# HOW GRADIENT ESTIMATOR VARIANCE AND BIAS COULD IMPACT LEARNING IN NEURAL CIRCUITS

**Arna Ghosh**
McGill University &
Mila-Quebec AI Institute
Montréal, QC, Canada
arna.ghosh@mail.mcgill.ca

**Yuhan Helena Liu**
University of Washington
Seattle, WA, USA
Mila-Quebec AI Institute
Montréal, QC, Canada

**Guillaume Lajoie**
Unversite de Montréal &
Mila-Quebec AI Institute
Montréal, QC, Canada

**Konrad Körding**
University of Pennsylvania
Philadelphia, PA, USA
CIFAR Learning in Machines & Brains
Toronto, ON, Canada

**Blake A. Richards**
McGill University, Mila-Quebec AI Institute &
Montreal Neurological Institute
Montréal, QC, Canada
CIFAR Learning in Machines & Brains
Toronto, ON, Canada
blake.richards@mcgill.ca

## ABSTRACT

There is growing interest in understanding how real brains may approximate gradients and how gradients can be used to train neuromorphic chips. However, neither real brains nor neuromorphic chips can perfectly follow the loss gradient, so parameter updates would necessarily use gradient estimators that have some variance and/or bias. Therefore, there is a need to understand better how variance and bias in gradient estimators impact learning dependent on network and task properties. Here, we show that variance and bias can impair learning on the training data, but some degree of variance and bias in a gradient estimator can be beneficial for generalization. We find that the ideal amount of variance and bias in a gradient estimator are dependent on several properties of the network and task: the size and activity sparsity of the network, the norm of the gradient, and the curvature of the loss landscape. As such, whether considering biologically-plausible learning algorithms or algorithms for training neuromorphic chips, researchers can analyze these properties to determine whether their approximation to gradient descent will be effective for learning given their network and task properties.

## 1 INTRODUCTION

Artificial neural networks (ANNs) typically use gradient descent and its variants to update their parameters in order to optimize a loss function (LeCun et al., 2015; Rumelhart et al., 1986). Importantly, gradient descent works well, in part, because when making small updates to the parameters, the loss function's gradient is along the direction of greatest reduction.[1] Motivated by these facts, a longstanding question in computational neuroscience is, does the brain approximate gradient descent (Lillicrap et al., 2020; Whittington & Bogacz, 2019)? Over the last few years, many papers show that, in principle, the brain could approximate gradients of some loss function (Murray, 2019; Liu et al., 2021; Payeur et al., 2021; Lillicrap et al., 2016; Scellier & Bengio, 2017). Also inspired by the brain, neuromorphic computing has engineered unique materials and circuits that emulate biological networks in order to improve efficiency of computation (Roy et al., 2019; Li et al., 2018b). But, unlike ANNs, both real neural circuits and neuromorphic chips must rely on approximations to the true gradient. This is due to noise in biological synapses and memristors, non-differentiable operations such as spiking, and the requirement for weight updates that do not use non-local information (which can lead to bias) (Cramer Benjamin et al., 2022; M. Payvand et al., 2020b; Laborieux et al., 2021; Shimizu et al., 2021; Neftci et al., 2017; N. R. Shanbhag et al., 2019). Thus, both areas of research

[1] If we assume a Euclidean metric in weight space.

could benefit from a principled analysis of how learning is impacted by loss gradient variance and bias.

In this work, we ask how different amounts of noise and/or bias in estimates of the loss gradient affect learning performance. As shown in a simple example in Fig. 9, learning performance can be insensitive to some degree of variance and bias in the gradient estimate, and even benefit from it, but excessive amounts of variance and/or bias clearly hinder learning performance. Results from optimization theory shed light on why imperfectly following the gradient—e.g. via stochastic gradient descent (SGD) or other noisy GD settings—can improve generalization in ANNs (Foret et al., 2020; Chaudhari et al., 2019; Yao et al., 2018; Ghorbani et al., 2019). However, most of these results treat unbiased gradient estimators. In contrast, in this work, we are concerned with the specific case of weight updates with intrinsic but known variance and bias, as is often the case in computational neuroscience and neuromorphic engineering. Moreover, we also examine how variance and bias can *hinder* training, because the amount of variance and bias in biologically-plausible and neuromorphic learning algorithms is often at levels that impair, rather than improve, learning (Laborieux et al., 2021), and sits in a different regime than that typically considered in optimization theory.

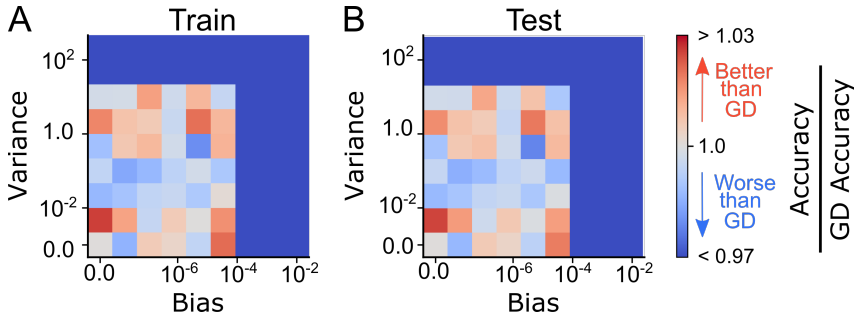


Figure 1: Train and test accuracy of a VGG-16 network trained for 50 epochs (to convergence) on CIFAR-10 using full-batch gradient descent (with no learning rate schedule) with varying amount of variance and bias (as a fraction of the gradient norm) added to the gradient estimates. These results (avg of 20 seeds) indicate that excessive noise and bias harms learning, but a small amount can aid it.

The observations in Fig. 9 give rise to an important question for computational neuroscientists and neuromorphic chip designers alike: what amount of variance and bias in a loss gradient estimate is tolerable, or even desirable? To answer this question, we first observe how variance and bias in gradient approximations impact the loss function in a single parameter update step on the training data. (We also extend this to multiple updates in Appendix A.) We utilize an analytical and empirical framework that is agnostic to the actual learning rule, and derive the factors that affect performance in an imperfect gradient setting. Specifically, we assume that each update is comprised of the contribution of the true gradient of the loss function with respect to the parameter, a fixed amount of bias, and some noise. Similar to Raman et al. (2019), we derive an expression for the change in the loss function after a discrete update step in parameter space: $w(t + \Delta t) = w(t) + \Delta w(t)\Delta t$, where $\Delta t$ is akin to learning rate in standard gradient descent algorithms. We then characterize the impact on learning using the decrease in loss function under an approximate gradient setting as compared to the decrease in loss function when following the true gradient.

Our analysis demonstrates that the impact of variance and bias are independent of each other. Furthermore, we empirically validate our inferences in ANNs, both toy networks, and various VGG configurations (Vedaldi & Zisserman, 2016) trained on CIFAR-10 (Krizhevsky & Hinton, 2009). Our findings can be summarized as follows:

1. The impact of variance is second order in nature. It is lower for networks with a threshold non-linearity, as compared to parameter matched linear networks. Under specific conditions, variance matters lesser for wider and deeper networks.
2. The impact of bias increases linearly with the norm of the gradient of the loss function, and depends on the direction of the gradient as well as the eigenvectors of the loss Hessian.
3. Both variance and bias can help to prevent the system from converging to sharp minima, which may improve generalization performance.

Altogether, our results provide guidelines for what network and task properties computational neuroscientists and neuromorphic engineers need to consider when designing and using noisy and/or biased gradient estimators for learning.

## 2 ESTIMATING THE IMPACT OF VARIANCE AND BIAS ON TRAINING

Our analysis focuses on situations where the synaptic weights of a network, $w$, are trained to optimize a loss, $\mathcal{L}[\boldsymbol{w}]$. We use an auxiliary variable, $t$, to denote different points in this optimization trajectory and $\Delta t$ to denote a single step in this trajectory. (We extend to the multi-step case in Corollary A.2 of the Appendix.) Compared to standard optimization protocols in the ANN literature, $\Delta t$ is akin to the learning rate. Therefore, the loss as measured at one particular point in the trajectory is denoted as $\mathcal{L}[\boldsymbol{w}(t)]$, and the loss at the next point in the trajectory, i.e., after a weight update step, is denoted by $\mathcal{L}[\boldsymbol{w}(t+\Delta t)]$. In the rest of this work, we characterize $\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)]$ as a function of the variance and bias in the weight update.

We assume that $\mathcal{L}$ is twice differentiable and $\Delta t$ is small enough such that $\mathcal{L}[\boldsymbol{w}(t+\Delta t)]$ can be effectively approximated around $\mathcal{L}[\boldsymbol{w}(t)]$ using a second order Taylor series, i.e., higher order terms beyond the second order can be ignored. Furthermore, we define a weight update equation that consists of the gradient, a fixed bias, and some noise. We denote the bias vector as $b\overrightarrow{\boldsymbol{\beta}}$, where $\overrightarrow{\boldsymbol{\beta}}$ is a norm $\sqrt{N}$ vector that indicates the direction of bias in the $N$-dimensional parameter space. A special case of $\overrightarrow{\boldsymbol{\beta}}$ would be $\overrightarrow{\boldsymbol{\beta}} = \overrightarrow{\mathbf{1}}$ when all parameters have the same bias. Similarly, we assume white noise in the gradient estimates. With these assumptions, we define the weight update equation to be:

$$\Delta w(t) := -\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] + b\overrightarrow{\boldsymbol{\beta}} + \sigma\sqrt{f(N)}\hat{n} \tag{1}$$

where $\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]$ denotes the first derivative of the gradient of $\mathcal{L}$ evaluated at $w(t)$ and $\hat{n}$ is a unit vector in the $N$-dimensional parameter space whose elements are zero-mean i.i.d. (see Appendix A for generality of this assumption), such that the total variance of $\Delta w(t)$ is $\sigma^2 f(N)$, where $f(N)$ denotes the dependence of variance on the total number of network parameters, $N$. Finally, we define $\Delta\mathcal{L}_t(b,\sigma^2)$ as the change in $\mathcal{L}[\boldsymbol{w}(t)]$ when performing the aforementioned weight update as compared to the change in $\mathcal{L}[\boldsymbol{w}(t)]$ when the weight update step follows the true gradient, i.e.,

$$\Delta\mathcal{L}_t(b,\sigma^2) = [\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)]]_{(b,\sigma^2)} - [\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)]]_{(0,0)} \tag{2}$$

where $[\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)]]_{(b,\sigma^2)}$ is the change in loss with bias $b$ and variance $\sigma^2$. For brevity, we drop the $t$ is the subscript in the rest of the paper. Formally, we present the following Lemma:

**Lemma 2.1.** ***Second order Taylor series expansion.*** *Assuming a small learning rate, the bias, $b$, and variance, $\sigma^2$, in gradient estimates can be linked to changes in $\mathcal{L}$ upon one weight update step as compared to the change in $\mathcal{L}$ under a true gradient descent weight update step:*

$$\begin{aligned}\mathbb{E}_{\hat{n}}\left[\Delta\mathcal{L}(b,\sigma^2)\right] &= \mathbb{E}_{\hat{n}}\left[[\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)]]_{(b,\sigma^2)} - [\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)]]_{(0,0)}\right] \\ &= b\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], \overrightarrow{\boldsymbol{\beta}}\right\rangle\Delta t + \frac{1}{2}b^2\left\langle\overrightarrow{\boldsymbol{\beta}}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}}\right\rangle\Delta t^2 \\ &\quad -\frac{1}{2}b\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], (\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T)\overrightarrow{\boldsymbol{\beta}}\right\rangle\Delta t^2 + \frac{1}{2}\frac{\sigma^2 f(N)}{N}\operatorname{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]]\Delta t^2\end{aligned} \tag{3}$$

*where $\langle\cdot,\cdot\rangle$ denotes the dot product,* Tr *denotes the Trace operator of a square matrix, orange terms indicate the impact of bias, and the green term indicates the impact of variance.*

In the next section, we present an in-depth analysis of the impact of variance when $f(N)$ is sublinear, as is the case for some learning algorithms, e.g. Equilibrium Propagation (Laborieux et al., 2021). We present a more general analysis along with the proofs of all lemmas and theorems in Appendix A.

### 2.1 ANALYSIS OF THE IMPACT OF VARIANCE AND BIAS ON TRAINING LOSS

One of the corollaries that follow from Lemma 2.1 is that the impact of variance and bias on the expected $\Delta\mathcal{L}_t(b, \sigma^2)$ are independent of each other. Moreover, it is clear that the impact of variance is directly proportional to the trace of $\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]$, i.e., the loss Hessian, and inversely proportional to $N$, i.e., the number of trainable parameters in the network.

Previous work investigating the object categorization loss landscape for feedforward ANNs demonstrated a pronounced effect of width, i.e., increasing width leads to smoother loss landscapes (Li et al., 2018a), thereby leading to lower trace of Hessian (Hochreiter & Schmidhuber, 1994). This empirical observation in deep ANNs implies that increasing the width both lowers the trace of the Hessian in addition to its obviously increasing the number of trainable network parameters. Therefore, we can say directly from Lemma 2.1 that the impact of variance is lower for wider networks. However, the role of depth is more complicated. Notably, increasing depth (with no skip connections) could theoretically make the loss landscape less smooth (Li et al., 2018a). Thus, we begin by analyzing the case of increasing depth in linear feedforward networks. We prove that, in fact, increasing depth leads to lower values for $\frac{\mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]}{N}$, thereby implying a lower impact of variance.

**Theorem 2.2.** ***Increasing depth lowers impact of variance*** *For linear feedforward ANNs, the impact of variance on $\Delta\mathcal{L}_t(0, \sigma^2)$ for a $(L+1)$ layer network is less than that of a $L$ layer network, i.e.,*

$$\Delta\mathcal{L}_t(0, \sigma^2)_{(L+1)-layer} \leq \Delta\mathcal{L}(0, \sigma^2)_{L-layer} \tag{4}$$

*where each layer has $d$ units and weights initialized by the Xavier initialization strategy and the total variance in gradient estimates does not grow with the size of the network, i.e., $f(N)$ is some constant*[2]

Although these assumptions on layer weights may not strictly hold over the course of training, empirical experience suggests that there exists a loose correspondence among these quantities if the conditions hold true during initialization. Taken together, overparameterization in either width or depth leads to a lower impact of variance in gradient estimates on the network's training.

Besides the network architecture, the non-linearity used in the network also impacts the function implemented by the network, and therefore affects the trace of the loss Hessian. Specifically, we demonstrate that the trace of the loss Hessian is lower for networks with a threshold non-linearity with gain less than or equal to 1, such as a ReLU operation, as compared to a linear network. Formally,

**Theorem 2.3.** ***ReLU lowers impact of variance*** *Let $\phi(.)$ denote a threshold non-linearity function. The impact of variance on $\Delta\mathcal{L}_t(0, \sigma^2)$ for a network with such non-linearities is less than that of an equivalent linear network, i.e.,*

$$\Delta\mathcal{L}_t(0, \sigma^2)_{\phi} \leq \Delta\mathcal{L}_t(0, \sigma^2)_{Linear} \tag{5}$$

*where the gain of $\phi(.)$ is less than or equal to 1 (e.g. ReLU).*

Summarily, the standard practices in deep neural networks, like overparameterized networks or the use of ReLU as a non-linearity, lead to a lower impact of variance on the network's training. Interestingly, brains are also arguably overparameterized and characterized by threshold non-linearities that lead to sparse activations. These properties could offer a potential solution to countering any negative impact of variance in gradient estimates in the brain and allow biological agents to reliably train complex tasks despite not being able to actually conduct true gradient descent (Lillicrap et al., 2020). It also suggests neuromorphic chips could tolerate more noise in gradient estimators if they are made larger.

In contrast to the impact of variance, the impact of bias is more nuanced and depends on the norm and direction of gradient, the direction of the bias vector, as well as the eigenvectors of the loss Hessian. Using a few algebraic manipulations, Lemma 2.1 can be extended to show that:

**Theorem 2.4.** *The impact of bias on $\Delta\mathcal{L}_t(b, 0)$ grows linearly with the norm of the gradient.*

$$\Delta\mathcal{L}_t(b, 0) = \frac{1}{2}b^2 Q\Delta t^2 + b\|\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\|P\Delta t \tag{6}$$

*where $P, Q$ are terms that are independent of the norm of the gradient, $\|\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\|$.*

[2]Check Appendix A for an extension to the general case where variance changes with number of parameters.

This result implies that initialization and architectural considerations could play important roles in mitigating the impact of bias on training by limiting the norm of the gradient. In the next section, we will empirically validate these results in linear and shallow networks trained on toy datasets as well as in VGG networks trained on CIFAR-10.

### 2.2 Empirical verification of the variance and bias results

Our experimental setup (illustrated in Fig. 2) is motivated by Raman et al. (2019), and follows the standard student-teacher framework. A "teacher" ANN is initialized with fixed parameters. The task is for a randomly initialized "student" network to learn the input-output mapping defined by the teacher. We defer the reader to the Appendix B for experimental details.

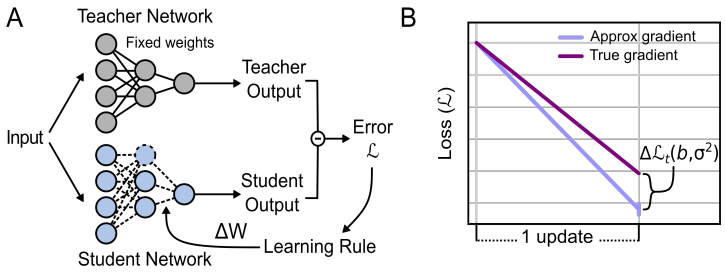


Figure 2: Experimental setup for empirical validation of our results, focusing on $\Delta\mathcal{L}_t(b, \sigma^2)$ to understand how gradient approximations impact learning in one update step. (A) The student-teacher framework motivated by Raman et al. (2019). (B) Illustration of $\Delta\mathcal{L}_t(b, \sigma^2)$ (defined in Eq. (2))

First, we validate our analytical results pertaining to the impact of variance, i.e., Theorems 2.2 and 2.3 in relatively shallow (1-8 hidden layers) fully connected ANNs. We fix the teacher network architecture and use a ReLU non-linearity to threshold the intermediate layer activations. Subsequently, we vary the depth and width of the student network with no non-linearity and observe $\Delta\mathcal{L}_t(0, \sigma^2)$ at different points in the loss landscape. For each such setting, we add a ReLU non-linearity to the student network to validate the effect of a threshold non-linearity on $\Delta\mathcal{L}_t(0, \sigma^2)$. We plot the mean $\Delta\mathcal{L}_t(0, \sigma^2)$ across different points in the loss landscape in Fig. 3. Note that in order to use a sample accurately reflecting the loss landscape that would be encountered during a learning trajectory, we train a control network with the same architecture and non-linearity as the student network using the true gradient and evaluate $\Delta\mathcal{L}_t(0, \sigma^2)$ for each update step along the learning trajectory. In doing so, we are able to observe the desired quantities across a wide range of gradient norm values.

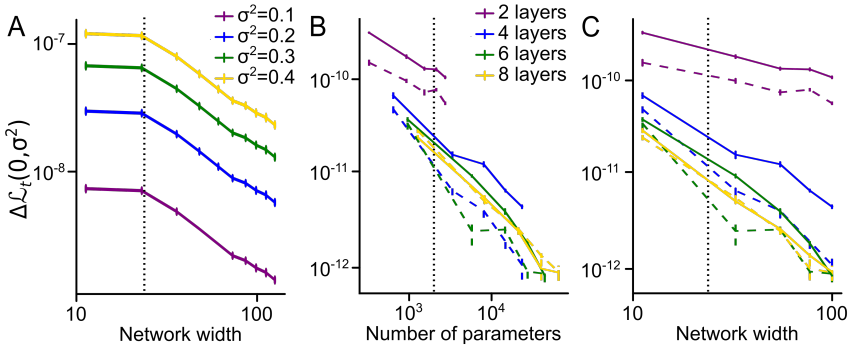


Figure 3: Impact of variance is lower for wider, deeper, and ReLU networks. Solid lines indicate linear networks, dashed lines indicate networks with ReLU non-linearity, and dotted line indicate the teacher network architecture. (A) Varying hidden layer width in one hidden layer linear ANNs, keeping input/output dimensions same. (B) Linear and ReLU networks with varying width and depth plotted with number of parameters in x-axis. (C) Same as B but plotted with network width in x-axis.

Furthermore, we validate Theorems 2.2 and 2.3 on VGG networks trained on the CIFAR-10 dataset. Specifically, we maintain the student-teacher framework and fix the teacher network to be a VGG-19 network trained on CIFAR-10 to 92.6% test accuracy, and we use different networks from the VGG family as student networks (with variance set to $\sigma^2 = 20$). For each network, we use both random weight initialization and Imagenet-pretrained weights to determine whether initialization has an effect on the impact of variance. Fig. 5 demonstrates that our results pertaining to depth and width hold for deep convolutional neural networks (CNNs).

Similarly, we validate Theorem 2.4, first in shallow fully connected ANNs and subsequently in deep CNNs trained on CIFAR-10. Specifically, we demonstrate in Fig. 4 that the impact of bias (with $b = 0.02$) on performance of a linear shallow network grows with the norm of the gradient. Note that the sign duality of $\Delta\mathcal{L}_t(b, 0)$ in this figure follows from Theorem 2.4 where the quantities $P$ and $Q$ depend on the direction of the bias vector with respect to the direction of gradient and the eigenvectors of the loss Hessian (see proof in Appendix A for more details). Therefore, bias can help or hinder learning depending on its direction. This behaviour is in contrast to variance, which always hinders learning for a convex loss (Lemma 2.1). Nonetheless, it is worth noting that when the gradient norm is small as bias increases it always hinders training (see Fig. 4B).

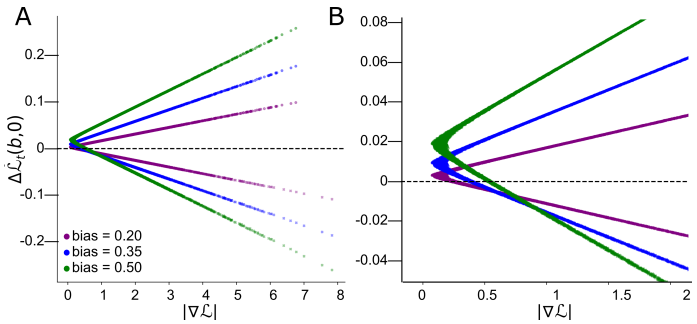


Figure 4: Impact of bias grows linearly with gradient norm and has a quadratic relationship with the amount of bias when the gradient norm tends to 0, i.e., validating expression from Theorem 2.4: $\Delta\mathcal{L}_t(b, 0) = \frac{1}{2}b^2Q\Delta t^2 + b\|\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\|P\Delta t$. (A) Impact of bias grows linearly with gradient norm, with the slope being the amount of bias (see second term in equation). (B) Impact of bias grows quadratically with amount of bias when gradient norm is small (see first term in equation).

Following this, we measure the absolute value of $\Delta\mathcal{L}_t(b, 0)$ and plot its mean across multiple update steps in VGG networks trained on CIFAR-10. In Fig. 6, we show that the relation stated in Theorem 2.4 holds across different VGG architectures, irrespective of the weight initialization. Furthermore, deeper VGG networks have a lower impact of bias owing to a lower gradient norm.

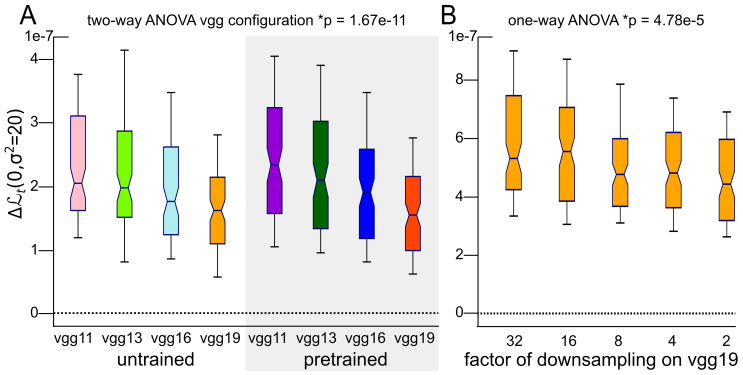


Figure 5: Impact of variance is lower for deeper and wider VGG networks when training on CIFAR-10. (A) Deeper VGG configurations, both Imagenet-pretrained or untrained, exhibit lower impact of variance. (B) Wider VGG-19 configuration networks exhibit lower impact of variance. Factor of downsampling indicates the reduction in the number of filters in convolutional layers and dimensionality of intermediate fully connected layers compared to the original VGG-19 configuration.

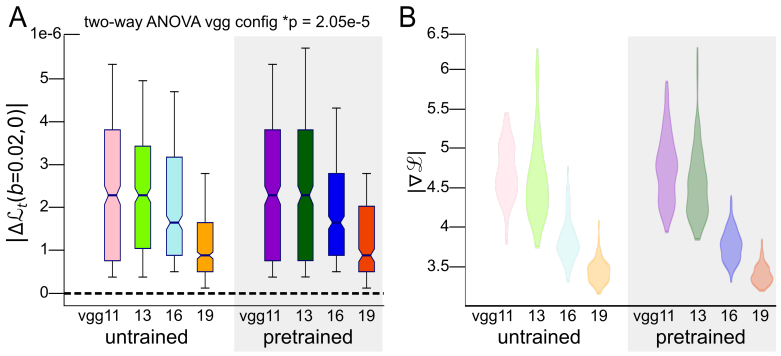


Figure 6: Impact of bias is lower for (A) deeper VGG networks when training on CIFAR-10. (B) Deeper VGG networks exhibit lower gradient norm, thereby mitigating the impact of bias in training.

## 3 Estimating the impact of variance and bias on generalization

In the previous section, we studied how variance and bias in gradient estimates impacts training. However, from a machine learning perspective, an important question is to understand the impact on the system's generalization. Specifically, we ask the question: under what conditions could variance and bias in gradient estimates aid generalization performance? Interestingly, current deep learning practices rely on not following the exact gradient of the loss function to train models, which have been demonstrated to help generalization performance. Some common examples of such practices include stochastic gradient descent (SGD) and dropout[3].

In this section, we use our framework to understand how variance and bias in gradient estimates could alter the learning trajectory, thereby impacting generalization. Specifically, we investigate the conditions under which a parameter update would necessarily lead to descent on the loss landscape. This matters for understanding generalization because the flatness of the loss landscape is a good proxy for generalization (Baldassi et al., 2020; Jiang et al., 2019; Sankar et al., 2021; Tsuzuku et al., 2020; Petzka et al., 2021): flatter minima tend to have better generalization performance than sharper ones. We leverage this viewpoint to understand the impact of variance and bias on generalization. Specifically, we investigate the conditions under which gradient approximations could help to avoid sharp minima, thereby promoting convergence to wide flat minima.

### 3.1 Analysis of the impact of variance and bias descent of narrow minima

To quantify the flatness of loss landscape, we use eigenvalues of the loss Hessian. Without loss of generality, we assume that the loss Hessian is a normal square matrix, i.e., its eigenvectors form an orthogonal basis of the $N$-dimensional parameter space. In order to understand the properties of loss minima that the network could potentially converge to, we derive the sufficiency conditions under which a parameter update ascends (or descends) the loss landscape. This strategy indicates that noise in gradient estimates would lead to the system not descending the loss landscape when the minima is too sharp, which can aid generalization.

**Theorem 3.1.** ***Noise helps avoid sharp minima*** *The sufficient condition under which noise prevents the system from descending the loss landscape, i.e.,* $\mathcal{L}[\boldsymbol{w}(t+\Delta t)] \geq \mathcal{L}[\boldsymbol{w}(t)]$*, is*

$$\lambda_1 \geq \lambda_N \geq \frac{2}{\Delta t}\frac{1}{1+\left(\frac{\sigma}{\|\nabla_{\boldsymbol{w}}\mathcal{L}\|}\right)^2} \tag{7}$$

*where* $\lambda_1 \geq \lambda_2 ... \geq \lambda_N$ *denote the eigenvalues of* $\nabla^2_{\boldsymbol{w}}\mathcal{L}$ *around the minima.*

Interestingly, the condition presented in Theorem 3.1 presents a lower bound for the smallest eigenvalue, i.e., the eigenvalue associated with the direction of least curvature, and connects it to the variance to gradient norm ratio. This ratio can be thought of as the inverse noise ratio when there is

[3]Discussion on the relationship between our analysis and SGD and dropout in the Appendix C.2

no bias in the approximation. Furthermore, we also show in Appendix A that the sufficient condition for descending the loss landscape is similar to the one presented in Theorem 3.1, but provides an upper bound for the largest eigenvalue, i.e., $\lambda_1 \leq \frac{2}{\Delta t}\frac{1}{1+\left(\frac{\sigma}{\|\nabla_{\boldsymbol{w}}\mathcal{L}\|}\right)^2}$. Note that as we get closer to the minimum, the gradient norm would decrease (property of smooth $\mathcal{L}$), and therefore, the inverse noise ratio would increase. Thus, the upper bound on curvature of loss minima that the system can potentially converge to decreases. Taken together, this shows that variance helps the system avoid converging to sharp minima and instead promotes converging to wide flat minima.

Bias, on the other hand, has a more nuanced effect on the learning trajectory. This complexity is unsurprising given the dependence of the impact of bias on direction of the bias vector, the direction of the gradient, and the eigenvectors of the loss Hessian. Once again, we derive the sufficiency condition for when a parameter update step would produce a decrease in the loss function. It turns out this condition depends on both minima flatness and amount of bias relative to the gradient norm.

**Theorem 3.2.** ***Bias prevents descent into local minima dependent on Hessian spectrum.*** *Bias in gradient estimates could prevent converging to a minima. Specifically, the sufficient condition for the weight updates to produce a decrease in $\mathcal{L}$ when using a biased estimate of the gradient is:*

$$\frac{b}{\|\nabla_{\boldsymbol{w}}\mathcal{L}\|} \leq \frac{1}{\sqrt{N}}\frac{2}{1+\psi\Delta t+\sqrt{1+4\psi\Delta t+\psi^2\Delta t^2}} \approx \frac{1}{\sqrt{N}}\frac{1}{(1+\frac{3}{2}\psi\Delta t)} \tag{8}$$

*where $\psi^2 = \sum_i \lambda_i^2$, i.e., the Frobenius norm of the loss Hessian, and $\lambda_1 \geq \lambda_2 ... \geq \lambda_N$ denote the eigenvalues of $\nabla^2_{\boldsymbol{w}}\mathcal{L}$ around the minima and $\nabla^2_{\boldsymbol{w}}\mathcal{L}$ is a normal matrix.*

The condition in Theorem 3.2 indicates that a system that follows a biased gradient approximation will descend the loss landscape until some neighbourhood around a minimum (characterized by the minima flatness and relative bias). Inside this neighbourhood, the system may not descend the loss landscape and therefore not converge to the minimum. The condition depends on minima flatness due to the $\psi$ term (Eq. (8)): sharper (higher curvature) minima should have higher $\psi$, which makes it less likely to satisfy the condition. The condition also depends on the amount of bias relative to gradient norm, i.e.,, higher relative bias makes it less likely to satisfy the condition for descent, which we also support through simulation (see Fig. 7C). Taken together, variance and bias in gradient estimates promote gradient descent dynamics that converge to wide flat minima, which can aid generalization. This effect is empirically demonstrated by our first example of VGG-16 networks trained on CIFAR-10 (Fig. 9).

### 3.2 Empirical verification of the impact of variance and bias on generalization

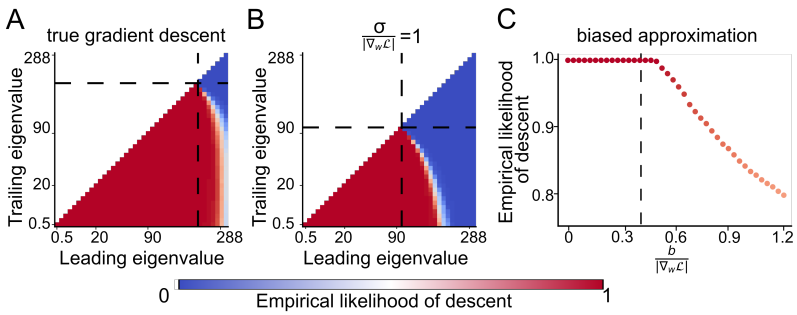


Figure 7: Empirical verification of Theorems 3.1 and 3.2 in linear ANNs. Dashed lines indicate the theoretical limit and colours indicate the empirical probability of descent. (A) True gradient descent. (B) Noise in gradient estimates, $\sigma = \|\nabla_{\boldsymbol{w}}\mathcal{L}\|$. (C) Bias in gradient estimates, ratio $\frac{b}{\|\nabla_{\boldsymbol{w}}\mathcal{L}\|}$ is varied.

We follow the same experimental setting as described in Section 2.2 for verifying Theorems 3.1 and 3.2. Owing to computational bottlenecks in loss Hessian estimation for high-dimensional nonlinear settings, we restrict our empirical validation to training linear networks for optimizing MSE loss. In doing so, we gain full control over the loss Hessian via the input covariance statistics[4]—Fig. 7A & B demonstrate that our inferences from Theorem 3.1 hold empirically. When the leading

[4]for MSE loss at minima, the Hessian is equal to the input covariance matrix barring a constant factor

eigenvalue is *less than* the theoretical limit, a weight update step always leads to a decrease in $\mathcal{L}$; when the trailing eigenvalue is *greater* than the theoretical limit, a weight update step always leads to an increase in $\mathcal{L}$. Furthermore, the theoretical limit is higher for the case when there is no variance than with variance. These observations demonstrate that following a noisy version of the gradient helps the network avoid sharp local minima. Similarly, Fig. 7C demonstrates that our inferences from Theorem 3.2 hold empirically. Specifically, when the ratio of bias to gradient norm is below the theoretical threshold, a weight update step always leads to a decrease in $\mathcal{L}$.

# 4 DISCUSSION

In both computational neuroscience and neuromorphic computing, gradient estimators are known to have variance and bias due to both algorithmic and hardware constraints (Laborieux et al., 2021; M. Payvand et al., 2020a; Lillicrap et al., 2020; Pozzi et al., 2018; Sacramento et al., 2018; Rubin et al., 2021; Roelfsema & Holtmaat, 2018; Bellec et al., 2020; Neftci et al., 2019; Huh & Sejnowski, 2018; Zenke & Neftci, 2021). Using mathematical analysis and empirical verification, we found that the impact of variance and bias on learning is determined by the network size, activation function, gradient norm, and loss Hessian, such that the amount of variance and bias that are permissible or even desirable depends on these properties. Though our empirical verification was done using feedforward ANNs, our mathematical analysis was formulated to be as general as possible, and we believe that our assumptions are reasonable for most cases that computational neuroscientists and neuromorphic chip engineers face. Furthermore, we add a corollary to Lemma 2.1 and discuss extensions of our analysis to multi-step optimization settings in Appendix A (see Corollary A.2). Thus, our work can inform research in these areas by providing a guide for how much variance and bias relative to the gradient norm is reasonable for a given network.

## 4.1 RELATED WORK

Our analysis was inspired in part by recent work by Raman et al. (2019) characterizing bounds on learning performance in neural circuits. They demonstrated that in the absence of noise in the gradient estimator, larger networks are always better at training, but with noise added, there is a size limit beyond which training is impaired. Our work was also informed by research into generalization in deep ANNs, which provided the rationale for our analyses examining the potential for a learning algorithm to descend into sharp minima or not (Foret et al., 2020; Chaudhari et al., 2019; Yao et al., 2018; Ghorbani et al., 2019; Smith et al., 2021). As well, our work has some clear relationship to other work examining the variance of gradient estimators (e.g. Werfel et al. (2003)), but here, we are asking a novel question, namely, how does a given amount of variance and bias impact performance for different network parameters?

More broadly, our work relates to the deep learning literature because modern ANNs rarely use the true loss gradient over the entire dataset to make weight updates. Instead, it is common practice to add noise in different forms, e.g. SGD (Bottou, 2012), dropout (Srivastava et al., 2014), dropconnect (Wan et al., 2013) and label noise (Blanc et al., 2020; HaoChen et al., 2021; Damian et al., 2021). But, the noise structure in these scenarios may differ from assumptions in our work. For example, SGD does not exhibit white noise structure (Xie et al., 2021; Zhu et al., 2019). [5] Likewise, dropout can be thought of as adding noise, but for a truly unbiased estimate of the gradient, one would have to consider all possible configurations of dropped neurons, something quite uncommon in practice.

## 4.2 LIMITATIONS AND FUTURE WORK

This work focused on characterizing the impact of variance and bias in gradient estimates, but it lacks any in-depth analysis of specific bio-plausible or neuromorphic learning algorithms. Similarly, our work only characterizes learning performance, but does not provide a concrete proposal to mitigate excessive noise or bias in gradient approximations. As such, our analyses can be used by other researchers to assess their learning algorithms but they do not directly speak to any specific learning algorithm nor provide mechanisms for improving existing algorithms. Nonetheless, we believe that our work provides a framework to help develop such strategies, and we leave it to future work.

[5] Though several analytical studies have used white noise to model SGD dynamics, and concluded that the SGD noise acts as a regularizer against converging to sharp minima (Li et al., 2021), similar to our analyses.

### REPRODUCIBILITY STATEMENT

We strongly believe that reproducibility is critical for research progress in both machine learning and computational neuroscience. To this end, we have provided thorough experimental details in the appendix, and in the supplementary materials, we have included the code to run all of the experiments and generate the figures. Our code can also be accessed from the project's github repo. We believe that this information will allow the community to validate/replicate our results and further build on our work.

### ACKNOWLEDGEMENTS

The authors would like to thank Abhishek Banerjee, Jonathan Cornford, and Shahab Bakhtiari for insightful discussions and helpful feedback over the course of the project, as well as Zahraa Chorghay and Colleen Gillon for their aesthetic contribution to the manuscript and figures. This research was generously supported by Vanier Canada Graduate scholarship and Healthy Brains for Healthy Lives fellowship (A.G.); NSERC PGS-D and NSF AccelNet IN-BIC program, Grant No. OISE-2019976 AM02 (Y.H.L.); NSERC Discovery Grant RGPIN-2018-04821, FRQS Research Scholar Award, Junior 1 LAJGU0401-253188, Canada Research Chair in Neural Computations and Interfacing (G.L.); NSERC (Discovery Grant: RGPIN-2020-05105; Discovery Accelerator Supplement: RGPAS-2020-00031), Healthy Brains, Healthy Lives (New Investigator Award: 2b-NISU-8) (B.A.R.); CIFAR Learning in Machines and Brains Program (K.K. & B.A.R.); Canada CIFAR AI Chair program (G.L. & B.A.R.). This work was enabled by the material support of NVIDIA in the form of computational resources and support provided by Mila.

## REFERENCES

Carlo Baldassi, Fabrizio Pittorino, and Riccardo Zecchina. Shaping the learning landscape in neural networks around wide flat minima. *Proceedings of the National Academy of Sciences*, 117(1): 161–170, 2020.

Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, 11(1):3625, July 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-17236-y. URL https://www.nature.com/articles/s41467-020-17236-y. Number: 1 Publisher: Nature Publishing Group.

Guy Blanc, Neha Gupta, Gregory Valiant, and Paul Valiant. Implicit regularization for deep neural networks driven by an ornstein-uhlenbeck like process. In *Conference on learning theory*, pp. 483–513. PMLR, 2020.

Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pp. 421–436. Springer, 2012.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019.

Cramer Benjamin, Billaudelle Sebastian, Kanya Simeon, Leibfried Aron, Grübl Andreas, Karasenko Vitali, Pehle Christian, Schreiber Korbinian, Stradmann Yannik, Weis Johannes, Schemmel Johannes, and Zenke Friedemann. Surrogate gradients for analog neuromorphic computing. *Proceedings of the National Academy of Sciences*, 119(4):e2109194119, January 2022. doi: 10.1073/pnas.2109194119. URL https://doi.org/10.1073/pnas.2109194119. Publisher: Proceedings of the National Academy of Sciences.

Alex Damian, Tengyu Ma, and Jason D Lee. Label noise sgd provably prefers flat global minimizers. *Advances in Neural Information Processing Systems*, 34, 2021.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pp. 2232–2241. PMLR, 2019.

Jeff Z HaoChen, Colin Wei, Jason Lee, and Tengyu Ma. Shape matters: Understanding the implicit bias of the noise covariance. In *Conference on Learning Theory*, pp. 2315–2357. PMLR, 2021.

Sepp Hochreiter and Jürgen Schmidhuber. Simplifying neural nets by discovering flat minima. *Advances in neural information processing systems*, 7, 1994.

Dongsung Huh and Terrence J Sejnowski. Gradient descent for spiking neural networks. *Advances in neural information processing systems*, 31, 2018.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2019.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Master's thesis*, 2009.

Axel Laborieux, Maxence Ernoult, Benjamin Scellier, Yoshua Bengio, Julie Grollier, and Damien Querlioz. Scaling Equilibrium Propagation to Deep ConvNets by Drastically Reducing Its Gradient Estimator Bias. *Frontiers in Neuroscience*, 15, 2021. ISSN 1662-453X. URL https://www.frontiersin.org/article/10.3389/fnins.2021.633674.

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. *Advances in neural information processing systems*, 31, 2018a.

Yibo Li, Zhongrui Wang, Rivu Midya, Qiangfei Xia, and J Joshua Yang. Review of memristor devices in neuromorphic computing: materials sciences and device challenges. *Journal of Physics D: Applied Physics*, 51(50):503002, September 2018b. ISSN 0022-3727. doi: 10.1088/1361-6463/aade3f. URL http://dx.doi.org/10.1088/1361-6463/aade3f. Publisher: IOP Publishing.

Zhiyuan Li, Sadhika Malladi, and Sanjeev Arora. On the validity of modeling sgd with stochastic differential equations (sdes). *Advances in Neural Information Processing Systems*, 34, 2021.

Timothy P. Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7: 13276, November 2016. URL http://dx.doi.org/10.1038/ncomms13276.

Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

Yuhan Helena Liu, Stephen Smith, Stefan Mihalas, Eric Shea-Brown, and Uygar Sümbül. Cell-type–specific neuromodulation guides synaptic credit assignment in a spiking neural network. *Proceedings of the National Academy of Sciences*, 118(51), 2021.

M. Payvand, M. E. Fouda, F. Kurdahi, A. Eltawil, and E. O. Neftci. Error-triggered Three-Factor Learning Dynamics for Crossbar Arrays. In *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 218–222, September 2020a. doi: 10.1109/AICAS48895.2020.9073998. Journal Abbreviation: 2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS).

M. Payvand, M. E. Fouda, F. Kurdahi, A. M. Eltawil, and E. O. Neftci. On-Chip Error-Triggered Learning of Multi-Layer Memristive Spiking Neural Networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 10(4):522–535, December 2020b. ISSN 2156-3365. doi: 10.1109/JETCAS.2020.3040248.

Alan Murray and Peter Edwards. Synaptic weight noise during mlp learning enhances fault-tolerance, generalization and learning trajectory. *Advances in neural information processing systems*, 5, 1992.

James M Murray. Local online learning in recurrent networks with random feedback. *ELife*, 8: e43299, 2019.

N. R. Shanbhag, N. Verma, Y. Kim, A. D. Patil, and L. R. Varshney. Shannon-Inspired Statistical Computing for the Nanoscale Era. *Proceedings of the IEEE*, 107(1):90–107, January 2019. ISSN 1558-2256. doi: 10.1109/JPROC.2018.2869867.

Emre O. Neftci, Charles Augustine, Somnath Paul, and Georgios Detorakis. Event-Driven Random Back-Propagation: Enabling Neuromorphic Deep Learning Machines. *Frontiers in Neuroscience*, 11, 2017. ISSN 1662-453X. URL https://www.frontiersin.org/article/10.3389/fnins.2017.00324.

Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

Antonio Orvieto, Hans Kersting, Frank Proske, Francis Bach, and Aurelien Lucchi. Anticorrelated noise injection for improved generalization. *arXiv preprint arXiv:2202.02831*, 2022.

Alexandre Payeur, Jordan Guerguiev, Friedemann Zenke, Blake A. Richards, and Richard Naud. Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *Nature Neuroscience*, 24(7):1010–1019, July 2021. ISSN 1546-1726. doi: 10.1038/s41593-021-00857-x. URL https://doi.org/10.1038/s41593-021-00857-x.

Henning Petzka, Michael Kamp, Linara Adilova, Cristian Sminchisescu, and Mario Boley. Relative flatness and generalization. *Advances in Neural Information Processing Systems*, 34, 2021.

Isabella Pozzi, Sander Bohté, and Pieter Roelfsema. A biologically plausible learning rule for deep learning in the brain. *arXiv preprint arXiv:1811.01768*, 2018.

Dhruva Venkita Raman, Adriana Perez Rotondo, and Timothy O'Leary. Fundamental bounds on learning performance in neural circuits. *Proceedings of the National Academy of Sciences*, 116 (21):10537–10546, 2019. ISSN 0027-8424.

Pieter R. Roelfsema and Anthony Holtmaat. Control of synaptic plasticity in deep cortical networks. *Nature Reviews Neuroscience*, 19(3):166–180, feb 2018. ISSN 14710048. doi: 10.1038/nrn.2018.6.

Thorsteinn Rögnvaldsson. On langevin updating in multilayer perceptrons. *Neural computation*, 6 (5):916–926, 1994.

Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, November 2019. ISSN 1476-4687. doi: 10.1038/s41586-019-1677-2. URL https://doi.org/10.1038/s41586-019-1677-2.

Jonathan E Rubin, Catalina Vich, Matthew Clapp, Kendra Noneman, and Timothy Verstynen. The credit assignment problem in cortico-basal ganglia-thalamic networks: A review, a problem and a possible solution. *European Journal of Neuroscience*, 53(7):2234–2253, 2021.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(9):533–536, 1986.

João Sacramento, Rui Ponte Costa, Yoshua Bengio, and Walter Senn. Dendritic cortical microcircuits approximate the backpropagation algorithm. *Advances in neural information processing systems*, 31, 2018.

Adepu Ravi Sankar, Yash Khasbage, Rahul Vigneswaran, and Vineeth N Balasubramanian. A deeper look at the hessian eigenspectrum of deep neural networks and its applications to regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35:11, pp. 9481–9488, 2021.

Benjamin Scellier and Yoshua Bengio. Equilibrium Propagation: Bridging the Gap between Energy-Based Models and Backpropagation. *Frontiers in Computational Neuroscience*, 11:24, 2017. ISSN 1662-5188. doi: 10.3389/fncom.2017.00024. URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC5415673/.

Genki Shimizu, Kensuke Yoshida, Haruo Kasai, and Taro Toyoizumi. Computational roles of intrinsic synaptic dynamics. *Computational Neuroscience*, 70:34–42, October 2021. ISSN 0959-4388. doi: 10.1016/j.conb.2021.06.002. URL https://www.sciencedirect.com/science/article/pii/S0959438821000672.

Samuel L Smith, Benoit Dherin, David Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. In *International Conference on Learning Representations*, 2021.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Normalized flat minima: Exploring scale invariant definition of flat minima for neural networks using pac-bayesian analysis. In *International Conference on Machine Learning*, pp. 9636–9647. PMLR, 2020.

Andrea Vedaldi and Andrew Zisserman. Vgg convolutional neural networks practical. *Department of Engineering Science, University of Oxford*, 66, 2016.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pp. 1058–1066. PMLR, 2013.

Justin Werfel, Xiaohui Xie, and H Seung. Learning curves for stochastic gradient descent in linear feedforward networks. *Advances in neural information processing systems*, 16, 2003.

James C.R. Whittington and Rafal Bogacz. Theories of Error Back-Propagation in the Brain. *Trends in Cognitive Sciences*, 2019. ISSN 1364-6613. doi: 10.1016/j.tics.2018.12.005. URL https://doi.org/10.1016/j.tics.2018.12.005.

Zeke Xie, Issei Sato, and Masashi Sugiyama. A diffusion theory for deep learning dynamics: Stochastic gradient descent exponentially favors flat minima. In *International Conference on Learning Representations*, 2021.

Zhewei Yao, Amir Gholami, Qi Lei, Kurt Keutzer, and Michael W Mahoney. Hessian-based analysis of large batch training and robustness to adversaries. *Advances in Neural Information Processing Systems*, 31, 2018.

Friedemann Zenke and Emre O Neftci. Brain-inspired learning on neuromorphic substrates. *Proceedings of the IEEE*, 109(5):935–950, 2021.

Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In *International Conference on Machine Learning*, pp. 7654–7663. PMLR, 2019.

# A PROOFS

**Lemma A.1.** ***Second order Taylor series expansion.*** *Let us assume for a small learning rate, $\mathcal{L}[\boldsymbol{w}(t+\Delta t)]$ can be approximated using the Taylor series expansion about the point $w(t)$ and dropping the third and higher order terms. The resulting expression linking bias, b, and variance, $\sigma^2 f(N)$, in gradient estimates to changes in $\mathcal{L}$ upon one weight update step as compared to the change in $\mathcal{L}$ under a true gradient descent weight update step is:*

$$\begin{aligned}\mathbb{E}_{\hat{n}}\left[\Delta Loss(b,\sigma^2)\right] &= \mathbb{E}_{\hat{n}}\left[[\mathcal{L}[\boldsymbol{w}(t+\Delta t)]-\mathcal{L}[\boldsymbol{w}(t)]]_{(b,\sigma^2)}-[\mathcal{L}[\boldsymbol{w}(t+\Delta t)]-\mathcal{L}[\boldsymbol{w}(t)]]_{(0,0)}\right]\\ &= b\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)],\overrightarrow{\boldsymbol{\beta}}\right\rangle\Delta t+\frac{1}{2}b^2\left\langle\overrightarrow{\boldsymbol{\beta}},\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}}\right\rangle\Delta t^2\\ &-\frac{1}{2}b\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)],(\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]+\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T)\overrightarrow{\boldsymbol{\beta}}\right\rangle\Delta t^2\\ &+\frac{1}{2}\frac{\sigma^2 f(N)}{N}\operatorname{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]]\Delta t^2 \end{aligned}\tag{9}$$

*where $N$ denotes the dimensionality of $w(t)$, $f(N)$ indicates how the total noise in gradient estimates changes with the number of parameters in the model, $\hat{n}$ denotes a random unit vector (entries drawn i.i.d.) in the N-dimensional parameter space and $b\overrightarrow{\boldsymbol{\beta}}$ denotes the bias vector, where $\overrightarrow{\boldsymbol{\beta}}$ is a norm $\sqrt{N}$ vector (i.e. a vector with 2-norm equal to $\sqrt{N}$) that indicates the direction of bias in the $N$-dimensional parameter space. A special case of $\overrightarrow{\boldsymbol{\beta}}$ would be $\overrightarrow{\boldsymbol{\beta}}=\overrightarrow{\mathbf{1}}$ when all parameters have the same bias level. Also,* Tr *denotes the Trace operator of a square matrix.*

*Proof.* In order to understand how the task error changes as system parameters evolve, we rely on the second order Taylor series expansion of the task error. We denote the system parameters at time $t$ as $w(t)$ and the corresponding task error as $\mathcal{L}[\boldsymbol{w}(t)]$. We assume the parameter update rule to be a noisy version of the gradient descent update:

$$\Delta w(t) = -\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]+b\overrightarrow{\boldsymbol{\beta}}+\sigma\sqrt{f(N)}\hat{n}\tag{10}$$

Writing out the Taylor series expansion for a small step $\Delta t$ in the above direction,

$$\mathcal{L}[\boldsymbol{w}(t+\Delta t)]=\mathcal{L}[\boldsymbol{w}(t)]+\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)],\Delta w(t)\rangle\Delta t+\frac{1}{2}\left\langle\Delta w(t),\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\Delta w(t)\right\rangle\Delta t^2\tag{11}$$

We compare the reduction in task error while following the above parameter update rule to the reduction in task error while following the uncorrupted or true gradient descent updates. Specifically, we observe the following quantity:

$$\begin{aligned}\Delta\mathcal{L}_t(bias=b,var=\sigma^2) &:= [\mathcal{L}[\boldsymbol{w}(t+\Delta t)]-\mathcal{L}[\boldsymbol{w}(t)]]_{(bias=b,var=\sigma^2)}\\ &\quad-[\mathcal{L}[\boldsymbol{w}(t+\Delta t)]-\mathcal{L}[\boldsymbol{w}(t)]]_{(bias=0,var=0)}\end{aligned}\tag{12}$$

For sake of brevity, we drop the $t$ from the subscript in the rest of the derivations. Plugging in Equations 10 and 11 in Equation 12,

$$\begin{aligned}\Delta\mathcal{L}(b,\sigma^2) &= \left[b\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)],\overrightarrow{\boldsymbol{\beta}}\right\rangle+\sigma\sqrt{f(N)}\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)],\hat{n}\right\rangle\right]\Delta t\\ &+\frac{1}{2}\left[-b\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)],\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}}\right\rangle-\sigma\sqrt{f(N)}\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)],\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\hat{n}\right\rangle\right.\\ &-b\left\langle\overrightarrow{\boldsymbol{\beta}},\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\right\rangle-\sigma\sqrt{f(n)}\left\langle\hat{n},\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\right\rangle\\ &+b\sigma\sqrt{f(n)}\left\langle\overrightarrow{\boldsymbol{\beta}},\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\hat{n}\right\rangle+b\sigma\sqrt{f(n)}\left\langle\hat{n},\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}}\right\rangle\\ &\left.+b^2\left\langle\overrightarrow{\boldsymbol{\beta}},\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}}\right\rangle+\sigma^2 f(N)\left\langle\hat{n},\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\hat{n}\right\rangle\right]\Delta t^2\end{aligned}\tag{13}$$

We can further simplify the above expression by taking the expectation over different samples to evaluate the task error and multiple weight update steps to understand the average impact of bias and variance in parameter update estimates. Under the aforementioned expectation, dot product of a

deterministic vector with the noise vector $\hat{n}$ is assumed to be 0. Therefore, the expression for $\Delta\mathcal{L}$ simplifies to:

$$\begin{aligned}\mathbb{E}_{\hat{n}}[\Delta\mathcal{L}(b,\sigma^2)] = b\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)],\overrightarrow{\boldsymbol{\beta}}\right\rangle\Delta t + \frac{1}{2}\Big[-b\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)],\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}}\right\rangle \\ - b\left\langle\overrightarrow{\boldsymbol{\beta}},\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\right\rangle + b^2\left\langle\overrightarrow{\boldsymbol{\beta}},\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}}\right\rangle \\ +\sigma^2 f(N)\mathbb{E}_{\hat{n}}[\langle\hat{n},\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\hat{n}\rangle]\Big]\Delta t^2 \end{aligned} \tag{14}$$

Furthermore, we assume the following properties for elements of the $N$-dimensional isotropic noise vector $\hat{n}$:

$$\mathbb{E}_{\hat{n}}[\hat{n}_i^2] = \frac{1}{N} \quad \text{And} \quad \mathbb{E}_{\hat{n}}[\hat{n}_i\hat{n}_j] = 0 \quad \forall \quad i \neq j \in 1...N$$

$$\begin{aligned}\text{Therefore,}\quad \mathbb{E}_{\hat{n}}\left[\langle\hat{n},\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\hat{n}\rangle\right] &= \mathbb{E}_{\hat{n}}\left[\sum_{i,j}\hat{n}_i\hat{n}_j\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]_{ij}\right] \\ &= \sum_i \mathbb{E}_{\hat{n}}\left[\hat{n}_i^2\right]\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]_{ii} + \sum_{i\neq j}\sum_j \mathbb{E}_{\hat{n}}\left[\hat{n}_i\hat{n}_j\right]\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]_{ij} \\ &= \frac{1}{N}\sum_i \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]_{ii} = \frac{1}{N}\operatorname{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]]\end{aligned} \tag{15}$$

Note that Eq. (15) is similar to Hutchinson's trace estimator. We can also leverage the following relationship:

$$\begin{aligned}&\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)],\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}}\right\rangle + \left\langle\overrightarrow{\boldsymbol{\beta}},\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\right\rangle \\ &\qquad = \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}} + \overrightarrow{\boldsymbol{\beta}}^T\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] \\ &\qquad\quad \text{[Since } \langle a,b\rangle = a^Tb\text{]} \\ &\qquad = \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}} + \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T\overrightarrow{\boldsymbol{\beta}} \\ &\qquad \text{[Since each term is a scalar, transpose of a term is equal to itself]} \\ &\qquad = \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T(\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T)\overrightarrow{\boldsymbol{\beta}} \\ &\qquad = \left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)],(\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T)\overrightarrow{\boldsymbol{\beta}}\right\rangle\end{aligned} \tag{16}$$

Now, incorporating Equations 15 and 16 in order to simplify Equation 14:

$$\begin{aligned}\mathbb{E}_{\hat{n}}\left[\Delta\mathcal{L}(b,\sigma^2)\right] &= b\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)],\overrightarrow{\boldsymbol{\beta}}\right\rangle\Delta t \\ &\quad + \frac{1}{2}b^2\left\langle\overrightarrow{\boldsymbol{\beta}},\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\overrightarrow{\boldsymbol{\beta}}\right\rangle\Delta t^2 \\ &\quad - \frac{1}{2}b\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)],(\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T)\overrightarrow{\boldsymbol{\beta}}\right\rangle\Delta t^2 \\ &\quad + \frac{1}{2}\frac{\sigma^2 f(N)}{N}\operatorname{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]]\Delta t^2\end{aligned} \tag{17}$$

□

**Note on i.i.d. noise assumption:** Our assumption of i.i.d. noise allows us to derive a closed form for $\Delta\mathcal{L}(b,\sigma^2)$ and investigate the impact of noise in gradient estimates in detail (below). However, it should be noted that there may be learning algorithms that do not adhere to these assumptions, which does mean we cannot state that our analysis is completely learning rule agnostic. Nonetheless, these assumptions (zero in expectation and iid) are sufficiently general that they apply to a number of existing algorithms (e.g. noise perturbation, AGREL, and EQ-prop with the beta parameters selected to have expectation of zero (Murray & Edwards, 1992; Scellier & Bengio, 2017)), and we suspect they can apply to many more.

**Corollary A.2.** ***Extending Taylor series expansion result to multiple sequential update steps.*** *With the same assumptions as Lemma A.1, let $\tilde{\boldsymbol{w}}_k$ be the weights of the network after $k$ sequential noisy updates and $\boldsymbol{w}_k$ be the weights of the network after $k$ steps of gradient descent, given both networks start from the same initial weights, $\boldsymbol{w}_0$. Let us denote the bias and variance vectors at each update step as $\boldsymbol{b}_i$ and $\sigma_i\hat{n}_i$. Then,*

$$\nabla_{\boldsymbol{w}}\mathcal{L}[\tilde{\boldsymbol{w}}_k] = \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k] + \Delta t\left(\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k](\sum_{i=1}^{k}\boldsymbol{b}_i) + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k](\sum_{i=1}^{k}\sigma_i\hat{n}_i)\right) + \mathcal{O}(\Delta t^2) \tag{18}$$

$$\begin{aligned}
\mathbb{E}_{\hat{n}_1,\hat{n}_2\ldots\hat{n}_k}[\mathcal{L}[\tilde{\boldsymbol{w}}_k]] = \mathcal{L}[\boldsymbol{w}_k] &+ \Delta t\sum_{i=1}^{k}\langle\boldsymbol{b}_i, \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_{i-1}]\rangle + \frac{1}{2}\Delta t^2\sum_{i=1}^{k}\left\langle\boldsymbol{b}_i, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_{i-1}]\boldsymbol{b}_i\right\rangle \\
&- \frac{1}{2}\Delta t^2\sum_{i=1}^{k}\left\langle\boldsymbol{b}_i, \left(\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_{i-1}] + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_{i-1}]^T\right)\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_{i-1}]\right\rangle \\
&+ \frac{1}{2}\Delta t^2\frac{1}{N}\sum_{i=1}^{k}\sigma_i^2 f(N)\,\mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_{i-1}]] \\
&+ \Delta t^2\sum_{i=1}^{k}\left\langle\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_{i-1}]\left(\sum_{j=1}^{i-1}\boldsymbol{b}_j\right), \boldsymbol{b}_i - 2\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_{i-1}]\right\rangle
\end{aligned} \tag{19}$$

*Proof.* We will prove the corollary by induction. First, we will use the Taylor series expansion of the gradient to show that Eq. (18) holds for $k = 1$:

$$\begin{aligned}
\nabla_{\boldsymbol{w}}\mathcal{L}[\tilde{\boldsymbol{w}}_1] &= \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_0 - \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_0]\Delta t + (\boldsymbol{b}_1 + \sigma_1\hat{n}_1)\Delta t] \\
&= \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_1 + (\boldsymbol{b}_1 + \sigma_1\hat{n}_1)\Delta t] \\
&= \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_1] + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_1](\boldsymbol{b}_1 + \sigma_1\hat{n}_1)\Delta t + \mathcal{O}(\Delta t^2)
\end{aligned} \tag{20}$$

Assuming Eq. (18) holds for for the first $k$ steps, we will now prove it holds for $(k+1)^{th}$ step. Given that any change in the loss Hessian, i.e. $\nabla^2\mathcal{L}$, will only result in third order effects, we will ignore these changes although we will use a different subscript to denote the step at which the Hessian is computed. This approximation is akin to assuming that the step sizes are small enough such that the loss landscape is roughly quadratic in nature. Similar to the proof for $k = 1$, we use the Taylor series expansion:

$$\begin{aligned}
\nabla_{\boldsymbol{w}}\mathcal{L}[\tilde{\boldsymbol{w}}_{k+1}] &= \nabla_{\boldsymbol{w}}\mathcal{L}[\tilde{\boldsymbol{w}}_k - \nabla_{\boldsymbol{w}}\mathcal{L}[\tilde{\boldsymbol{w}}_k]\Delta t + (\boldsymbol{b}_{k+1} + \sigma_{k+1}\hat{n}_{k+1})\Delta t] \\
&= \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k + \tilde{\boldsymbol{w}}_k - \boldsymbol{w}_k - \nabla_{\boldsymbol{w}}\mathcal{L}[\tilde{\boldsymbol{w}}_k]\Delta t + (\boldsymbol{b}_{k+1} + \sigma_{k+1}\hat{n}_{k+1})\Delta t] \\
&= \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k + \tilde{\boldsymbol{w}}_k - \boldsymbol{w}_k - \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\Delta t + \mathcal{O}(\Delta t^2) + (\boldsymbol{b}_{k+1} + \sigma_{k+1}\hat{n}_{k+1})\Delta t] \\
&= \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_{k+1} + \tilde{\boldsymbol{w}}_k - \boldsymbol{w}_k + (\boldsymbol{b}_{k+1} + \sigma_{k+1}\hat{n}_{k+1})\Delta t] + \mathcal{O}(\Delta t^2)
\end{aligned} \tag{21}$$

Now, we can simplify the expression for $\tilde{\boldsymbol{w}}_k - \boldsymbol{w}_k$ as:

$$\begin{aligned}
\tilde{\boldsymbol{w}}_k - \boldsymbol{w}_k &= \boldsymbol{w}_0 + \Delta t\sum_{i=1}^{k}(-\nabla_{\boldsymbol{w}}\mathcal{L}[\tilde{\boldsymbol{w}}_{i-1}] + \boldsymbol{b}_i + \sigma_i\hat{n}_i) - \boldsymbol{w}_0 + \Delta t\sum_{i=1}^{k}\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_{i-1}] \\
&= -\Delta t\sum_{i=1}^{k}(\nabla_{\boldsymbol{w}}\mathcal{L}[\tilde{\boldsymbol{w}}_{i-1}] - \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_{i-1}]) + \Delta t\sum_{i=1}^{k}(\boldsymbol{b}_i + \sigma_i\hat{n}_i) \\
&= \Delta t\sum_{i=1}^{k}(\boldsymbol{b}_i + \sigma_i\hat{n}_i) + \mathcal{O}(\Delta t^2) \qquad \text{[Using Eq. (18)]}
\end{aligned} \tag{22}$$

Plugging Eq. (22) in Eq. (21), we can show the desired result:

$$\begin{aligned}\nabla_{\boldsymbol{w}}\mathcal{L}[\tilde{\boldsymbol{w}}_{k+1}] &= \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_{k+1} + \Delta t\sum_{i=1}^{k+1}(\boldsymbol{b}_i + \sigma_i\hat{n}_i) + \mathcal{O}(\Delta t^2)] + \mathcal{O}(\Delta t^2) \\ &= \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_{k+1}] + \Delta t\left(\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_{k+1}](\sum_{i=1}^{k+1}\boldsymbol{b}_i) + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_{k+1}](\sum_{i=1}^{k+1}\sigma_i\hat{n}_i)\right) + \mathcal{O}(\Delta t^2)\end{aligned} \tag{23}$$

Now, we know that Eq. (19) is true for $k = 1$ from Lemma A.1. Assuming that it holds for the first $k$ steps, we will show that it holds for the $(k+1)^{th}$ step too. Using Eq. (11), we can write:

$$\begin{aligned}\mathcal{L}[\tilde{\boldsymbol{w}}_{k+1}] &= \mathcal{L}[\tilde{\boldsymbol{w}}_k] + \langle\nabla_{\boldsymbol{w}}\mathcal{L}[\tilde{\boldsymbol{w}}_k], \tilde{\boldsymbol{w}}_{k+1} - \tilde{\boldsymbol{w}}_k\rangle + \frac{1}{2}\left\langle\tilde{\boldsymbol{w}}_{k+1} - \tilde{\boldsymbol{w}}_k, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\tilde{\boldsymbol{w}}_k](\tilde{\boldsymbol{w}}_{k+1} - \tilde{\boldsymbol{w}}_k)\right\rangle \\ &= \mathcal{L}[\tilde{\boldsymbol{w}}_k] + \mathcal{O}(\Delta t^3) \\ &\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k] + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\left(\sum_{i=1}^{k}(\boldsymbol{b}_i + \sigma_i\hat{n}_i)\right)\Delta t, -\nabla_{\boldsymbol{w}}\mathcal{L}[\tilde{\boldsymbol{w}}_k] + \boldsymbol{b}_{k+1} + \sigma_{k+1}\hat{n}_{k+1}\right\rangle\Delta t \\ &+ \frac{1}{2}\left\langle-\nabla_{\boldsymbol{w}}\mathcal{L}[\tilde{\boldsymbol{w}}_k] + \boldsymbol{b}_{k+1} + \sigma_k\hat{n}_{k+1}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\left(-\nabla_{\boldsymbol{w}}\mathcal{L}[\tilde{\boldsymbol{w}}_k] + \boldsymbol{b}_{k+1} + \sigma_{k+1}\hat{n}_{k+1}\right)\right\rangle\Delta t^2\end{aligned}$$

Imposing the independence of the noise statistics and ignoring higher order terms, i.e. $\mathcal{O}(\Delta t^3)$, we get to our desired result:

$$\begin{aligned}\mathbb{E}_{\hat{n}_1,\hat{n}_2\ldots\hat{n}_{k+1}}[\mathcal{L}[\tilde{\boldsymbol{w}}_{k+1}]] &= \mathbb{E}_{\hat{n}_1,\hat{n}_2\ldots\hat{n}_k}[\mathcal{L}[\tilde{\boldsymbol{w}}_k]] - \|\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\|^2\Delta t \\ &+ \frac{1}{2}\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k], \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\right\rangle\Delta t^2 \\ &+ \langle\boldsymbol{b}_{k+1}, \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\rangle\Delta t + \frac{1}{2}\left\langle\boldsymbol{b}_{k+1}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\boldsymbol{b}_{k+1}\right\rangle\Delta t^2 \\ &- \frac{1}{2}\left\langle\boldsymbol{b}_{k+1}, (\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k] + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]^T)\nabla_{\boldsymbol{w}}\mathcal{L}[w_k]\right\rangle\Delta t^2 \\ &+ \frac{1}{2}\frac{\sigma^2_{k+1}f(N)}{N}\operatorname{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]]\Delta t^2 \\ &+ \left\langle\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\left(\sum_{j=1}^{k}\boldsymbol{b}_j\right), \boldsymbol{b}_{k+1} - 2\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\right\rangle\Delta t^2\end{aligned} \tag{24}$$

Now, using Eq. (19) for the $k^{th}$ step and using the relation that $\mathcal{L}[\boldsymbol{w}_{k+1}] = \mathcal{L}[\boldsymbol{w}_k] - \|\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\|^2\Delta t + \frac{1}{2}\left\langle\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k], \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}_k]\right\rangle\Delta t^2$, we get the desired relation for the $(k+1)^{th}$ step. □

**A qualitative description of the effect of bias:** The impact of bias across iterative optimization can be thought of in three key scenarios: (1) the network converges to the same loss basin as gradient descent but the bias alters the specific steady state; (2) bias alters the network training trajectory to the extent that it converges in a loss basin that is different from what gradient descent would converge to; (3) the overall effect of bias hinders network training to the extent it becomes unstable and performance collapses.

In the first case, we can suppose that either the bias is relatively small, or, generally speaking, there is not a strong correlation in the degree of misalignment between the bias vectors and the gradient and Hessian over K steps. In this case, the bias will have an impact that is similar to that of the isotropic noise, and the ultimate result can be understood from a steady state perspective. Notably, at the steady state, the bias would balance the gradient, thereby causing the expected weight change to be 0. Assuming a second order approximation of the loss basin, the bias in the gradients would imply an excess loss of $\frac{1}{2}\boldsymbol{b}_i^T H\boldsymbol{b}_i$ as compared to the minima of the loss basin, which is where gradient descent would have converged to. This would result in minor degradation in performance, as shown in Fig. 1 for lower bias values.

The second case is complicated and ultimately requires further learning rule specific assumptions. Given that our goal in this paper is to remain learning rule agnostic, we can only say that this case

would occur when the bias vectors have a semi-structured relationship to the gradient and the Hessian that pushes the weight updates in very different trajectories compared to gradient descent.

The third case would result from either excessive bias or a very strong relationship between the K bias vectors and their product with the gradient or the Hessian. Thus, we can say that learning rules with very strong or highly systematic bias may not converge.

**Additional assumption on loss function in Theorems A.3 - A.5:** Below, we prove the relationship between network width, depth or non-linearity and the impact of noise on learning for the mean squared error loss. Although we do not prove the result for general loss functions, we believe that our proofs present an outline that other researchers can leverage in future work to extend our results to other loss functions, based on their specific application, and thereby extend our results to guide their network design choices.

**Theorem A.3.** *[$f(N) = \mathcal{O}(1)$ **case] Increasing depth lowers impact of variance** For linear feedforward ANNs, the impact of variance on $\Delta\mathcal{L}(0, \sigma^2)$, for a $(L+1)$ layer network is less than that of a $L$ layer network, i.e.*

$$\Delta\mathcal{L}(0, \sigma^2)_{(L+1)-layer} \leq \Delta\mathcal{L}(0, \sigma^2)_{L-layer} \tag{25}$$

*where each layer has $d$ units and weights initialized by the Xavier initialization strategy and the total variance in gradient estimates does not grow with the size of the network, i.e. $f(N)$ is some constant*

*Proof.* We will first show the theorem holds for $L = 1$, i.e. the impact of variance on a single layer neural network is more than that of a two layer neural network. For the sake of simplicity, we assume $\mathcal{L}$ to be the standard mean squared error loss, i.e. $\mathcal{L} = \frac{1}{|\mathbf{D}|}\sum_{n=1}^{|\mathbf{D}|}(y_n - \hat{y}_n)^2$, where $\hat{y}_n$ denotes the output of the neural network and $\mathbf{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2) \dots (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$ denotes the training dataset. Note that for mean squared error, $\nabla^2_{\hat{y}_n}\mathcal{L} = \frac{2}{|\mathbf{D}|}$. We assume that there is no bias term in determining the unit activations.

Let us denote the weight matrix of a single layer network as $\boldsymbol{W} \in \mathbf{R}^d$ such that $\hat{y}_n = \sum_j W_j x_{nj}$. Thus, we can use the chain rule of differentiation to infer:

$$\begin{aligned}
\nabla_{W_k}\mathcal{L} &= \sum_n (\nabla_{\hat{y}_n}\mathcal{L}) x_{nk} \\
\nabla^2_{W_k}\mathcal{L} &= \sum_n (\nabla^2_{\hat{y}_n}\mathcal{L}) x^2_{nk} \\
\text{Hence,} \quad \mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}] &= \sum_k \nabla^2_{W_k}\mathcal{L} = \frac{2}{|\mathbf{D}|}\sum_{n,k} x^2_{nk}
\end{aligned} \tag{26}$$

Let us assume that $x$ is i.i.d. and normalized to have zero mean and unit variance in all dimensions, i.e. $\frac{1}{|\mathbf{D}|}\sum_n x^2_{ni} = 1$. Therefore, for a $L = 1$ network: $\mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}] = 2d$. Let us denote the weight matrices of a two layer network as $W^{(1)} \in \mathbf{R}^{d\times d}$ and $W^{(2)} \in \mathbf{R}^d$ such that $\hat{y}_n = \sum_{i,j} W^{(2)}_i W^{(1)}_{ij} x_{nj}$. Again,

using the chain rule:

$$\nabla_{W_k^{(2)}}\mathcal{L} = \sum_n (\nabla_{\hat{y}_n}\mathcal{L})(\sum_j W_{kj}^{(1)} x_{nj})$$
$$\nabla^2_{W_k^{(2)}}\mathcal{L} = \sum_n (\nabla^2_{\hat{y}_n}\mathcal{L})(\sum_j W_{kj}^{(1)} x_{nj})^2$$
$$\nabla_{W_{kl}^{(1)}}\mathcal{L} = \sum_n (\nabla_{\hat{y}_n}\mathcal{L}) W_k^{(2)} x_{nl}$$
$$\nabla^2_{W_{kl}^{(1)}}\mathcal{L} = \sum_n (\nabla^2_{\hat{y}_n}\mathcal{L})(W_k^{(2)} x_{nl})^2$$

$$\text{Hence,} \quad \mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}] = \sum_k \nabla^2_{W_k^{(2)}}\mathcal{L} + \sum_{k,l} \nabla^2_{W_{kl}^{(1)}}\mathcal{L}$$
$$= \sum_n (\nabla^2_{\hat{y}_n}\mathcal{L}) \left[ \sum_k (\sum_j W_{kj}^{(1)} x_{nj})^2 + \sum_{k,l} (W_k^{(2)} x_{nl})^2 \right]$$
$$= \frac{2}{|\mathbf{D}|} \sum_n \left[ \sum_k (\sum_j W_{kj}^{(1)} x_{nj})^2 + \sum_k (W_k^{(2)})^2 \sum_l (x_{nl})^2 \right]$$
$$= \frac{\sum_{n,k} (\sum_j W_{kj}^{(1)} x_{nj})^2}{\sum_{n,l} x_{nl}^2} \frac{2}{|\mathbf{D}|} \sum_{n,l} x_{nl}^2 + \frac{2}{|\mathbf{D}|} \sum_k (W_k^{(2)})^2 \sum_{n,l} (x_{nl})^2$$
$$= (\rho + \sum_k (W_k^{(2)})^2) \, \mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{1-layer} \qquad \text{[From Eq. (26)]} \tag{27}$$

$$\text{where,} \quad \rho = \frac{\sum_{n,k} (\sum_j W_{kj}^{(1)} x_{nj})^2}{\sum_{n,l} x_{nl}^2}$$

Now, we assume that all weights are initialized according to the standard Xavier initialization strategy, i.e. $\mathbf{E}_{\boldsymbol{W}}[W_{jk}^{(1)}] = \mathbf{E}_{\boldsymbol{W}}[W_k^{(2)}] = 0$ and $\mathbf{E}_{\boldsymbol{W}}[(W_{jk}^{(1)})^2] = \mathbf{E}_{\boldsymbol{W}}[(W_k^{(2)})^2] = \frac{1}{d}$. Thus, $\sum_k (W_k^{(2)})^2 \approx 1$ where the approximation is stronger for larger values of $d$. Similarly, $\sum_{j,k} (W_{jk}^{(1)})^2 \approx d$. Now, we can use the dot product inequality to simplify $\rho$:

$$\rho = \frac{\sum_{n,k} (\sum_j W_{kj}^{(1)} x_{nj})^2}{\sum_{n,l} x_{nl}^2} \leq \frac{\sum_{n,k} (\sum_j (W_{kj}^{(1)})^2 \sum_j (x_{nj})^2}{\sum_{n,l} x_{nl}^2}$$
$$\implies \rho \leq \sum_{j,k} (W_{kj}^{(1)})^2 \approx d$$
$$\implies \rho + \sum_k (W_k^{(2)})^2 \leq d + 1$$

$$\text{Therefore, from Eq. (27)} \qquad \mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{2-layer} = (\rho + \sum_k (W_k^{(2)})^2) \, \mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{1-layer}$$
$$\leq (1 + d) \, \mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{1-layer} \tag{28}$$

Also, note that by applying the above relations, we can write for a $L = 2$ network: $\mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}] = 2d(\rho + 1) \leq 2d(d + 1)$. Now, the impact of variance, $\Delta\mathcal{L}(0, \sigma^2)$, on a 2-layer network depends on the term $\frac{1}{N} \mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{2-layer}$, where $N = d + d^2$ is the total number of parameters in the network. Similarly, the impact of variance on a 1-layer network depends on the term $\frac{1}{N} \mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{1-layer}$, where $N = d$. Therefore, extending inequality 28:

$$\Delta\mathcal{L}(0,\sigma^2)_{2-layer} = \frac{1}{d+d^2}\operatorname{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{2-layer} \leq \frac{1+d}{d+d^2}\operatorname{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{1-layer}$$
$$\implies \Delta\mathcal{L}(0,\sigma^2)_{2-layer} \leq \frac{1}{d}\operatorname{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{1-layer} = \Delta\mathcal{L}(0,\sigma^2)_{1-layer}$$
$$\text{Hence,} \qquad \Delta\mathcal{L}(0,\sigma^2)_{2-layer} \leq \Delta\mathcal{L}(0,\sigma^2)_{1-layer} \tag{29}$$

We can extend this result to compare $L$ layer networks to $(L+1)$ layer networks. Notably, a $L$ layer network can be thought of as a single-layer network operating on a $(L-1)$ layer network and a $(L+1)$ layer network as a two-layer network operating on an equivalent $(L-1)$ layer network. Using the above result, we can therefore conclude that $\Delta\mathcal{L}(0,\sigma^2)_{(L+1)-layer} \leq \Delta\mathcal{L}(0,\sigma^2)_{L-layer}$. □

**Theorem A.4.** ***[$f(N) = \mathcal{O}(N)$ case] Increasing width over depth lowers impact of variance*** *For linear feedforward ANNs, the impact of variance on $\Delta\mathcal{L}(0,\sigma^2)$, for a $(L+1)$ layer network with $d'$ units in each hidden layer is more than that of a $L$ layer network with same number of parameters, $N$ when $f(N)$ grows linearly with number of parameters if $d' \leq d^2$, where $d$ is the input dimensionality.*

*Proof.* When $f(N) = \mathcal{O}(N)$, the impact of variance of $\Delta\mathcal{L}(0,\sigma^2)$ can be written as:

$$\Delta\mathcal{L}(0,\sigma^2) = \frac{1}{2}\sigma^2\operatorname{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]\Delta t^2$$

and therefore, only depends on the trace. Similar to the proof for Theorem A.3, we will first show the result holds for $L=2$ and extend it for any $L$. As before, we assume that $x$ is i.i.d. and normalized to have zero mean and unit variance in all dimensions, i.e. $\frac{1}{|\mathbf{D}|}\sum_n x_{ni}^2 = 1$. Furthermore, we assume that the weights in all layers are initialized to maintain this zero-mean and unit-variance property in each hidden unit, i.e. $\frac{1}{|\mathbf{D}|}\sum_n h_{ni}^{(l)} = 0$ and $\frac{1}{|\mathbf{D}|}\sum_n \left(h_{ni}^{(l)}\right)^2 = 1$. This property imposes certain constraints on the weight matrices as show below:

$$\frac{1}{|\mathbf{D}|}\sum_n \left(h_{ni}^{(l)}\right)^2 = 1 = \frac{1}{|\mathbf{D}|}\sum_n \left(\sum_j W_{ij}^{(l)} h_{nj}^{(l-1)}\right)^2 = \frac{1}{|\mathbf{D}|}\sum_n\sum_{jk} W_{ij}^{(l)} W_{ik}^{(l)} h_{nj}^{(l-1)} h_{nk}^{(l-1)}$$
$$= \sum_{jk} W_{ij}^{(l)} W_{ik}^{(l)} \left[\frac{1}{|\mathbf{D}|}\sum_n h_{nj}^{(l-1)} h_{nk}^{(l-1)}\right]$$
$$\implies 1 = \sum_j \left(W_{ij}^{(l)}\right)^2 \qquad \text{[i.i.d. assumption on } h^{(l-1)}\text{]} \tag{30}$$

Therefore, $\sum_{ij}\left(W_{ij}^{(l)}\right)^2 = \sum_i 1 = d^{(l)}$, where $d^{(l)}$ is the output dimensionality of layer $l$.

Using these relations and the results from the proof of Theorem A.3, we can write the Trace for a 2-layer network with input dimensionality as $d$, hidden layer dimensionality as $\hat{d}$ and output dimensionality 1 as:

$$\operatorname{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{2-layer} = \frac{1}{|\mathbf{D}|}\sum_n \left[\sum_i \left(h_{ni}^{(1)}\right)^2 + \sum_{ij}\left(W_i^{(2)}\right)^2 x_{nj}^2\right]$$
$$= \sum_{i=1}^{\hat{d}}\left(\frac{1}{|\mathbf{D}|}\sum_n \left(h_{ni}^{(1)}\right)^2\right) + \left[\sum_{i=1}^{\hat{d}}\left(W_i^{(2)}\right)^2\right]\left[\sum_{j=1}^{d}\left(\frac{1}{|\mathbf{D}|}\sum_n x_{nj}^2\right)\right]$$
$$= \hat{d} + d \tag{31}$$

Similarly, the Trace of a 3-layer network with input dimensionality as $d$, hidden layer dimensionality as $d'$, with weights of each layer being i.i.d. with respect to other layers, and output dimensionality

as 1 as:

$$\begin{aligned}\mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{3-layer} &= \frac{1}{|\mathbf{D}|}\sum_n\left[\sum_i\left(h^{(2)}_{ni}\right)^2+\sum_{ij}\left(W^{(3)}_i\right)^2\left(h^{(1)}_{nj}\right)^2+\sum_k\sum_j\left(\sum_i W^{(3)}_i W^{(2)}_{ij}\right)^2 x^2_{nk}\right]\\ &= d'+d'+\left[\sum_i\left(W^{(3)}_i\right)^2\right]\left[\sum_{ij}\left(W^{(2)}_{ij}\right)^2\right]\left[\frac{1}{|\mathbf{D}|}\sum_{nk}x^2_{nk}\right] \qquad \text{[i.i.d. weights]}\\ &= 2d'+dd' \end{aligned} \tag{32}$$

Now, number of parameters in the 2-layer network are: $(d\times\hat{d})+(\hat{d}\times 1)=\hat{d}(d+1)$; and number of parameters in the 3-layer network are: $(d\times d')+(d'\times d')+(d'\times 1)=dd'+(d')^2+d'$. For equal number of parameters in both networks, $\hat{d}=d'+\frac{(d')^2}{d+1}$. Therefore, the trace of the 2-layer network can be equivalently written as: $\mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{2-layer}=d'+\frac{(d')^2}{d+1}+d$. Solving for the condition when the wider 2-layer network has a lower trace than the 3-layer trace, we get to the desired conditions, i.e. $d'\leq d^2$. Moreover, if the networks are extremely wide, i.e. in the regime where the hidden layer dimensionality $d'$ is greater than $d^2$, then the trace of the deeper network will be lower. □

**Theorem A.5.** ***ReLU lowers impact of variance*** *Let $\phi(.)$ denote a threshold non-linearity function. The impact of variance on $\Delta\mathcal{L}$ for a network with such non-linearities is less than that of an equivalent linear network, i.e.*

$$\Delta\mathcal{L}(0,\sigma^2)_\phi \leq \Delta\mathcal{L}(0,\sigma^2)_{Linear} \tag{33}$$

*where the gain of $\phi(.)$ is less than or equal to 1 (e.g. ReLU).*

*Proof.* We will show this result for a two layer neural network but the result holds for other cases too. Let $W^{(1)}\in\mathbf{R}^{d\times d}$ and $W^{(2)}\in\mathbf{R}^d$ denote the weight matrices for the first and second layer respectively. As above, we assume for sake of simplicity that there is no bias term in determining the network activations. Therefore, the output of the network can be expressed as: $\hat{y}_n=\sum_{i,j}W^{(2)}_i W^{(1)}_{ij}x_{nj}$ for a linear network; and $\hat{y}_n=\sum_i W^{(2)}_i\phi\left(\sum_j W^{(1)}_{ij}x_{nj}\right)$ for a network with the aforementioned threshold non-linearity. We used the chain rule to derive an expression for the trace of the hessian for a two-layer linear network in Eq. (27). Similarly, using the chain rule we can compute the trace of Hessian for the non-linear networks:

$$\begin{aligned}\nabla_{W^{(2)}_k}\mathcal{L} &= \sum_n(\nabla_{\hat{y}_n}\mathcal{L})\phi\left(\sum_j W^{(1)}_{kj}x_{nj}\right)\\ \nabla^2_{W^{(2)}_k}\mathcal{L} &= \sum_n(\nabla^2_{\hat{y}_n}\mathcal{L})\phi\left(\sum_j W^{(1)}_{kj}x_{nj}\right)^2\\ \nabla_{W^{(1)}_{kl}}\mathcal{L} &= \sum_n(\nabla_{\hat{y}_n}\mathcal{L})W^{(2)}_k\phi'\left(\sum_j W^{(1)}_{kj}x_{nj}\right)x_{nl}\\ \nabla^2_{W^{(1)}_{kl}}\mathcal{L} &= \sum_n(\nabla^2_{\hat{y}_n}\mathcal{L})\left[W^{(2)}_k\phi'\left(\sum_j W^{(1)}_{kj}x_{nj}\right)x_{nl}\right]^2 \qquad \text{[Assuming } \phi''(.)\approx 0\text{]}\\ \mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_\phi &= \sum_k\nabla^2_{W^{(2)}_k}\mathcal{L}+\sum_{k,l}\nabla^2_{W^{(1)}_{kl}}\mathcal{L}\\ &= \sum_n(\nabla^2_{\hat{y}_n}\mathcal{L})\left[\sum_k\phi\left(\sum_j W^{(1)}_{kj}x_{nj}\right)^2+\sum_{k,l}\left[W^{(2)}_k\phi'\left(\sum_j W^{(1)}_{kj}x_{nj}\right)x_{nl}\right]^2\right] \end{aligned} \tag{34}$$

For a threshold non-linearity with gain $\leq 1$, $\phi(z) \leq z$ and $\phi'(z) \leq 1 \; \forall z \in \mathbf{R}$. Using these two relations, we can further simplify Eq. (34):

$$\mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_\phi \leq \sum_n (\nabla^2_{\hat{y}_n}\mathcal{L}) \left[ \sum_k \left( \sum_j W^{(1)}_{kj} x_{nj} \right)^2 + \sum_{k,l} \left( W^{(2)}_k x_{nl} \right)^2 \right]$$

Plugging in the expression of Trace from Eq. (27):

$$\begin{aligned} &\mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_\phi \leq \mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}]_{Linear} \\ \implies &\Delta\mathcal{L}(0,\sigma^2)_\phi \leq \Delta\mathcal{L}(0,\sigma^2)_{Linear} \end{aligned} \tag{35}$$

The above relation can be easily extended to other network architectures to complete the proof. □

**Note on commonly used non-linear activation functions:** By construction, ReLU fits the definition of $\phi$ and therefore, the above theorem holds. However, the conditions that $\phi(z) \leq z$ and $\phi'(z) \leq 1$ are not satisfied by Sigmoid non-linearity. Due to its non-zero offset, i.e. for zero input the activation function returns 0.5, the first condition ($\phi(z) \leq z$) is not satisfied by the standard Sigmoid function.

**Theorem A.6.** *The impact of bias on $\Delta\mathcal{L}$ grows linearly with the norm of the gradient.*

$$\Delta\mathcal{L}(b,0) = \frac{1}{2} b^2 Q \Delta t^2 + b\|\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\| P \Delta t \tag{36}$$

*where $P, Q$ are terms that are independent of the norm of the gradient, $\|\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\|$.*

*Proof.* Since the impact of bias and variance are decoupled, we can study each of their impacts independently. Therefore, we look at the expression in Lemma A.1 for some bias value $b$ and variance, $\sigma = 0$.

$$\begin{aligned} \mathbb{E}_{\hat{n}}\left[\Delta\mathcal{L}(b,0)\right] &= b\left\langle \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], \overrightarrow{\boldsymbol{\beta}} \right\rangle \Delta t + \frac{1}{2} b^2 \left\langle \overrightarrow{\boldsymbol{\beta}}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] \overrightarrow{\boldsymbol{\beta}} \right\rangle \Delta t^2 \\ &\quad - \frac{1}{2} b \left\langle \nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)], (\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T) \overrightarrow{\boldsymbol{\beta}} \right\rangle \Delta t^2 \\ &= \frac{1}{2} b^2 \left\langle \overrightarrow{\boldsymbol{\beta}}, \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] \overrightarrow{\boldsymbol{\beta}} \right\rangle \Delta t^2 + b\|\nabla_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]\| P \Delta t \end{aligned} \tag{37}$$

$$\tag{38}$$

where, $P = \left\langle \widehat{\nabla_{\boldsymbol{w}}\mathcal{L}}, \overrightarrow{\boldsymbol{\beta}} \right\rangle - \frac{1}{2} \left\langle \widehat{\nabla_{\boldsymbol{w}}\mathcal{L}}, (\nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)] + \nabla^2_{\boldsymbol{w}}\mathcal{L}[\boldsymbol{w}(t)]^T) \overrightarrow{\boldsymbol{\beta}} \right\rangle \Delta t$ and $\widehat{\nabla_{\boldsymbol{w}}\mathcal{L}}$ denotes the unit vector along the gradient. It is clear that $P$ is independent of the norm of the gradient and only depends on the direction of the bias vector, the gradient of the loss function and the eigenvectors of the loss Hessian. □

**Theorem A.7.** ***Noise helps avoid sharp minima*** *The sufficient condition under which noise prevents the system from descending the loss landscape, i.e. $\mathcal{L}[\boldsymbol{w}(t+\Delta t)] \geq \mathcal{L}[\boldsymbol{w}(t)]$, is*

$$\lambda_1 \geq \lambda_N \geq \frac{2}{\Delta t} \frac{1}{1 + \left(\frac{\sigma}{\|\nabla_{\boldsymbol{w}}\mathcal{L}\|}\right)^2} \tag{39}$$

*where $\lambda_1 \geq \lambda_2 ... \geq \lambda_N$ denote the eigenvalues of $\nabla^2_{\boldsymbol{w}}\mathcal{L}$ around the minima.*

*Proof.* For sake of brevity, we assume that the bias in gradient estimates is 0 and focus only on the effect of variance. We start the proof by recalling the Taylor series expansion from Eq. (11). It is clear that while following the true gradient:

$$\left[\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)]\right]_{(bias=0,var=0)} = -\|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2 \Delta t + \frac{1}{2} \left\langle \nabla_{\boldsymbol{w}}\mathcal{L}, \nabla^2_{\boldsymbol{w}}\mathcal{L} \nabla_{\boldsymbol{w}}\mathcal{L} \right\rangle \Delta t^2 \tag{40}$$

Using Lemma A.1 and Eq. (40), we can write the expected decrease in $\mathcal{L}$ in one weight update step (with 0 bias and $\sigma^2$ variance) as:

$$\left[\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)]\right]_{(0,\sigma^2)} = -\|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2 \Delta t + \frac{1}{2} \left( \left\langle \nabla_{\boldsymbol{w}}\mathcal{L}, \nabla^2_{\boldsymbol{w}}\mathcal{L} \nabla_{\boldsymbol{w}}\mathcal{L} \right\rangle + \frac{\sigma^2}{N} \mathrm{Tr}[\nabla^2_{\boldsymbol{w}}\mathcal{L}] \right) \Delta t^2 \tag{41}$$

We note the following inequalities from linear algebra which will be useful in proving the result:

$$\begin{aligned} \lambda_N \|\nabla_{\boldsymbol{w}} \mathcal{L}\|^2 \leq \left\langle \nabla_{\boldsymbol{w}} \mathcal{L}, \nabla^2_{\boldsymbol{w}} \mathcal{L} \nabla_{\boldsymbol{w}} \mathcal{L} \right\rangle \leq \lambda_1 \|\nabla_{\boldsymbol{w}} \mathcal{L}\|^2 \\ \lambda_N \leq \quad \frac{1}{N} \operatorname{Tr}[\nabla^2_{\boldsymbol{w}} \mathcal{L}] \quad \leq \lambda_1 \end{aligned} \tag{42}$$

Using the relations in ineq. 42, we can lower bound the change in $\mathcal{L}$ from Eq. (41):

$$\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)] \geq -\|\nabla_{\boldsymbol{w}} \mathcal{L}\|^2 \Delta t + \frac{1}{2} \left( \lambda_N \|\nabla_{\boldsymbol{w}} \mathcal{L}\|^2 + \sigma^2 \lambda_N \right) \Delta t^2 \tag{43}$$

In order to understand the conditions under which a weight update step causes an increase in $\mathcal{L}$, we focus on the conditions when $\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)] \geq 0$. As such, if the RHS of inequality 43 is greater than 0, then the weight update step necessarily implies $\mathcal{L}[\boldsymbol{w}(t+\Delta t)] \geq \mathcal{L}[\boldsymbol{w}(t)]$. Therefore,

$$\begin{aligned} -\|\nabla_{\boldsymbol{w}} \mathcal{L}\|^2 \Delta t + \frac{1}{2} \left( \lambda_N \|\nabla_{\boldsymbol{w}} \mathcal{L}\|^2 + \sigma^2 \lambda_N \right) \Delta t^2 \geq 0 \\ \lambda_N \geq \frac{2}{\Delta t} \frac{1}{1 + \left( \frac{\sigma}{\|\nabla_{\boldsymbol{w}} \mathcal{L}\|} \right)^2} \end{aligned} \tag{44}$$

Similarly, we can upper bound $\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)]$ and set the upper bound to be less than or equal to 0 in order to study the sufficient conditions under which the weight update step leads to decrease in the loss function.

$$\begin{aligned} \mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)] \leq -\|\nabla_{\boldsymbol{w}} \mathcal{L}\|^2 \Delta t + \frac{1}{2} \left( \lambda_1 \|\nabla_{\boldsymbol{w}} \mathcal{L}\|^2 + \sigma^2 \lambda_1 \right) \Delta t^2 \leq 0 \\ \implies \lambda_1 \leq \frac{2}{\Delta t} \frac{1}{1 + \left( \frac{\sigma}{\|\nabla_{\boldsymbol{w}} \mathcal{L}\|} \right)^2} \end{aligned} \tag{45}$$

□

**Theorem A.8.** ***Bias prevents descent into local minima dependent on Hessian spectrum.*** *Bias in gradient estimates could prevent converging to a minima. Specifically, the sufficient condition for the weight updates to produce a decrease in $\mathcal{L}$ when using a biased estimate of the gradient is:*

$$\frac{b}{\|\nabla_{\boldsymbol{w}} \mathcal{L}\|} \leq \frac{1}{\sqrt{N}} \frac{2}{1 + \psi \Delta t + \sqrt{1 + 4\psi \Delta t + \psi^2 \Delta t^2}} \approx \frac{1}{\sqrt{N}} \frac{1}{(1 + \frac{3}{2} \psi \Delta t)} \tag{46}$$

*where $\psi^2 = \sum_i \lambda_i^2$, i.e. the Frobenius norm of the loss Hessian, and $\lambda_1 \geq \lambda_2 ... \geq \lambda_N$ denote the eigenvalues of $\nabla^2_{\boldsymbol{w}} \mathcal{L}$ around the minima and $\nabla^2_{\boldsymbol{w}} \mathcal{L}$ is a normal matrix.*

*Proof.* For sake of brevity, we assume that the noise in gradient estimates is 0 and focus only on the effect of bias. We start the proof by combining Lemma A.1 and Eq. (40) to get the expected weight decrease in $\mathcal{L}$ in one weight update step (with bias=b and no variance):

$$\begin{aligned} \mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)] = -\|\nabla_{\boldsymbol{w}} \mathcal{L}\|^2 \Delta t + \frac{1}{2} \left\langle \nabla_{\boldsymbol{w}} \mathcal{L}, \nabla^2_{\boldsymbol{w}} \mathcal{L} \nabla_{\boldsymbol{w}} \mathcal{L} \right\rangle \Delta t^2 + b \left\langle \nabla_{\boldsymbol{w}} \mathcal{L}, \overrightarrow{\boldsymbol{\beta}} \right\rangle \Delta t \\ + \frac{1}{2} b^2 \left\langle \overrightarrow{\boldsymbol{\beta}}, \nabla^2_{\boldsymbol{w}} \mathcal{L} \overrightarrow{\boldsymbol{\beta}} \right\rangle \Delta t^2 - \frac{1}{2} b \left\langle \nabla_{\boldsymbol{w}} \mathcal{L}, (\nabla^2_{\boldsymbol{w}} \mathcal{L} + \nabla^2_{\boldsymbol{w}} \mathcal{L}^T) \overrightarrow{\boldsymbol{\beta}} \right\rangle \Delta t^2 \end{aligned} \tag{47}$$

Furthermore, we denote the eigenvalues of the Loss Hessian, i.e. $\nabla^2_{\boldsymbol{w}} \mathcal{L}$, around the minima as $\lambda_1 \geq \lambda_2 ... \geq \lambda_N$ and the corresponding eigenvectors as $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2 ... \boldsymbol{\nu}_N$. Assuming $\nabla^2_{\boldsymbol{w}} \mathcal{L}$ is normal, we can express $\nabla_{\boldsymbol{w}} \mathcal{L}$ and $\overrightarrow{\boldsymbol{\beta}}$ in the basis space of $\{\boldsymbol{\nu}_1, \boldsymbol{\nu}_2 ... \boldsymbol{\nu}_N\}$, i.e. $\nabla_{\boldsymbol{w}} \mathcal{L} = \sum_i \alpha_i \boldsymbol{\nu}_i$ and

$\overrightarrow{\boldsymbol{\beta}} = \sum_i \beta_i \boldsymbol{\nu}_i$. Using standard results from linear algebra, we can write the following expressions:

$$\begin{aligned}
\left\langle \nabla_{\boldsymbol{w}} \mathcal{L}, \nabla^2_{\boldsymbol{w}} \mathcal{L} \nabla_{\boldsymbol{w}} \mathcal{L} \right\rangle &= \sum_i \lambda_i \alpha_i^2 \\
\left\langle \nabla_{\boldsymbol{w}} \mathcal{L}, \overrightarrow{\boldsymbol{\beta}} \right\rangle &= \sum_i \alpha_i \beta_i \\
\left\langle \overrightarrow{\boldsymbol{\beta}}, \nabla^2_{\boldsymbol{w}} \mathcal{L} \overrightarrow{\boldsymbol{\beta}} \right\rangle &= \sum_i \lambda_i \beta_i^2 \\
\left\langle \nabla_{\boldsymbol{w}} \mathcal{L}, (\nabla^2_{\boldsymbol{w}} \mathcal{L} + \nabla^2_{\boldsymbol{w}} \mathcal{L}^T) \overrightarrow{\boldsymbol{\beta}} \right\rangle &= \left\langle \nabla_{\boldsymbol{w}} \mathcal{L}, \nabla^2_{\boldsymbol{w}} \mathcal{L} \overrightarrow{\boldsymbol{\beta}} \right\rangle + \left\langle \nabla_{\boldsymbol{w}} \mathcal{L}, \nabla^2_{\boldsymbol{w}} \mathcal{L}^T \overrightarrow{\boldsymbol{\beta}} \right\rangle \\
&= \left\langle \nabla_{\boldsymbol{w}} \mathcal{L}, \nabla^2_{\boldsymbol{w}} \mathcal{L} \overrightarrow{\boldsymbol{\beta}} \right\rangle + \nabla_{\boldsymbol{w}} \mathcal{L}^T \nabla^2_{\boldsymbol{w}} \mathcal{L}^T \overrightarrow{\boldsymbol{\beta}} \\
&= \left\langle \nabla_{\boldsymbol{w}} \mathcal{L}, \nabla^2_{\boldsymbol{w}} \mathcal{L} \overrightarrow{\boldsymbol{\beta}} \right\rangle + \left\langle \nabla^2_{\boldsymbol{w}} \mathcal{L} \nabla_{\boldsymbol{w}} \mathcal{L}, \overrightarrow{\boldsymbol{\beta}} \right\rangle \\
&= 2 \sum_i \lambda_i \alpha_i \beta_i
\end{aligned} \tag{48}$$

Plugging in the relations from Eq. (48) into Eq. (47), we get:

$$\begin{aligned}
\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)] &= -\|\nabla_{\boldsymbol{w}} \mathcal{L}\|^2 \Delta t + \frac{1}{2} \sum_i \lambda_i \alpha_i^2 \Delta t^2 + b \sum_i \alpha_i \beta_i \Delta t \\
&\quad + \frac{1}{2} b^2 \sum_i \lambda_i \beta_i^2 \Delta t^2 - b \sum_i \lambda_i \alpha_i \beta_i \Delta t^2 \\
&= -\|\nabla_{\boldsymbol{w}} \mathcal{L}\|^2 \Delta t + \frac{\Delta t}{2} \sum_i \left( \lambda_i \alpha_i^2 \Delta t + 2 b \alpha_i \beta_i (1 - \lambda_i \Delta t) + \lambda_i b^2 \Delta t \beta_i^2 \right)
\end{aligned} \tag{49}$$

We can factorize the quadratic in the second summation term as follows:

$$\lambda_i \alpha_i^2 \Delta t + 2 b \alpha_i \beta_i (1 - \lambda_i \Delta t) + \lambda_i b^2 \Delta t \beta_i^2 = \alpha_i^2 \left( \lambda_i \Delta t + 2 \gamma_i (1 - \lambda_i \Delta t) + \lambda_i \Delta t \gamma_i^2 \right)$$

$$\text{where,} \quad \gamma_i = \frac{b \beta_i}{\alpha_i}$$

$$\text{The roots of the quadratic are:} \quad \frac{-(1 - \lambda_i \Delta t) \pm \sqrt{1 - 2 \lambda_i \Delta t}}{\lambda_i \Delta t}$$

Assuming small lr, i.e. $\lambda_i \Delta t << 1$ and using the relation: $\quad (1+x)^n \approx 1 + nx \quad$ when $|x| << 1$

$$\text{The roots are:} \quad \approx \frac{-(1 - \lambda_i \Delta t) \pm (1 - \frac{1}{2} 2 \lambda_i \Delta t)}{\lambda_i \Delta t}$$

$$\text{Therefore, the roots are:} \quad 0, \quad -2 \frac{1 - \lambda_i \Delta t}{\lambda_i \Delta t}$$

$$\begin{aligned}
\text{So,} \quad \lambda_i \alpha_i^2 \Delta t + 2 b \alpha_i \beta_i (1 - \lambda_i \Delta t) + \lambda_i b^2 \Delta t \beta_i^2 &= \alpha_i^2 \lambda_i \Delta t \gamma_i \left( \gamma_i + 2 \frac{1 - \lambda_i \Delta t}{\lambda_i \Delta t} \right) \\
&= b \beta_i (b \beta_i \lambda_i \Delta t + 2 \alpha_i (1 - \lambda_i \Delta t))
\end{aligned} \tag{50}$$

We can further impose the dot product inequality to bound the summation term in Eq. (49):

$$\begin{aligned}
\sum_i \lambda_i \alpha_i^2 \Delta t + 2 b \alpha_i \beta_i (1 - \lambda_i \Delta t) + \lambda_i b^2 \Delta t \beta_i^2 &= \sum_i \left( b \beta_i (b \beta_i \lambda_i \Delta t + 2 \alpha_i (1 - \lambda_i \Delta t)) \right) \\
&\leq \sqrt{\sum_i b^2 \beta_i^2} \sqrt{\sum_i (b \beta_i \lambda_i \Delta t + 2 \alpha_i (1 - \lambda_i \Delta t))^2}
\end{aligned}$$

Using the norm relation: $\quad \| \overrightarrow{\boldsymbol{\beta}} \|^2 = \sum_i \beta_i^2 = N,$

$$\sum_i \lambda_i \alpha_i^2 \Delta t + 2 b \alpha_i \beta_i (1 - \lambda_i \Delta t) + \lambda_i b^2 \Delta t \beta_i^2 \leq b \sqrt{N} \| b \Delta t \zeta_1 + 2 \zeta_2 - 2 \Delta t \zeta_3 \| \tag{51}$$

where $\zeta_1 = [\lambda_1\beta_1 \quad \lambda_2\beta_2 \quad ... \quad \lambda_i\beta_i \quad ...]^T$, $\zeta_2 = [\alpha_1 \quad \alpha_2 \quad ... \quad \alpha_i \quad ...]^T$ and $\zeta_3 = [\lambda_1\alpha_1 \quad \lambda_2\alpha_2 \quad ... \quad \lambda_i\alpha_i \quad ...]^T$. We can bound the norms of each of these vectors to further bound the RHS of inequality 51:

$$\begin{aligned}
\|\zeta_1\| &= \sqrt{\sum_i \lambda_i^2\beta_i^2} \leq \sqrt{\sum_i \lambda_i^2}\sqrt{\sum_i \beta_i^2} = \psi\sqrt{N} \\
\|\zeta_2\| &= \sqrt{\sum_i \alpha_i^2} = \|\nabla_{\boldsymbol{w}}\mathcal{L}\| \\
\|\zeta_3\| &= \sqrt{\sum_i \lambda_i^2\alpha_i^2} \leq \sqrt{\sum_i \lambda_i^2}\sqrt{\sum_i \alpha_i^2} = \psi\|\nabla_{\boldsymbol{w}}\mathcal{L}\| \qquad (52)
\end{aligned}$$

where $\psi = \sqrt{\sum_i \lambda_i^2} = \|\nabla^2_{\boldsymbol{w}}\mathcal{L}\|_F$, i.e. $\psi$ is the Frobenius norm of the loss Hessian. Applying the triangle inequality and ineq. 52, we can upper bound the expression in inequality 51:

$$\begin{aligned}
\|b\Delta t\zeta_1 + 2\zeta_2 - 2\Delta t\zeta_3\| &\leq b\Delta t\|\zeta_1\| + 2\|\zeta_2\| + 2\Delta t\|\zeta_3\| \\
&\leq b\Delta t\psi\sqrt{N} + 2\|\nabla_{\boldsymbol{w}}\mathcal{L}\| + 2\Delta t\psi\|\nabla_{\boldsymbol{w}}\mathcal{L}\| \\
\sum_i(\lambda_i\alpha_i^2\Delta t + 2b\alpha_i\beta_i(1-\lambda_i\Delta t) + \lambda_i b^2\Delta t\beta_i^2) &\leq b\sqrt{N}(b\psi\Delta t\sqrt{N} + 2\|\nabla_{\boldsymbol{w}}\mathcal{L}\| + 2\psi\Delta t\|\nabla_{\boldsymbol{w}}\mathcal{L}\|) \\
\sum_i(\lambda_i\alpha_i^2\Delta t + 2b\alpha_i\beta_i(1-\lambda_i\Delta t) + \lambda_i b^2\Delta t\beta_i^2) &\leq b^2\psi\Delta tN + 2b\sqrt{N}\|\nabla_{\boldsymbol{w}}\mathcal{L}\|(1+\psi\Delta t) \qquad (53)
\end{aligned}$$

Plugging inequality 53 into Eq. (49), we can upper bound the change in $\mathcal{L}$.

$$\begin{aligned}
&\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)] \leq -\|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2\Delta t + \frac{\Delta t}{2}(b^2\psi\Delta tN + 2b\sqrt{N}\|\nabla_{\boldsymbol{w}}\mathcal{L}\|(1+\psi\Delta t)) \\
\implies &\mathcal{L}[\boldsymbol{w}(t+\Delta t)] - \mathcal{L}[\boldsymbol{w}(t)] \leq -\|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2\Delta t + \|\nabla_{\boldsymbol{w}}\mathcal{L}\|b\sqrt{N}\Delta t(1+\psi\Delta t) + \frac{b^2N\psi\Delta t^2}{2} \qquad (54)
\end{aligned}$$

To derive the conditions when $\mathcal{L}$ will *necessarily* decrease upon one step of weight update, we can set the above upper limit to be $\leq 0$, implying that $\mathcal{L}[w(t+\Delta t] \leq \mathcal{L}[\boldsymbol{w}(t)]$. Therefore,

$$\begin{aligned}
&-\|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2\Delta t + \|\nabla_{\boldsymbol{w}}\mathcal{L}\|b\sqrt{N}\Delta t(1+\psi\Delta t) + \frac{b^2N\psi\Delta t^2}{2} \leq 0 \\
\implies &Q(\|\nabla_{\boldsymbol{w}}\mathcal{L}\|) = \|\nabla_{\boldsymbol{w}}\mathcal{L}\|^2\Delta t - \|\nabla_{\boldsymbol{w}}\mathcal{L}\|b\sqrt{N}\Delta t(1+\psi\Delta t) - \frac{b^2N\psi\Delta t^2}{2} \geq 0 \qquad (55)
\end{aligned}$$

It can be clearly seen that $Q(0) < 0$ and $Q$ is an upward facing parabola. Therefore, one of the roots are negative and the other positive. Since $\|\nabla_{\boldsymbol{w}}\mathcal{L}\|$ cannot be negative, the above condition can only be satisfied iff:

$$\begin{aligned}
\|\nabla_{\boldsymbol{w}}\mathcal{L}\| &\geq \frac{b\sqrt{N}\Delta t(1+\psi\Delta t) + \sqrt{b^2N\Delta t^2(1+\psi\Delta t)^2 + 2b^2N\psi\Delta t^3}}{2\Delta t} \\
\|\nabla_{\boldsymbol{w}}\mathcal{L}\| &\geq b\sqrt{N}\frac{1+\psi\Delta t + \sqrt{1+4\psi\Delta t + \psi^2\Delta t^2}}{2}
\end{aligned}$$

Again, using the relation: $(1+x)^n \approx 1+nx \quad$ when $|x| << 1$

$$\sqrt{1+4\psi\Delta t+\psi^2\Delta t^2} \approx 1 + \frac{1}{2}(4\psi\Delta t + \psi^2\Delta t^2) \approx 1 + 2\psi\Delta t$$

$$\begin{aligned}
\text{Therefore,} \quad &\|\nabla_{\boldsymbol{w}}\mathcal{L}\| \geq b\sqrt{N}(1+\frac{3}{2}\psi\Delta t) \\
\implies &\frac{b}{\|\nabla_{\boldsymbol{w}}\mathcal{L}\|} \leq \frac{1}{\sqrt{N}}\frac{1}{(1+\frac{3}{2}\psi\Delta t)} \qquad (56)
\end{aligned}$$

□

**Tightness of Bound:** One of the key inequalities used for proving Theorem A.8 is the dot product inequality. It is well known that the dot product inequality is weaker for high dimensions, specifically

because two random vectors in high-dimensional space are more likely to be orthogonal, i.e. their dot product is $\approx 0$. This is also reflected in Fig. 8 where as the number of dimensions increases, the bound in Theorem A.8 becomes a weaker. Therefore, even though we have a sufficient condition that produces a decrease in $\mathcal{L}$, it may not accurately reflect the conditions under which $\mathcal{L}$ can decrease almost always. To derive these conditions and derive a more stronger bound, we turn to the probably approximately correct framework and concentration inequalities. The concentration inequality for dot product in high dimensions states that for unit vectors $\hat{X}, \hat{Y}$ in N-dimensional space, $\left\langle \hat{X}, \hat{Y} \right\rangle$ concentrates around $\frac{1}{\sqrt{N}}$. Therefore, we can assume that with very high probability, $\left\langle \hat{X}, \hat{Y} \right\rangle \leq \frac{\sqrt[4]{N}}{\sqrt{N}} = \frac{1}{\sqrt[4]{N}}$. Incorporating this into inequality 51:

$$\begin{aligned}\sum_i \lambda_i \alpha_i^2 \Delta t + 2b\alpha_i\beta_i(1-\lambda_i\Delta t) + \lambda_i b^2 \Delta t \beta_i^2 &= \sum_i \left(b\beta_i(b\beta_i\lambda_i\Delta t + 2\alpha_i(1-\lambda_i\Delta t))\right) \\ &\leq \frac{1}{\sqrt[4]{N}} \sqrt{\sum_i b^2\beta_i^2} \sqrt{\sum_i (b\beta_i\lambda_i\Delta t + 2\alpha_i(1-\lambda_i\Delta t))^2}\end{aligned} \tag{57}$$

Following the rest of the proof as above, we can add a correction factor of $\sqrt[4]{N}$ in the final inequality. Therefore,

$$\|\nabla_{\boldsymbol{w}}\mathcal{L}\| \geq b\sqrt[4]{N}\left(1 + \left(1 + \frac{\sqrt[4]{N}}{2}\right)\psi\Delta t\right) \tag{58}$$

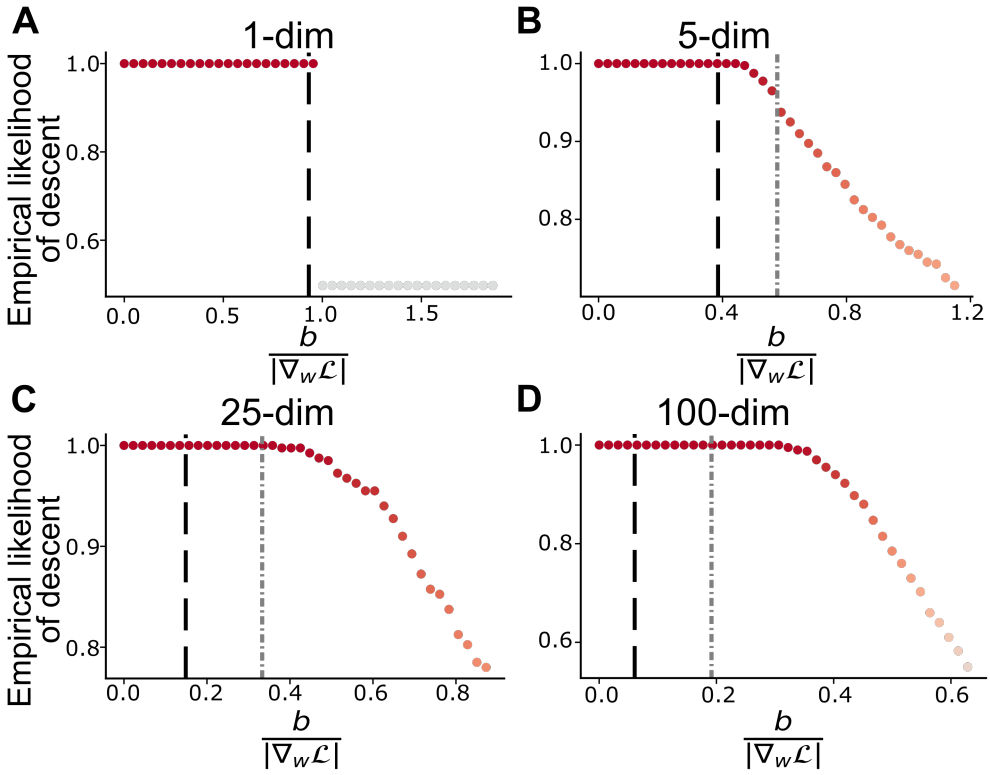


Figure 8: Empiricial verification of Theorem A.8 for increasing dimensionality of parameter space. Black dashed line indicates the bound in Theorem A.8 and Gray dashed dotted line indicates the probably approximately correct bound that offers a stronger condition for higher dimensions. (A) 1 dimensional parameter space; (B) 5-dimensional parameter space where the probably approximate bound satisfies the loss decrease condition with $\approx 95\%$ probability; (C) 25 dimensional parameter space where the black line is a weak bound; (D) 100 dimensional parameter space where the gray line is a stronger bound for decrease in $\mathcal{L}$.

## B EXPERIMENTAL DETAILS

All experiments were implemented using PyTorch 1.10.2 (license: `https://github.com/pytorch/pytorch/blob/master/LICENSE`). For all experiments, the direction of bias was chosen to be along the direction of vector $\vec{\mathbf{1}}$ for simplicity. However, this choice is not a necessity for our experimental results and merely a design choice. Note that our theoretical results are generic enough for any bias direction. Furthermore, for Figures 3-7, torch DoubleTensor was used, which allowed for float64 level of precision (resolution $\approx$ 2.22e-16). Implementation details pertaining to each figure are presented below

### B.1 FIGURE 1

A VGG-16 (initialized with ImageNet-pretrained weights) was trained to perform object recognition on the CIFAR-10 dataset using the Negative Log Likelihood loss function. Each update entailed computing the gradient over the entire dataset, instead of using the more commonly used stochastic gradient descent. This procedure is more commonly known as the full-batch gradient descent. Some amount of bias and variance was added artificially to the computed gradient before making updates. The amount of variance and bias were chosen to be a function of the gradient norm, such that at each step the ratio of the net variance and/or bias to the gradient norm was constant. We plot the performance degradation/improvement corresponding to this fixed variance and bias ratio in the colorplot. Note that we performed a sweep over different variance and bias ratios in order to generate the 2d colorplot. The standard SGD optimizer from PyTorch was used and the optimizer step was called at the end of each epoch to use the full-batch gradient for updates. Hyperparameter values for training are as follows: epochs = 50, learning rate = 0.02, weight decay = 0.0, momentum = 0.9, batch size = 5000. Each bias and variance configuration was run for 3 seeds and accuracy values were averaged over these 3 seeds for plotting. The training was run on an RTX8000 gpu and accelerated using the ffcv library (Github repo: `https://github.com/libffcv/ffcv`, License: `https://github.com/libffcv/ffcv/blob/main/LICENSE`) and each run (1 seed) took approximately 10 minutes.

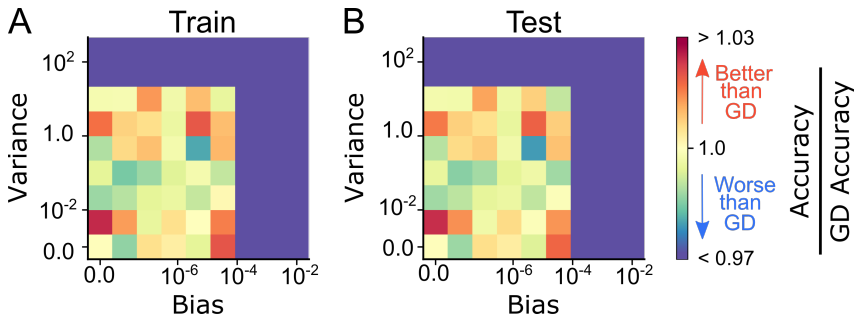


Figure 9: Figure 1 from main text plotted in Spectral colormap. Train and test accuracy of a VGG-16 network trained for 50 epochs (to convergence) on CIFAR-10 using full-batch gradient descent (with no learning rate schedule) with varying amount of variance and bias (as a fraction of the gradient norm) added to the gradient estimates. These results (avg of 20 seeds) indicate that excessive noise and bias harms learning, but a small amount can aid it.

### B.2 FIGURE 3

We ran multiple repeats to sample different noise vectors, typically 20. However, to ensure that the mean of the sampled noise vectors was 0, we artificially sampled an additional noise vector such that the sum of all sampled vectors was 0. This procedure ensured that the bias in gradient estimates was 0 and we could observe the impact of variance, independent of the bias. We observed that this is a feature of finite sampling and if we increase the number of noise vector samples, the sample mean reduces. However, in line with time and compute budget we added this artificial sampling technique.

(A) The teacher network was set to be a single linear layer mapping the input to output. For the student networks with varying width in the hidden layer, the first layer projecting the input to the

latent space is kept fixed and initialized such that the variance of the input is preserved. The weights from the hidden units to output are learned using gradient descent. Decreasing the width below the input dimension also proportionally reduces the variance and therefore trace of the hessian reduces. This results in no change in the $\Delta\mathcal{L}$ when width is less than the teacher width.

(B,C) The teacher network was set to have 4 hidden layers with ReLU non-linearity. The student networks were allowed to have varying depth and width in each layer. The width of all layers was set to the same value. For each configuration, a linear and a ReLU student network was run.

All these experiments were ran on CPU and took a couple of hours. For each configuration, 20 different teacher initializations and 20 different student initializations were used, thereby indicating results averaged over different datasets and ANN initialzations.

### B.3 FIGURE 4

Similar to Figure 3A, these experiments were ran on a linear teacher and student network to validate the relationship between $\Delta\mathcal{L}$ and bias. For each experiment some bias value was set and the bias vector was artificially added to the gradient vector. Each experiment entailed setting the teacher weights to a random configuration and using different such configurations, typically 20. For each teacher configuration, we ran multiple seeds, typically 20, to simulate multiple student network initializations. For each seed, we trained a control network for 10 epochs in order to efficiently sample parameter configurations exhibiting wide range of gradient norm. All these experiments were ran on CPU and took around an hour.

### B.4 FIGURE 5

The teacher network was a VGG-19 configuration trained on the CIFAR-10 dataset to achieve 92.6% accuracy.

(A) For each student network configuration, ImageNet-pretrained and random initialization weights were used. The last layer was re-initialized to have 10 output units as opposed to the typical 1000 units. Each experiment entailed training the student network to match the teacher outputs for 10 different seeds, i.e. student initializations. For each seed, the student was trained for 5 epochs to sample a wide range of gradient norm. For each update step, 20 random vectors were sampled to add noise to gradient. The norm of the vectors was chosen such that the variance in gradient estimates is 20. Similar to Figure 3, the artificial sampling correction was used to ensure an unbiased gradient estimate.

(B) Downsampling indicates reducing the width of network by the downsampling factor. The number of channels in the convolutional layers was reduced by the downsampling factor and the number of units in the linear layers (except the final output layer) were reduced by the downsampling factor. Since these are alterations to the VGG configurations, no ImageNet pretrained weights were available. Hence, only random initializations were used. Same procedure as above was used for each experiment.

Each experiment was run on an RTX8000 GPU and required 10 GB RAM. One experiment took around an hour and multiple experiments were run in parallel on the institute cluster. Note that there was no observed relationship in $\Delta\mathcal{L}$ and gradient norm in any of the experiments.

### B.5 FIGURE 6

Same procedure as above but did not require random sampling of noise vectors. Instead, a constant vector along $\vec{\mathbf{1}}$ was added to the gradient such that the amount of bias in gradient estimates is $b = 0.02$.

### B.6 FIGURE 7

All experiments were run on a linear network setting. For each experiment, the Hessian of the (MSE) loss function was changed by altering the covariance matrix of the input. To compute the empirical likelihood of descent, we sampled different student network initializations (here 100) and a

weight update step was performed with respective variance or bias values. For each such update, we counted the number of updates that led to a decrease in the loss and finally divided this number by the total number of updates performed in the experiment, thereby yielding the empirical likelihood of descending the loss landscape. We also ran this process over multiple teacher network initializations (here 10). It is worth noting that for the case of noisy gradients, we followed similar suit as described in experimental procedure for Fig. 3 to ensure that the amount of noise added was zero-mean and the mean decrease in loss was measured.

(A,B) Similar noise vector sampling procedure was employed and the amount of noise to gradient norm ratio was fixed for all experiments. These experiments followed similar suit as Figure 3.

(C) Similar procedure as Figure 4. Although, here the bias to gradient norm ratio was fixed for all experiments.

These experiments were performed on CPU and took less than an hour.

# C Detailed Discussion on Related works

## C.1 Relation to methods and findings from Raman *et al.*

Our study is significantly motivated and influenced by the setup used by Raman et al. (2019). Despite these strong links, there are certain distinctions in our approach that we elaborate on in this section. In their setup, Raman *et al.* assume the weight update step can be decomposed into a term aligned with the gradient, a term orthogonal to the gradient, and synapse-intrinsic noise. They proceed to derive the optimal network size, specifically width, beyond which learning on the training set is impeded. Instead, in our work, we analyze the role of different architectural choices, like depth, width, and nonlinearity, in mitigating the impact of variance and bias in gradient approximations on both training and generalization performance. Notably, our setting can be used to understand both the impact of intrinsic noise (see extensions to Theorem 2.2) as well as gradient approximation noise on the network performance.

## C.2 Relationship to stochastic gradient descent and dropout

Deep learning pipelines rarely use the true gradient of the loss function over the entire dataset to make weight updates. Instead, it is common practice to add noise in different forms, e.g. mini-batch stochasticity (Bottou, 2012), dropout (Srivastava et al., 2014), dropconnect (Wan et al., 2013) and label noise (Blanc et al., 2020; HaoChen et al., 2021; Damian et al., 2021). Although the noise structure may differ from assumptions in our work, our analytical results can be extended to incorporate the specifics in each case and understanding its impact on learning dynamics.

For stochastic gradient descent (SGD), the noise in gradient (under small learning rate) is unbiased. Although SGD does not exhibit white noise structure, several analytical and empirical studies have used white noise structure to model SGD dynamics, and concluded that the SGD noise acts as a regularizer against converging to sharp minima (Foret et al., 2020; Chaudhari et al., 2019; Yao et al., 2018; Ghorbani et al., 2019; Smith et al., 2021). Furthermore, it has been shown that anti-correlated noise injection improves generalization (Orvieto et al., 2022). Moreover, some studies have leveraged Langevin dynamics to understand the learning trajectory under noisy weight updates (Murray & Edwards, 1992; Rögnvaldsson, 1994). Taken together, these results are in agreement with our analysis of variance in gradient estimates and therefore indicate that our work has far reaching consequences in understanding gradient-based learning. Similarly, dropout can be thought of as adding noise to the credit assignment pathway. However, dropout involves turning off a random subset of units in the network and each such configuration yields an approximation to the gradient, characterized by some variance and bias with respect to the true gradient. For a truly unbiased estimate of the gradient, one must compute an estimate of the gradient across all configurations, something quite uncommon. Empirical results in the deep neural networks have demonstrated that dropout leads to better generalization, albeit at the cost of slightly worse training performance. This result indicates that our analysis of bias in gradient estimates is also relevant to understanding how different training strategies affect gradient-based learning dynamics.

### C.3 A NOTE ON PRACTICAL APPLICATIONS

We believe that our work is an initial step towards understanding the impact of gradient approximations on learning and generalization and the role of network architecture choice in mitigating such impacts. Notably, we believe that our work could provide guidelines for neuromorphic chip design by suggesting possible network architecture choices as well as choice of nonlinearity when the properties of approximation, i.e. variance and bias in gradient estimates, are characterized. Furthermore, we believe that our work scopes out the knowledge landscape in computational neuroscience for biologically-plausible learning rules and future work could aim to characterize popular learning rules using this framework. Similar characterizations could be carried out in biological agents while they learn to perform a task and therefore understand what regime of variance and bias exists in biological circuits. In doing so, we believe we can better formulate learning rules as well as make testable predictions about learning in the brain.