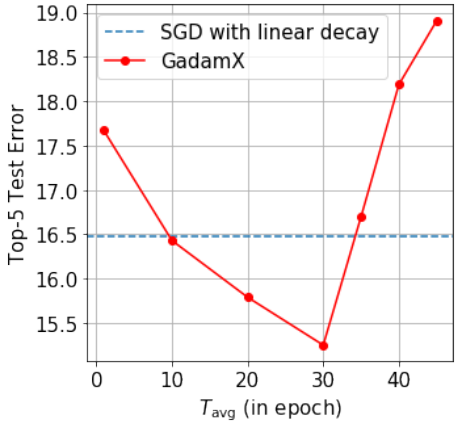
reasonable T_{avg}, Gadam(X) always outperform their base optimisers with standard learning rate decay, and tuning T_{avg} yields even more improvements over the heuristics employed in the main text, even if selecting any sensible T_{avg} already can lead to a promising performance over standard learning rate decay.



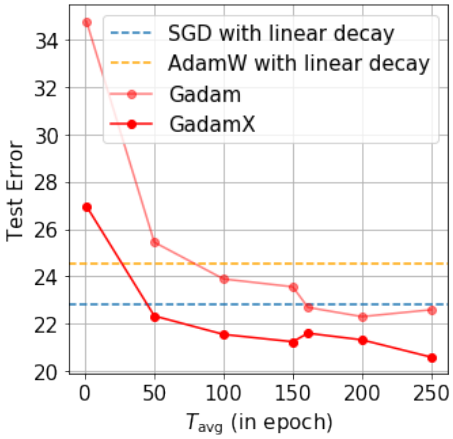
(a) LSTM. *dashed/solid* lines denote val/test perp.



(b) VGG-16 on CIFAR-100



(c) WRN-28-10 on ImageNet 32×32



(d) PRN-110 on CIFAR-100

Figure 10: Effect of different T_{avg} on the performance of various tasks and architectures.

Table 6: Best results obtained from tuning T_{avg}

Dataset	CIFAR-100		ImageNet 32×32	PTB
Architecture	VGG-16		WRN-28-10	LSTM
Optimiser	Gadam	GadamX	GadamX	Gadam
Test Acc./Perp.	76.11	77.22	84.75	58.61

Here we also conduct preliminary experiments on *GadamAuto* (Table 7), a variant of Gadam that uses a constant learning rate schedule and automatically determines the starting

point of averaging and training termination - this is possible given the insensitivity of the end-results towards T_{avg} as shown above, and is desirable as the optimiser both has fewer hyperparameters to tune and trains faster. We use VGG-16 network on CIFAR-100. For all experiments, we simply use a flat learning rate schedule. The results are shown in Table 7. We use a patience of 10 for both the determination of the averaging activation and early termination. We also include SWA experiments with SGD iterates.

Table 7: GadamAuto test performance at termination.

Optimiser	Data-set	Test Accuracy
Gadam-Auto	CIFAR-100	75.39
SWA-Auto	CIFAR-100	73.93

It can be seen that, while automatic determination for averaging trigger and early termination work well for Gadam (GadamAuto posts a performance only marginally worse than the manually tuned Gadam), they lead to a rather significant deterioration in test in SWA (SWA-Auto performs worse than tuned SWA, and even worse than tuned SGD). This highlights the benefit of using adaptive optimiser as the base optimiser in IA, as the poor performance in SWA-Auto is likely attributed to the fact that SGD is much more hyperparameter-sensitive (to initial learning rate and learning rate schedule, for example. SWA-Auto uses a constant schedule, which is sub-optimal for SGD), and that validation performance often fluctuates more during training for SGD: SWA-Auto determines averaging point based on the number of epochs of validation accuracy stagnation. For a noisy training curve, averaging might be triggered too early; while this can be ameliorated by setting a higher patience, doing so will eventually defeat the purpose of using an automatic trigger. Both issues highlighted here are less serious in adaptive optimisation, which likely leads to the better performance of GadamAuto.

Nonetheless, the fact that scheduled Gadam still outperforms GadamAuto suggests that there is still ample room of improvement to develop a truly automatic optimiser that performs as strong as or even stronger than tuned ones. One desirable alternative we propose for the future work is the integration of *Rectified Adam* (Liu et al., 2019), which is shown to be much more insensitive to choice of hyperparameter even compared to Adam.

8 Conclusion

We propose a Gaussian process perturbation between the batch and true risk surfaces and derive the phenomenon of improved generalisation for large learning rates and larger weight decay when combined with iterate averaging observed in practice. We extend this formalism to include adaptive methods and show that we expect further improvement when using adaptive algorithms. Based on this theory we develop two adaptive algorithms, Gadam and GadamX, variants of Adam with iterate averaging. We extensively validate Gadam and GadamX on computer vision tasks and a natural language experiment, showing strong performance against baseline and state of the art. Another interesting consequence of our work is that in all our experiments *the last iterate is the best.* Unlike SGD, where the epoch of best test/validation error is typically not the last and techniques such as early stopping are often employed, we find consistent near-monotonic improvements in test/validation error

using our algorithms. We also find from preliminary analysis that our algorithms require less hyper-parameter tuning than SGD and variants thereof. This may be of interest for practitioners that want to get good results fast, as opposed to state of the art slowly.

Acknowledgements

DG would like to thank the Oxford-Man Institute for financial support, the JADE computing facility and its Oxford-based administrator, A. Gittings, in particular. The authors thank D. Campbell and S. Koepke for feedback. SA was supported by EP/T028572/1. XW is supported by the Clarendon Scholarship.

References

Milton Abramowitz, Irene A Stegun, and Robert H Romer. Handbook of mathematical functions with formulas, graphs, and mathematical tables, 1988.

Robert J Adler and Jonathan E Taylor. *Random fields and geometry*. Springer Science & Business Media, 2009.

Greg W Anderson, Alice Guionnet, and Ofer Zeitouni. *An introduction to random matrices*. Number 118. Cambridge university press, 2010.

George E. Andrews, Richard Askey, and Ranjan Roy. *Special Functions*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1999. doi: 10.1017/CBO9781107325937.

Francis Bach and Eric Moulines. Non-strongly-convex smooth stochastic approximation with convergence rate $\mathcal{O}(1/n)$. *Advances in neural information processing systems*, 26, 2013.

Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can we gain more from orthogonality regularizations in training deep networks? In *Advances in Neural Information Processing Systems*, pages 4261–4271, 2018.

Nicholas P Baskerville, Jonathan P Keating, Francesco Mezzadri, and Joseph Najnudel. The loss surfaces of neural networks with general activation functions. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(6):064001, 2021a.

Nicholas P Baskerville, Jonathan P Keating, Francesco Mezzadri, and Joseph Najnudel. A spin-glass model for the loss surfaces of generative adversarial networks. *arXiv preprint arXiv:2101.02524*, 2021b.

Irwan Bello, William Fedus, Xianzhi Du, Ekin D Cubuk, Aravind Srinivas, Tsung-Yi Lin, Jonathon Shlens, and Barret Zoph. Revisiting resnets: Improved training and scaling strategies. *arXiv preprint arXiv:2103.07579*, 2021.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Jinghui Chen and Quanquan Gu. Closing the generalization gap of adaptive gradient methods in training deep neural networks. *arXiv preprint arXiv:1806.06763*, 2018.

Soumith Chintala et al. Pytorch Imagenet baseline, 2017. URL `https://github.com/pytorch/examples/blob/master/imagenet/main.py`. "2016 (accessed September, 2020)".

Dami Choi, Christopher J Shallue, Zachary Nado, Jaehoon Lee, Chris J Maddison, and George E Dahl. On empirical comparisons of optimizers for deep learning. *arXiv preprint arXiv:1910.05446*, 2019.

Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015.

Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of ImageNet as an alternative to the CIFAR datasets. *arXiv preprint arXiv:1707.08819*, 2017.

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. RandAugment: Practical data augmentation with no separate search. *arXiv preprint arXiv:1909.13719*, 2019.

Aaron Defazio and Léon Bottou. On the ineffectiveness of variance reduced optimization for deep learning. In *Advances in Neural Information Processing Systems*, pages 1753–1763, 2019.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages "1019–1028". "JMLR. org", 2017.

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.

John C Duchi. Introductory lectures on stochastic optimization. *The Mathematics of Data*, 25:99, 2018.

Yingbo Gao, Christian Herold, Zijian Yang, and Hermann Ney. Revisiting checkpoint averaging for neural machine translation. *arXiv preprint arXiv:2210.11803*, 2022.

Elizabeth Gardner and Bernard Derrida. Optimal storage properties of neural network models. *Journal of Physics A: Mathematical and general*, 21(1):271, 1988.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *Advances in Neural Information Processing Systems*, pages 8789–8798, 2018.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

Diego Granziol. Curvature is Key: Sub-Sampled Loss Surfaces and the Implications for Large Batch Training. *arXiv preprint arXiv:2006.09092*, 2020a.

Diego Granziol. Flatness is a false friend. *arXiv preprint arXiv:2006.09091*, 2020b.

Diego Granziol, Xingchen Wan, and Timur Garipov. MLRG deep curvature. *arXiv preprint arXiv:1912.09656*, 2019.

Diego Granziol, Samuel Albanie, Xingen Wan, and Stephen Roberts. Explaining the adaptive generalisation gap. *arXiv preprint arXiv*, 2020a.

Diego Granziol, Timur Garipov, Dmitry Vetrov, Stefan Zohren, Stephen Roberts, and Andrew Gordon Wilson. Towards understanding the true loss surface of deep neural networks using random matrix theory and iterative spectral methods, 2020b. URL `https://openreview.net/forum?id=H1gza2NtwH`.

Nicholas JA Harvey, Christopher Liaw, Yaniv Plan, and Sikander Randhawa. Tight analyses for non-smooth stochastic gradient descent. In *Conference on Learning Theory*, pages 1579–1613. PMLR, 2019.

Haowei He, Gao Huang, and Yang Yuan. Asymmetric valleys: Beyond sharp and flat local minima. *arXiv preprint arXiv:1902.00744*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016a.

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, 2016b.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. In *Advances in Neural Information Processing Systems*, pages 2160–2170, 2018.

Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 304–320, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

Prateek Jain, Dheeraj Nagaraj, and Praneeth Netrapalli. Making the last iterate of SGD information theoretically optimal. In *Conference on Learning Theory*, pages 1752–1755. PMLR, 2019.

Stanislaw Jastrzebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in SGD. *arXiv preprint arXiv:1711.04623*, 2017.

Stanislaw Jastrzebski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho, and Krzysztof Geras. The break-even point on the optimization trajectories of deep neural networks. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=r1g87C4KwB`.

Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from Adam to SGD. *arXiv preprint arXiv:1712.07628*, 2017.

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

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992.

Harold Kushner and G George Yin. *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer Science & Business Media, 2003.

Simon Lacoste-Julien, Mark Schmidt, and Francis Bach. A simpler approach to obtaining an $\mathcal{O}(1/t)$ convergence rate for the projected stochastic subgradient method. *arXiv preprint arXiv:1212.2002*, 2012.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pages 6389–6399, 2018.

Yuanzhi Li, Colin Wei, and Tengyu Ma. Towards explaining the regularization effect of initial large learning rate in training neural networks. In *Advances in Neural Information Processing Systems*, pages 11674–11685, 2019.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for Bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, pages 13132–13143, 2019.

Stefano Sarao Mannelli, Florent Krzakala, Pierfrancesco Urbani, and Lenka Zdeborova. Passed & spurious: Descent algorithms and local minima in spiked matrix-tensor models. *arXiv preprint arXiv:1902.00139*, 2019.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. 1993.

James Martens. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

Mark D McDonnell. Training wide residual networks for deployment using a single bit for each weight. *arXiv preprint arXiv:1802.08530*, 2018.

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.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182*, 2017.

Marc Mezard, Giorgio Parisi, and Miguel Virasoro. *Spin glass theory and beyond: An Introduction to the Replica Method and its Applications*, volume 9. World Scientific Publishing Company, 1987.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Gergely Neu and Lorenzo Rosasco. Iterate averaging as regularization for stochastic gradient descent. In *Conference On Learning Theory*, pages 3222–3242. PMLR, 2018.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

Alexander Rakhlin, Ohad Shamir, and Karthik Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. *arXiv preprint arXiv:1109.5647*, 2011.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

Valentina Ros, Gerard Ben Arous, Giulio Biroli, and Chiara Cammarota. Complex energy landscapes in spiked-tensor and simple glassy models: Ruggedness, arrangements of local minima, and phase transitions. *Physical Review X*, 9(1):011003, 2019.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International conference on machine learning*, pages 71–79. PMLR, 2013.

Chenze Shao and Yang Feng. Overcoming catastrophic forgetting beyond continual learning: Balanced training for neural machine translation. *arXiv preprint arXiv:2203.03910*, 2022.

Satish Shirali and Harkrishan L Vasudeva. *An Introduction to Mathematical Analysis*. Alpha Science International, Limited, 2014.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

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

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Phuong Thi Tran et al. On the convergence proof of AMSGrad and a new version. *IEEE Access*, 7:61706–61716, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

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 4148–4158, 2017.

Less Wright. Ranger - a synergistic optimizer. `https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer`, 2019.

Lei Wu, Chao Ma, and E Weinan. How SGD selects the global minima in over-parameterized learning: A dynamical stability perspective. In *Advances in Neural Information Processing Systems*, pages 8279–8288, 2018a.

Yuhuai Wu, Mengye Ren, Renjie Liao, and Roger Grosse. Understanding short-horizon bias in stochastic meta-optimization. *arXiv preprint arXiv:1803.02021*, 2018b.

Qizhe Xie, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. Self-training with noisy student improves ImageNet classification, 2019.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

Hongwei Yong, Jianqiang Huang, Xiansheng Hua, and Lei Zhang. Gradient centralization: A new optimization technique for deep neural networks. *arXiv preprint arXiv:2004.01461*, 2020.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. *arXiv preprint arXiv:1905.04899*, 2019.

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

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger Grosse. Three mechanisms of weight decay regularization. *arXiv preprint arXiv:1810.12281*, 2018.

Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George Dahl, Chris Shallue, and Roger B Grosse. Which algorithmic choices matter at which batch sizes? Insights from a noisy quadratic model. In *Advances in Neural Information Processing Systems*, pages 8194–8205, 2019a.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. Mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps forward, 1 step back. In *Advances in Neural Information Processing Systems*, pages 9593–9604, 2019b.

Juntang Zhuang, Tommy Tang, Sekhar Tatikonda, Nicha Dvornek, Yifan Ding, Xenophon Papademetris, and James S Duncan. AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients. *arXiv preprint arXiv:2010.07468*, 2020.

Appendix A. Proofs

In this section we give any proofs that were omitted from the main text.

A.1 Proof of Theorem 5

The proof of Theorem 5 was given in the main text but depends on several intermediate results which we now state and prove.

Lemma 8 *Take any $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n-1} \in \mathbb{R}^P$ let $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, for any $\boldsymbol{\mu} \in \mathbb{R}^P$ and Σ such that $\det \Sigma \geq A\sigma^{2P}$ for some constants $A, \sigma > 0$. Consider $P \to \infty$ with $P \gg \log n$ and let $\delta > 0$ be $o(P^{\frac{1}{2}})$ (note that δ and n need not diverge with P, but they can). Define*

$$B_i = \{\boldsymbol{x} \in \mathbb{R}^P \mid ||\boldsymbol{x} - \boldsymbol{x}_i|| < \delta\},$$

then as $P \to \infty$

$$\mathbb{P}\left(\boldsymbol{X} \in \bigcup_i B_i\right) \to 0 \tag{59}$$

and moreover as $P, n \to \infty$

$$n^l \mathbb{P}\left(\boldsymbol{X} \in \bigcup_i B_i\right) \to 0, \tag{60}$$

for any fixed $l > 0$.

Proof With the Euclidean volume measure, we have

$$Vol\left(\bigcup_i B_i\right) \leq nV_P\delta^P = V_P(\delta n^{1/P})^P$$

where V_P is the volume of the unit sphere in P dimensions. Therefore a sphere of radius $\delta n^{1/P}$ is large enough to enclose all of the B_i and so the probability that $\boldsymbol{X}$ lies in any of the B_i is bounded above by the probability that it lies inside the sphere of radius $\delta n^{1/P}$ centred on its mean $\boldsymbol{\mu}$. Note that with $\hat{\sigma}^2 = (\det \Sigma)^{1/P}$, changing variables $\boldsymbol{x} = \hat{\sigma}^{-1}\Sigma^{1/2}\boldsymbol{y}$ gives

$$\int_{\mathbb{R}^P} d\boldsymbol{x} e^{-\frac{\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{x}}{2}} = \int_{\mathbb{R}^P} d\boldsymbol{y} e^{-\frac{\boldsymbol{y}^2}{2\hat{\sigma}^2}}$$

since the Jacobian is 1. Thus we can reduce to a single dimensional Gaussian integral

$$\begin{aligned}
\mathbb{P}\left(\boldsymbol{X} \in \bigcup_i B_i\right) &\leq \frac{1}{(2\pi\hat{\sigma}^2)^{\frac{P}{2}}} \frac{2\pi^{\frac{P}{2}}}{\Gamma(\frac{P}{2})} \int_0^{\delta n^{\frac{1}{P}}} dr\ e^{-\frac{r^2}{2\hat{\sigma}^2}} r^{P-1} \\
&= \frac{2}{\Gamma(\frac{P}{2})} \int_0^{\frac{\delta n^{\frac{1}{P}}}{\sqrt{2}\hat{\sigma}}} dr\ e^{-r^2} r^{P-1} \\
&= \frac{1}{\Gamma(\frac{P}{2})} \int_0^{\frac{\delta n^{\frac{2}{P}}}{2\hat{\sigma}^2}} dr\ e^{-r} r^{\frac{P}{2}-1} \\
&\leq \frac{1}{\Gamma(\frac{P}{2})} \int_0^{\frac{\delta n^{\frac{2}{P}}}{2A^{1/P}\sigma^2}} dr\ e^{-r} r^{\frac{P}{2}-1} \qquad \text{(using } \hat{\sigma}^2 \geq A^{1/P}\sigma^2\text{)} \\
&\leq \frac{1}{\Gamma(\frac{P}{2})} \int_0^{\frac{\delta n^{\frac{2}{P}}}{2\alpha\sigma^2}} dr\ e^{-r} r^{\frac{P}{2}-1} \qquad \text{(with } \alpha \equiv \inf_P A^{1/P} > 0\text{)} \\
&\equiv \frac{1}{\Gamma(\frac{P}{2})} \gamma\left(\frac{P}{2}; \frac{n^{\frac{2}{P}}\delta^2}{2\sigma^2\alpha}\right) \qquad (61)
\end{aligned}$$

where γ is the lower incomplete gamma function. Since $P \gg \log n$ and $\delta = o(P^{\frac{1}{2}})$, it follows that

$$x \equiv \frac{n^{\frac{2}{P}}\delta^2}{2\sigma^2\alpha} = o(P)$$

and so Lemma 9 can be applied to yield the result. Indeed, recalling that $n \ll e^P$, we have

$$n^l \mathbb{P}\left(\boldsymbol{X} \in \bigcup_i B_i\right) \leq e^{lP} r(P/2, x) \sim \frac{1}{\sqrt{2\pi}} \exp\left(lP - x + \frac{P}{2}\log x - \frac{P}{2} - \frac{P}{2}\log\frac{P}{2} - \frac{1}{2}\log\frac{P}{2}\right)$$

for any $l > 0$. But $x = o(P)$ so for P large enough, the term inside the exponential is negative and diverging with P, as required. ■

Lemma 9 *Define the function*

$$r(a; x) = \frac{\gamma(a; x)}{\Gamma(a)}, \qquad (62)$$

where γ is the lower incomplete gamma function. Assume that $x \ll a$, where x may or may not diverge with a, then as $a \to \infty$, $r(a; x) \to 0$, and more precisely

$$r(a; x) \sim \frac{1}{\sqrt{2\pi}} \exp\left(-x + a\log x - a - a\log a - \frac{1}{2}\log a\right). \qquad (63)$$

Proof We have $\gamma(a;x) = a^{-1}x^a\,{}_1F_1(a;1+a;-x)$, where ${}_1F_1$ is the confluent hypergeometric function of the first kind (Andrews et al., 1999). Then

$$r(a;x) = \frac{a^{-1}x^a\,{}_1F_1(a;1+a;-x)}{\Gamma(a)} = \frac{a^{-1}x^a\Gamma(a+1)}{\Gamma(a)^2}\int_0^1 e^{xt}t^{a-1}dt \tag{64}$$

where we have used a result of Abramowitz et al. (1988). The integral in (64) can be evaluated asymptotically in the limit $x \to \infty$ with $x \ll a$. Writing the integrand as $e^{xt+(a-1)\log t}$ it is plainly seen to have no saddle points in $[0,1]$ given the condition $x \ll a$. The leading order term therefore originates at the right edge $t = 1$. A simple application of Laplace's method leads to

$$\begin{aligned} r(a;x) &\sim \frac{a^{-1}x^a\Gamma(a+1)e^{-x}}{\Gamma(a)^2(a-1-x)} \\ &\sim \frac{x^a e^{-x}}{a\Gamma(a)} \\ &\sim \frac{x^a e^{-x}}{a\sqrt{2\pi a^{-1}}(ae^{-1})^a} \\ &= \frac{1}{\sqrt{2\pi}}\exp\left(-x + a\log x - a - a\log a - \frac{1}{2}\log a\right) \end{aligned}$$

where the penultimate line makes uses of Stirling's approximation (Andrews et al., 1999). Since $a \gg x$,

$$-x + a\log x - a - a\log a - \frac{1}{2}\log a \sim -a\log a \to -\infty$$

which completes the proof. ■

The following two lemmas were stated in the main text. Their proofs, which we now give, depend on the preceding lemmas.

Lemma 3 *Let $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ be a sequence of jointly multivariate Gaussian random variables in $\mathbb{R}^P$ such that*

$$\boldsymbol{X}_i \;\mid\; \{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{i-1}\} \sim \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$$

where there exists a $\sigma > 0$ and a constant $A > 0$ such that $\det\Sigma_i \geq A\sigma^P$ for all P and i. Let also $\boldsymbol{X}_0$ be any deterministic element of $\mathbb{R}^P$. For $1 \leq m \leq n$, define the events

$$A_m(\delta) = \{||\boldsymbol{X}_i - \boldsymbol{X}_j||_2 > \delta \mid 0 \leq i < j \leq m\}.$$

Consider $P \to \infty$ with $P \gg \log n$ and let $\delta > 0$ be $o(P^{\frac{1}{2}})$ (note that δ and n need not diverge with P, but they can). Then $\mathbb{P}(A_n(\delta)) \to 1$ as $P \to \infty$.

Proof of Lemma 3 Let us use the definitions of B_i from Lemma 8, i.e. let

$$B_i = \{\boldsymbol{x} \in \mathbb{R}^P \mid ||\boldsymbol{x} - \boldsymbol{X}_i|| < \delta\}$$

for $0 < j < n$. Since $A_i(\delta) \subset A_{i-1}(\delta)$ for any i, the chain rule of probability gives

$$\mathbb{P}(A_n(\delta)) = \mathbb{P}\left(\bigcap_{i\leq n} A_i(\delta)\right) = \mathbb{P}(A_1(\delta)) \prod_{i=2}^{n-1} \mathbb{P}(A_i \mid A_{i-1})$$

but

$$\mathbb{P}(A_i(\delta) \mid A_{i-1}(\delta)) = 1 - \mathbb{P}\left(\boldsymbol{X}_i \in \bigcup_{j<i} B_j\right)$$

and so

$$\mathbb{P}(A_n(\delta)) = \mathbb{P}(A_1(\delta)) \prod_{i=2}^{n-1} \left(1 - \mathbb{P}\left(\boldsymbol{X}_i \in \bigcup_{j<i} B_j\right)\right) \tag{65}$$

$$= \mathbb{P}\left(\boldsymbol{X}_1 \in B_0\right) \prod_{i=2}^{n-1} \left(1 - \mathbb{P}\left(\boldsymbol{X}_i \in \bigcup_{j<i} B_j\right)\right). \tag{66}$$

For fixed n, the result is now immediate from (60) in Lemma 8, since all the probabilities in (65) converge to 1 as $P \to \infty$ and there are only a finite number of terms. Now consider the case that n also diverges. For any n define

$$s_n = \sup_{2\leq i\leq n} \mathbb{P}\left(\boldsymbol{X}_i \in \bigcup_{j<i} B_j\right),$$

and then

$$\mathbb{P}(A_n(\delta)) \geq \mathbb{P}\left(\boldsymbol{X}_1 \in B_0\right) \prod_{i=2}^{n-1} \left(1 - s_{i-2}\right).$$

But, by Lemma 8 we can write $s_n = (n+1)^{-2} f_{n,P}$ where $f_{n,P} \to 0$ as $P \to \infty$, say, hence

$$\mathbb{P}(A_n(\delta)) \geq \mathbb{P}\left(\boldsymbol{X}_1 \in B_0\right) \prod_{i=2}^{n-1} \left(1 - (i-1)^{-2} f_{i-2,P}\right) \geq \mathbb{P}\left(\boldsymbol{X}_1 \in B_0\right) \prod_{i=2}^{\infty} \left(1 - (i-1)^{-2} f_{i-2,P}\right)$$

for large n, since $|f_{n-2,P}| < 1$ and all the extra terms added are strictly between 0 and 1. But

$$\log \prod_{i=2}^{\infty} \left(1 - (i-1)^{-2} f_{i-2,P}\right) \geq -\sum_{i=2}^{\infty} (i-1)^{-2} f_{i-2,P} \geq -\sup_j f_{j-2,P} \sum_{i=2}^{\infty} (i-1)^{-2} = -\frac{\pi^2}{6} \sup_j f_{j-2,P}$$

and so

$$\mathbb{P}(A_n(\delta)) \geq e^{-\sup_j f_{j-2,P} \pi^2/6} \mathbb{P}\left(\boldsymbol{X}_1 \in B_0\right)$$

but $f_{j-2,P} \to 0$ for any j, so as $P \to \infty$, $\mathbb{P}(A_n(\delta))$ is lower bounded by a term converging to $\mathbb{P}(\boldsymbol{X}_1 \in B_0)$ which, in turn, converges to 1 by Lemma (8).

(60) gives the result. ■

Recall the Gaussian process covariance structure from the main text (21):

$$\mathrm{Cov}(\epsilon_i(\boldsymbol{w}), \epsilon_j(\boldsymbol{w}')) = (w_i - w_i')(w_j' - w_j)k''\left(-\frac{1}{2}||\boldsymbol{w} - \boldsymbol{w}'||_2^2\right) + \delta_{ij}k'\left(-\frac{1}{2}||\boldsymbol{w} - \boldsymbol{w}'||_2^2\right) \tag{21}$$

Lemma 4 *Assume the covariance structure (21). Take any $a_i \in \mathbb{R}$ and define $\bar{\boldsymbol{\epsilon}} = \sum_{i=1}^n a_i \boldsymbol{\epsilon}_i$, where $\boldsymbol{\epsilon}_i = \boldsymbol{\epsilon}(\boldsymbol{w}_i)$. Then*

$$\mathrm{Tr}\ \mathrm{Cov}(\bar{\boldsymbol{\epsilon}}) = k'(0)P\sum_{i=1}^n a_i^2 + 2P\sum_{1\le i<j\le n} a_i a_j \left[k'\Big(-\frac{d_{ij}^2}{2}\Big) + P^{-1}k''\Big(-\frac{d_{ij}^2}{2}\Big)d_{ij}^2\right] \tag{22}$$

where we define $d_{ij} = ||\boldsymbol{w}_i - \boldsymbol{w}_j||_2$.

Proof of Lemma 4 Each of the $\boldsymbol{\epsilon}_i$ is Gaussian distributed with covariance matrix $\mathrm{Cov}(\boldsymbol{\epsilon}_i)$ given by (21) and the covariance between different gradients $\mathrm{Cov}(\boldsymbol{\epsilon}_i, \boldsymbol{\epsilon}_j)$ is similarly given by (21). By standard multivariate Gaussian properties

$$\mathrm{Cov}(\bar{\boldsymbol{\epsilon}}) = \sum_{i=1}^n a_i^2\ \mathrm{Cov}(\boldsymbol{\epsilon}_i) + \sum_{i\neq j} a_i a_j \mathrm{Cov}(\boldsymbol{\epsilon}_i, \boldsymbol{\epsilon}_j), \tag{67}$$

then taking the trace

$$\mathrm{Tr}\ \mathrm{Cov}(\bar{\boldsymbol{\epsilon}}) = \sum_{i=1}^n a_i^2\, \mathrm{Tr}(\mathrm{Cov}(\boldsymbol{\epsilon}_i)) + 2\sum_{1\le i<j\le n} a_i a_j\, \mathrm{Tr}(\mathrm{Cov}(\boldsymbol{\epsilon}_i, \boldsymbol{\epsilon}_j)). \tag{68}$$

Using the covariance structure from (21) gives

$$\mathrm{Tr}\ \mathrm{Cov}(\bar{\boldsymbol{\epsilon}}) = k'(0)\sum_{i=1}^n a_i^2\, \mathrm{Tr}\, I + 2\sum_{1\le i<j\le n} a_i a_j \Big[k'(-\frac{d_{ij}^2}{2})\, \mathrm{Tr}\, I + k''(-\frac{d_{ij}^2}{2})\, \mathrm{Tr}(\boldsymbol{w}_i - \boldsymbol{w}_j)(\boldsymbol{w}_j - \boldsymbol{w}_i)^T\Big] \tag{69}$$

from which the result follows. ■

A.2 Proof Corollary 6

The proof given here is quite similar to the proof of Theorem 5 in the main text and so we here present only the differences.

Proof of Corollary 6 The proof is just as in Theorem 2 (or Theorems 3 or 4), differing only in the values of the $\bar{a}_i$. Indeed, a little thought reveals that the generalisation of $\bar{a}_i$ to the case $\kappa > 1$ is

$$\bar{a}_i = \frac{\alpha\kappa}{n}(1-\alpha\lambda)^{\kappa\left(1+\lfloor\frac{i}{\kappa}\rfloor\right)-1-i}\frac{1-(1-\alpha\lambda)^{\kappa\left(\lfloor\frac{n}{\kappa}\rfloor-\lfloor\frac{i}{\kappa}\rfloor\right)}}{1-(1-\alpha\lambda)^{\kappa}}. \tag{70}$$

Note that $\kappa\left\lfloor\frac{i}{\kappa}\right\rfloor - i$ is just the (negative) remainder after division of i by κ. Then for large n

$$\begin{aligned}
\sum_i \bar{a}_i^2 &\sim \frac{\alpha^2\kappa^2}{n^2}\frac{(1-\alpha\lambda)^{2(\kappa-1)}}{\left(1-(1-\alpha\lambda)^{\kappa}\right)^2}\left\lfloor\frac{n}{\kappa}\right\rfloor\sum_{i=0}^{\kappa-1}(1-\alpha\lambda)^{-2i} \\
&\leq \frac{\alpha^2\kappa}{n}\frac{(1-\alpha\lambda)^{2(\kappa-1)}}{\left(1-(1-\alpha\lambda)^{\kappa}\right)^2}\sum_{i=0}^{\kappa-1}(1-\alpha\lambda)^{-2i} \\
&= \frac{\alpha^2\kappa}{n}\frac{(1-\alpha\lambda)^{2(\kappa-1)}}{\left(1-(1-\alpha\lambda)^{\kappa}\right)^2}\frac{1-(1-\alpha\lambda)^{-2\kappa}}{1-(1-\alpha\lambda)^{-2}} \\
&= \frac{\alpha^2\kappa}{n}\frac{1}{\left(1-(1-\alpha\lambda)^{\kappa}\right)^2}\frac{1-(1-\alpha\lambda)^{2\kappa}}{1-(1-\alpha\lambda)^{2}}.
\end{aligned}$$

and similarly

$$\sum_{i<j}\bar{a}_i\bar{a}_j \sim \frac{\alpha^2\kappa^2}{n^2}\frac{(1-\alpha\lambda)^{2(\kappa-1)}}{(1-(1-\alpha\lambda)^{\kappa})^2}\sum_{i<j}(1-\alpha\lambda)^{\kappa\lfloor i/\kappa\rfloor-i+\kappa\lfloor j/\kappa\rfloor-j} \tag{71}$$

$$\sim \frac{\alpha^2\kappa^2}{n^2}\frac{(1-\alpha\lambda)^{2(\kappa-1)}}{(1-(1-\alpha\lambda)^{\kappa})^2}\sum_{j}(1-\alpha\lambda)^{\kappa\lfloor j/\kappa\rfloor-j}\left\lfloor\frac{j}{\kappa}\right\rfloor\frac{1-(1-\alpha\lambda)^{-\kappa}}{1-(1-\alpha\lambda)^{-1}} \tag{72}$$

$$\sim \frac{\alpha^2\kappa^2}{n^2}\frac{(1-\alpha\lambda)^{2(\kappa-1)}}{(1-(1-\alpha\lambda)^{\kappa})^2}\left(\frac{1-(1-\alpha\lambda)^{-\kappa}}{1-(1-\alpha\lambda)^{-1}}\right)^2\sum_{j=0}^{\lfloor n/\kappa\rfloor} j \tag{73}$$

$$\sim \frac{\alpha^2}{2}\frac{(1-\alpha\lambda)^{2(\kappa-1)}}{(1-(1-\alpha\lambda)^{\kappa})^2}\left(\frac{1-(1-\alpha\lambda)^{-\kappa}}{1-(1-\alpha\lambda)^{-1}}\right)^2 \tag{74}$$

$$= \frac{\alpha^2}{2}\frac{(1-\alpha\lambda)^{-2}}{(1-(1-\alpha\lambda)^{-1})^2}. \tag{75}$$

■

Appendix B. Gadam and Lookahead

Previous works also use EMA in weight space to achieve optimisation and/or generalisation improvements: Izmailov et al. (2018) entertain EMA in SWA, although they conclude

simple averaging is more competitive. Recently, Zhang et al. (2019b) proposes *Lookahead* (LH), a plug-in optimiser that uses EMA on the slow weights to improve convergence and generalisation. Nonetheless, having argued the dominance of noise in the high-dimensional deep learning regime, we argue that simple averaging is more theoretically desirable *for generalisation*. Following the identical analysis to the noisy quadratic with i.i.d noise, we consider the $1D$ case without loss of generality and denote $\rho \in [0, 1]$ as the coefficient of decay, asymptotically the EMA point $\boldsymbol{w}_{\text{ema}}$ is governed by:

$$\mathcal{N}\Big(\frac{(1-\rho)w_0(1-\alpha\lambda)^{n+1}[1-(\frac{\rho}{1-\alpha\lambda})^{n-1}]}{1-\alpha\lambda-\rho}, \frac{1-\rho}{1+\rho}\frac{\alpha\sigma^2\kappa}{\lambda}\Big) \tag{76}$$

Where $\kappa = (1-(1-\alpha\lambda)^{n-2}))$. An alternative analysis of EMA arriving at similar result was done in Zhang et al. (2019a), but their emphasis of comparison is between the EMA and *iterates* instead of EMA and the *IA point* in our case. From (76), while the convergence in mean is less strongly affected, the noise is reduced by a factor of $\frac{1-\rho}{1+\rho}$. So whilst we reduce the noise possibly by a very large factor, it does not vanish asymptotically. Hence viewing EMA or IA as noise reduction schemes, we consider IA to be far more aggressive. Secondly, EMA implicitly assumes that more recent iterates are better, or otherwise more important, than the previous iterates. While justified initially (partially explaining LH's efficacy in accelerating optimisation), it is less so in the late stage of training. We nonetheless believe LH could be of great value. Like our proposed methods, LH features weight-space average to achieve optimisation and generalisation benefits, however LH maintains different update rules for the *fast* and *slow* weights, and uses exponentially moving average to update the parameters. In this section, we both comment on the key theoretical differences between Gadam and Lookahead and make some preliminary practical comparisons. We also offer an attempt to bring together the *optimisation* benefit of Lookahead and the *generalisation* benefit of Gadam, with promising preliminary results.

B.1 Major Differences between Gadam and Lookahead

Averaging method. Lookahead opts for a more complicated averaging scheme: they determine the 'fast'- and 'slow'- varying weights during optimisation, and maintains an EMA to average the weight. On the other hand, Gadam uses a more straightforward simple average. As we discussed in the main text, EMA is more theoretically justified during the initial rather than later stage of training. This can also be argued from a Bayesian viewpoint following Maddox et al. (2019), who argued that iterates are simply the draws from the posterior predictive distribution of the neural network, where as averaging leads to a rough estimation of its posterior mean. It is apparent that if the draws from this distribution are *equally* good (which is likely to be the case if we start averaging only if validation metrics stop improving), assigning the iterates with an exponential weight just based on when they are drawn constitutes a rather arbitrary prior in Bayesian sense.

Averaging frequency. Lookahead averages every k iterations whereas in Gadam, while possible to do so as well, by default averages much less frequently. We detail our rationale for this in Section 7.1.

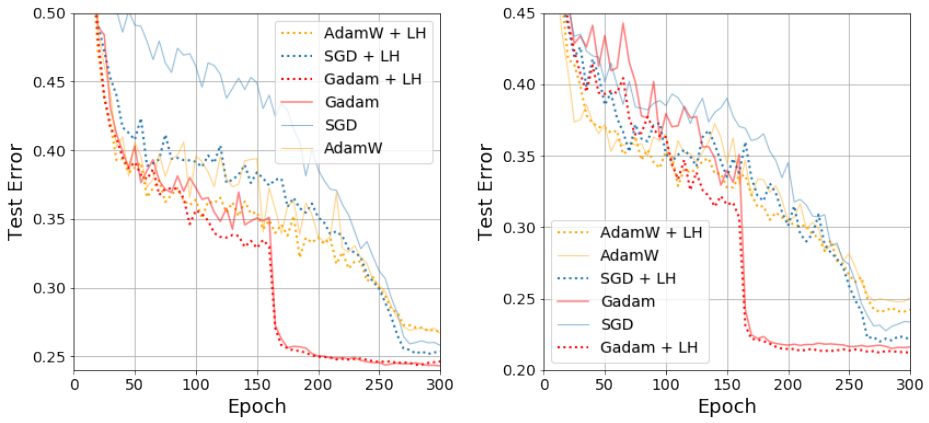
Starting point of averaging. While Lookahead starts averaging at the beginning of the training, Gadam starts averaging either from a pre-set starting point or an automatic trigger

(for GadamAuto). While authors of Lookahead (Zhang et al., 2019b) argue that starting averaging eliminates the hyperparameter on when to start averaging, it is worth noting that Lookahead also introduces two additional hyperparameters α and k, which are non-trivially determined from grid search (although the authors argue that the final result is not very sensitive to them).

We believe the difference here is caused by the different design philosophies of Gadam and Lookahead: by using EMA and starting averaging from the beginning, Lookahead benefits from faster convergence and some generalisation improvement whereas in Gadam, since the averages of iterates are not used during training to promote independece between iterates, Gadam does not additionally accelerate optimisation but, by our theory, should generalise better. As we will see in the next section, this theoretical insight is validated by the experiments and leads to combinable benefits.

Empirical Comparison We make some empirical evaluations on CIFAR-100 data-set with different network architectures, and we use different base optimiser for Lookahead. For all experiments, we use the author-recommended default values of $k = 5$ (number of lookahead steps) and $\alpha = 0.5$. We focus on the combination of Lookahead and adaptive optimisers, as this is the key focus of this paper, although we do include results with Lookahead with SGD as the base optimiser.

We first test AdamW and SGD with and without Lookahead and the results are in Figure 11. Whilst SGD + LH outperforms SGD in final test accuracy by a rather significant margin in both architectures, Lookahead does not always lead to better final test accuracy in AdamW (although it does improve the convergence speed and reduce fluctuations in test error during training, which is unsurprising as EMA shares similar characteristics with IA in reducing sensitivity to gradient noise). On the other hand, it is clear that Gadam delivers both more significant and more consistent improvements over AdamW, both here and in the rest of the paper.



(a) VGG-16 (b) PRN-110

Figure 11: Test accuracy of Lookahead in CIFAR-100 against number of epochs.

Nonetheless, we believe that Lookahead, being an easy-to-use plug-in optimiser that clearly improves convergence speed, offers significant combinable potential with Gadam,

which focuses on generalisation. Indeed, by using Lookahead *before* the 161st epoch where we start IA, and switching to IA *after* the starting point, we successfully combine Gadam and LH into a new optimiser which we term Gadam + LH. With reference to Figure 11, in VGG-16, Gadam + LH both converges at the fastest speed in all the optimisers tested and achieves a final test accuracy only marginally worse than Gadam (but still stronger than all others). On the other hand, in PRN-110, perhaps due to the specific architecture choice, the initial difference in convergence speed of all optimisers is minimal, but Gadam + LH clearly performs very promisingly in the end: it is not only stronger than our result without Lookahead in Figure 11(b), but also, by visual inspection, significantly stronger than the SGD + LH results on the same data-set and using the same architecture reported in the original Lookahead paper (Zhang et al., 2019b).

Appendix C. Experiment Setup

Unless otherwise stated, all experiments are run with PyTorch 1.1 on Python 3.7 Anaconda environment with GPU acceleration. We use one of the three possible GPUs for our experiment: NVIDIA GeForce GTX 1080 Ti, GeForce RTX 2080 Ti or Tesla V100. We always use a single GPU for any single run of experiment.

C.1 Validating Experiments

VGG-16 on CIFAR-100. In this expository experiment, we use the original VGG-16 *without* batch normalisation (batch normalisation has non-trivial impact on conventional measures of sharpness and flatness. See Li et al. (2018)). We conduct all experiments with initial learning rate 0.05. For fair comparison to previous literature, we use the linear decay schedules advocated in Izmailov et al. (2018), for both SGD and IA. For IA we run the set of terminal learning rates during averaging $\{0.03, 0.01, 0.003\}$, whereas for SGD we decay it linearly to 0.0005

C.2 Language Modelling Experiments

In language modelling experiments, we use the codebase provided by `https://github.com/salesforce/awd-lstm-lm`. For ASGD, we use the hyperparameters recommended by Merity et al. (2017) and set the initial learning rate to be 30. Note that in language experiments, consistent with other findings decoupled weight decay seems to be not as effective L_2, possibly due to LSTM could be more well-regularised already, and that batch normalisation, which we argue to be central to the efficacy of decoupled weight decay, is not used in LSTM. Thus, for this set of experiments we simply use Adam and Padam as the iterates for Gadam and GadamX. For Adam/Gadam, we tune the learning rate by searching initial learning rate in the range of $\{0.0003, 0.001, 0.003, 0.01\}$ and for Padam and GadamX, we set the initial learning rate to be 1 and partially adaptive parameter $p = 0.2$, as recommended by the authors (Chen and Gu, 2018). We further set the weight decay to be their recommended value of 1.2×10^{-6}. For the learning rate schedule, we again follow Merity et al. (2017) for a piece-wise constant schedule, and decay the learning rate by a factor of 10 at the $\{100, 150\}$-th epochs for all experiments without using iterate averaging. For experiments with iterate averaging, instead of decaying the learning rate by half before averaging starts,

Table 8: Baseline Results from Previous Works

Network	Optimiser	Accuracy/Perplexity	Reference
CIFAR-100			
VGG-16	SGD	73.80	Huang and Wang (2018)
VGG-16	FGE	74.26	Izmailov et al. (2018)
PRN-164	SGD	75.67	He et al. (2016b)
PRN-110	SGD	76.35	online repository**
ResNet-164	FGE	79.84	Izmailov et al. (2018)
ResNeXt-29	SGD	82.20	Xie et al. (2017)
ResNeXt-29	SGD	81.47	Bansal et al. (2018)
CIFAR-10			
VGG-19	SGD	93.34	online repository**
VGG-16	SGD	93.90	Huang and Wang (2018)
PRN-110	SGD	93.63	He et al. (2016b)
PRN-110	SGD	95.06	online repository**
ImageNet 32×32			
WRN-28-10	SGD	59.04/81.13*	Chrabaszcz et al. (2017)
Modified WRN	SGD	60.04/82.11*	McDonnell (2018)
PTB			
LSTM 3-layer	NT-ASGD	61.2/58.8***	Merity et al. (2017)

Notes:
* Top-1/Top-5 Accuracy
** Link: `https://github.com/bearpaw/pytorch-classification`
*** Validation/Test Perplexity

we keep the learning rate constant throughout to make our experiment comparable with the ASGD schedule. We run all experiments for 200 (instead of 500 in Merity et al. (2017)) epochs.

Learning rate schedule. As discussed in the main text, the experiments shown in Table 10 and Figure 8 are run with constant schedules (except for Padam). Padam runs with a step decay of factor of 10 at $\{100, 150\}$-th epochs. However, often even the adaptive methods such as Adam are scheduled with learning rate decay for enhanced performance. Therefore, we also conduct additional scheduled experiments with Adam, where we follow the same schedule of Padam. The results are shown in Appendix Section D.2.

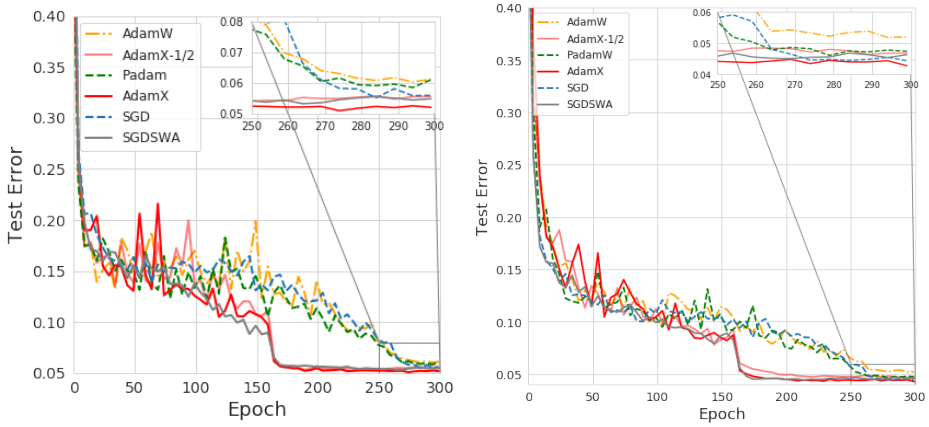
C.3 Experiment Baselines

To validate the results we obtain and to make sure that any baseline algorithms we use are properly and fairly tuned, we also survey the previous literature for baseline results where the authors use same (or similar) network architectures on the same image classification/language tasks, and the comparison of our results against theirs is presented in Table 8. It is clear that for most of the settings, our baseline results achieve similar or better performance compared to the previous work for comparable methods; this validates the rigour of our tuning process.

Appendix D. Additional Experimental Results

D.1 Testing Performance of CIFAR-10

We report the testing performance of VGG-16 and PRN-110 on CIFAR-10 in Figure 12 and Table 9. Perhaps due to the fact that CIFAR-10 poses a simpler problem compared to CIFAR-100 and ImageNet in the main text, the convergence speeds of the optimisers differ rather minimally. Nonetheless, we find that GadamX still outperforms all other optimisers by a non-trivial margin in terms of final test accuracy.



(a) VGG-16 (b) PRN-110

Figure 12: Test Error on CIFAR-10

Table 9: Top-1 Test Accuracy on CIFAR-10 Data-set

Architecture	optimiser	Test Accuracy
VGG-16	SGD	94.14±0.37
	SWA	94.69±0.36
	Adam(W)	93.90 ±0.11
	Padam(W)	94.13 ±0.06
	Gadam	94.62±0.15
	GadamX	**94.88**±0.03
PRN-110	SGD	95.40±0.25
	SWA	95.55±0.12
	Adam(W)	94.69±0.14
	Padam(W)	95.28±0.13
	Gadam	95.27±0.02
	GadamX	**95.95**±0.06

D.2 Word Level Language Modelling with Learning Rate Schedules and Non-monotonic Trigger

Table 10: Validation and Test Perplexity on Word-level Language Modelling.

Data-set	optimiser	Perplexity	
		Validation	Test
PTB	ASGD	64.88±0.07	61.98±0.19
	Adam	65.96±0.08	63.16±0.24
	Padam	65.69±0.07	62.15±0.12
	Gadam	**61.35**±0.05	**58.77**±0.08
	GadamX	63.49±0.19	60.45±0.04

Word-level Language Modelling on PTB Here we include additional results on word-level language modelling using *scheduled* Adam and NT-ASGD, where the point to start averaging is learned non-monotonically and automatically. Where scheduling further improves the Adam performance marginally, the automatically triggered ASGD actually does not perform as well as the manually triggered ASGD that starts averaging from 100th epoch onwards, as we discussed in the main text - this could be because that ASGD converges rather slowly, the 200-epoch budget is not sufficient, or the patience (we use patience = 10) requires further tuning. Otherwise, our proposed Gadam and GadamX without IA schedules still outperform the variants tested here *without careful learning rate scheduling*. The results are summarised in Figure 13 and Table 11.

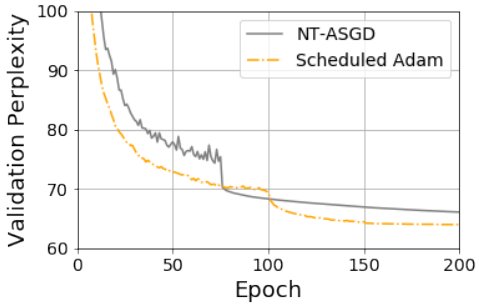


Figure 13: Validation Perplexity of NT-ASGD and Scheduled Adam on 3-layer LSTM PTB Word-level Modelling.

Data-set	optimiser	Perplexity	
		Validation	Test
PTB	NT-ASGD	66.01	64.73
	Scheduled Adam	63.99	61.51
	Gadam (Ours)	**61.35**	**58.77**
	GadamX (Ours)	63.49	60.45

Table 11: Validation and Test Perplexity on Word-level Language Modelling. The Gadam(X) results are lifted from Table 10.

D.3 Relation between Improvement from Averaging and Number of Parameters in Previous Work

In this section we demonstrate that our claim that there should be a dependence on number of parameters P on the margin of improvement from averaging is also present in previous works that use IA or a related ensemble method. Here we use the results from Table 1 of Izmailov et al. (2018). Since the different network architectures are trained with different budget of epochs which make the direct comparison

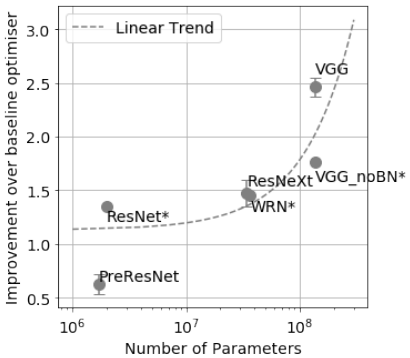


Figure 14: Number of parameters P against improvement margin for both results obtained by us and in Izmailov et al. (2018) (annotated with aster-

of SWA results difficult, we instead consider their FGE (Garipov et al., 2018) results which the author argue to have the similar properties to and that is actually approximated by SWA. We plot their result along with us in Figure 14. While we do not obtain a perfect linear relationship possibly due to a wide range of possible interfering factors such as difference in architecture, use of batch normalisation, choice of optimiser and hyperparameter tuning, again we nevertheless observe that there exists a roughly positive relationship between P and the margin of test improvement.

D.4 Linear vs Step Scheduling

In this work, for the *baseline* methods in image classification tasks we use *linear* instead of the more conventionally employed *step* scheduling because we find linear scheduling to generally perform better in the experiments we conduct. In this section, we detail the results of these experiments, and in this section, 'linear' refers to the schedule introduced in Section 6.1 and 'step' refers to the schedule that reduces the learning rate by a factor of 10 in $\{150, 250\}$ epochs for 300-epoch experiments (CIFAR datasets), or in $\{25, 40\}$ epochs for 50-epoch experiments (ImageNet dataset). The results are shown in Table 12.

Table 12: Testing performance of linear and step learning rate schedules on baseline methods on CIFAR-100.

Architecture	Optimiser	Step	Linear
VGG-16	SGD	73.28	**74.15**
	AdamW	73.20	**73.26**
	Padam	74.46	**74.56**
PRN-110	SGD	**77.23**	77.22
	AdamW	75.27	**75.47**
	Padam	73.95	**77.30**

Appendix E. Importance of Iterate Averaging for Convergence

We argue that despite of the universal practical use of the final iterate of optimisation, it is heuristically motivated and in most proofs of convergence, some form of iterative averaging is required and used implicitly to derive the theoretical bounds. For β-Lipschitz, convex empirical risks, denoted the (overall) loss L. The difference between the $t+1$'th iterate and the optimal solution $L^*_{\boldsymbol{w}}$ can be bounded. The sum of differences along the trajectory (known as the *regret*) telescopes, hence resulting in a convergence rate for the average regret which is an upper bound for the loss of the average point (Nesterov, 2013; Duchi, 2018):

$$\begin{aligned} \delta L = L_{\boldsymbol{w}_{t+1}} - L_{\boldsymbol{w}^*} &\leq \nabla L_{\boldsymbol{w}_t}(\boldsymbol{w}_{t+1} - \boldsymbol{w}^*) + \frac{\beta}{2}||\boldsymbol{w}_{t+1} - \boldsymbol{w}_t||^2 \\ \mathbb{E}(\delta L) &\leq \hat{\nabla} L_{\boldsymbol{w}_t}(\boldsymbol{w}_t - \boldsymbol{w}^*) - (\alpha - \frac{\beta\alpha^2}{2})||\hat{\nabla} L_{\boldsymbol{w}_t}||^2 + \alpha\sigma_t^2 \end{aligned} \tag{77}$$

where $\hat{\nabla} L_{\boldsymbol{w}_t}$ is the noisy gradient at $\boldsymbol{w}_t$ and σ_t^2 is its variance: $\text{Var}(\hat{\nabla} L_{\boldsymbol{w}_t})$. Noting that $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha \hat{\nabla} L_{\boldsymbol{w}_t}$:

$$\frac{R}{T} = \mathbb{E}\Big[\frac{1}{T}\sum_{t=1}^{T-1} L_{\boldsymbol{w}_{t+1}} - L_{\boldsymbol{w}^*}\Big] \tag{78}$$

Using Jensen's inequality, we have:

$$\begin{aligned} \frac{R}{T} &\leq \frac{1}{T}\sum_{t=0}^{T-1} \frac{||\boldsymbol{w}_t - \boldsymbol{w}^*||^2 - ||\boldsymbol{w}_{t+1} - \boldsymbol{w}^*||^2}{2\alpha} + \alpha\sigma_t^2 \\ \mathbb{E}[L_{\frac{1}{T}\sum_{t=1}^{T-1} \boldsymbol{w}_{t+1}} - L_{\boldsymbol{w}^*}] &\leq \frac{R}{T} \leq \frac{||\boldsymbol{w}_0 - \boldsymbol{w}^*||^2}{2\alpha T} + \alpha\sigma_m^2 \end{aligned} \tag{79}$$

where $\sigma_m^2 = \arg\max_{\boldsymbol{w}_t} \mathbb{E}||\hat{\nabla} L_{\boldsymbol{w}_t} - \nabla L_{\boldsymbol{w}_t}||^2$, and R is the regret. Setting $\alpha = (\beta + \sigma\frac{\sqrt{T}}{D})^{-1}$ in equation 78 gives us the optimal convergence rate. Similar convergence results can be given for a decreasing step size $\alpha_t \propto t^{-1/2}\alpha_0$. For adaptive optimisers, the noisy gradient is preconditioned by some non-identity matrix $\bar{\boldsymbol{B}}^{-1}$:

$$\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{w}_k - \alpha \bar{\boldsymbol{B}}^{-1} \nabla L_k(\boldsymbol{w}) \tag{80}$$

Methods of proof (Reddi et al., 2019; Tran et al., 2019) rely on bounding the regret $\mathcal{O}(\sqrt{T})$ and showing that the average regret $\frac{R}{T} \to 0$ and Equation 78 explicitly demonstrates that the average regret is an upper bound on the expected loss for the average point in the trajectory. Hence existing convergence results in the literature prove convergence for the iterate average, but not the final iterate.

Optimal learning rates. Setting $\alpha = (\beta + \sigma\frac{\sqrt{T}}{D})^{-1}$ gives us the optimal convergence rate of $\frac{\beta R^2}{T} + \frac{\sigma D}{\sqrt{T}}$. Similar convergence results can be given for a decreasing step size $\alpha_t \propto t^{-1/2}\alpha_0$ (Duchi, 2018) when the number of iterations T is not known in advance. Given the use of both iterate averaging and learning rate schedule in the proofs, it is difficult to understand the relative importance of the two and how this compares with the typical heuristic of using the final point.

E.1 Relevance of Local Geometry Arguments

One argument as to why IA improves generalisation (Izmailov et al., 2018) is about the local geometry of the solution found: Izmailov et al. (2018) discuss the better generalisation of SWA to the "flatter" minimum it finds. The same argument is used to explain the apparent worse generalisation of adaptive method: Wu et al. (2018a) showed empirically that adaptive methods are not drawn to flat minima unlike SGD. From both Bayesian and minimum description length arguments (Hochreiter and Schmidhuber, 1997), flatter minima generalise better, as they capture more probability mass. He et al. (2019) formalise the intuition under the assumption of a shift between the training and testing loss surface and investigate the presence of "flat valleys" in loss landscape. They argue that averaging leads to a biased solution to the "flatter" valley, which has *worse* training but *better* generalisation performance due to the shift. This suggests IA has an inherent regularising effect, which contrasts with our previous claim that IA should improve both.

However, one issue in the aforementioned analysis, is that they train their SGD baseline and averaged schemes on different learning rate schedules. While this is practically justified, and even desirable, exactly because IA performs better with high learning rate as argued, for *theoretical analysis* on the relevance of the landscape geometry to solution quality, it introduces interfering factors. It is known that the learning rate schedule can have a significant impact on both performance and curvature (Jastrzebski et al., 2020). We address this by considering IA and the iterates, for the same learning rate to specifically alleviate this issue. We use the VGG-16 *without* BN[6] using both AdamW/Gadam and SGD/SWA. In addition to the test and training statistics, we also examine the spectral norm, Frobenius norm and trace which serve as different measures on the "sharpness" of the solutions using the spectral tool by Granziol et al. (2019); we show the results in Table 13. We find a rather mixed result with

Table 13: Performance and Hessian-based sharpness metrics on CIFAR-100 using VGG-16. The numerical results for iterates are in brackets.

Optimiser	Terminal LR	Train acc.	Test acc.	Spectral Norm	Frobenius Norm	Trace
AdamW	$3E{-}6$	99.93	69.43	62	$9.3E{-}4$	$4.7E{-}5$
Gadam	$3E{-}5$	**99.97** (94.12)	69.67 (67.16)	120 (2500)	$1.4E{-}3$(0.86)	$6.4E{-}5$($2.2E{-}3$)
Gadam	$3E{-}4$	98.62 (89.34)	**71.55** (64.68)	43 (280)	$1.1E{-}3$ (0.023)	$1.1E{-}4$ ($5.1E{-}4$)
SGD	$3E{-}4$	99.75	71.64	4.40	$1.2E{-}5$	$4.7E{-}6$
SWA	$3E{-}3$	**99.98** (98.87)	71.32 (69.88)	1.85 (14.6)	$4.4E{-}6$ ($1.3E{-}4$)	$1.1E{-}6$ ($8.6E{-}5$)
SWA	$3E{-}2$	91.58 (77.29)	**73.40** (63.42)	1.35 (12.0)	$8.4E{-}6$ ($7.0E{-}5$)	$1.8E{-}5$ ($9.8E{-}5$)

respect to the local geometry argument. While averaging indeed leads to solutions with lower curvature, we find no clear correlation between flatness and generalisation. One example is that compared to SGD, the best performing Gadam run has 14× larger spectral norm, 92× larger Frobenius norm and 23× larger Hessian trace, yet the test accuracy is only 0.09% worse. Either our metrics do not sufficiently represent sharpness, which is unlikely since we included multiple metrics commonly used, or that it is not the most relevant *explanation* for the generalisation gain. We hypothesise the reason here is that the critical assumption, upon which the geometry argument builds, that there exist only *shifts* between test and train surfaces is unsound despite a sound analysis *given* that. For example, recent work has shown under certain assumptions that the true risk surface is *everywhere* flatter than the empirical counterpart Granziol et al. (2020b). Furthermore, for any arbitrary learning rate, as predicted IA helps *both* optimisation and generalisation *compared to iterates of the same learning rate*; any trade-offs between optimisation and generalisation seem to stem from the choice of *learning rates* only.

6. It is argued that BN impacts the validity of conventional measures of sharpness Liu et al. (2019) hence we deliberately remove BN here, nor do we tune optimisers rigorously, since the point here is for theoretical exposition instead of empirical performance..