Vanishing Curvature in Randomly Initialized Deep ReLU Networks

Antonio Orvieto* ETH Zurich
Jonas Kohler* ETH Zurich
Dario Pavllo ETH Zurich
Thomas Hofmann ETH Zurich
Aurelien Lucchi University of Basel

Abstract

Deep ReLU networks are at the basis of many modern neural architectures. Yet, the loss landscape of such networks and its interaction with state-of-the-art optimizers is not fully understood. One of the most crucial aspects is the landscape at random initialization, which often influences convergence speed dramatically. In their seminal works, Xavier & Bengio, 2010 and He et al., 2015 propose an initialization strategy that is supposed to prevent gradients from vanishing. Yet, we identify shortcomings of their expectation analysis as network depth increases, and show that the proposed initialization can actually fail to deliver stable gradient norms. More precisely, by leveraging an in-depth analysis of the median of the forward pass, we first show that, with high probability, vanishing gradients cannot be circumvented when the network width scales with less than $\Omega(\text{depth})$. Second, we extend this analysis to second-order derivatives and show that random i.i.d. initialization also gives rise to Hessian matrices with eigenspectra that vanish in depth. Whenever this happens, optimizers are initialized in a very flat, saddle point-like plateau, which is particularly hard to escape with stochastic gradient descent (SGD) as its escaping time is inversely related to curvature magnitudes. We believe that this observation is crucial for fully understanding (a) the historical difficulties of training deep nets with vanilla SGD and (b) the success of adaptive gradient methods, which naturally adapt to curvature and thus quickly escape flat plateaus.

Proceedings of the 25$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

1 Introduction and related work

In the last decade, network depth has emerged as a key component for the success of modern deep learning [He et al., 2016b], providing significant improvements in terms of generalization, particularly in the field of computer vision [He et al., 2016a, He et al., 2016b] and natural language processing [Brown et al., 2020]. These benefits are mostly attributed to gains in representational power [Telgarsky, 2016]. From an optimization perspective, however, depth introduces several problems when training after random initialization. First, in the infinite depth limit, a collapse in the rank of the network mapping prevents information propagation, which renders learning impossible [Schoenholz et al., 2016, Pennington et al., 2018, Daneshmand et al., 2020]. Secondly, even in finite but large depth the so-called *vanishing gradient* problem commonly makes it difficult to train deep ReLU networks with stochastic gradient descent [Hochreiter, 1991, Bengio et al., 1994, Pascanu et al., 2013] – even when using state-of-the-art initialization schemes such as He initialization [He et al., 2015] (see Figure 4).

A substantial research effort was put into finding architectural "fixes" to mitigate the problems mentioned above – e.g. skip connections [He et al., 2016a] and batch normalization [Ioffe and Szegedy, 2015]. While on the one hand these components are of great practical and intellectual value, on the other hand they make it harder to deliver a solid and detailed mathematical understanding of modern neural networks. In this work, we take a step back from modern architectural components and propose to revisit vanishing gradient phenomenon in deep ReLU MLPs, to enhance our understanding of basic neural network landscapes as depth increases – going beyond the expectation analysis in [He et al., 2016a], and showcasing the interesting and pathologic behavior of very deep networks. To complete the picture, we provide novel results on the Hessian of deep ReLU networks at initialization, and show important implications for gradient-based optimizers and adaptive methods such as RMSprop [Tieleman and Hinton, 2012].

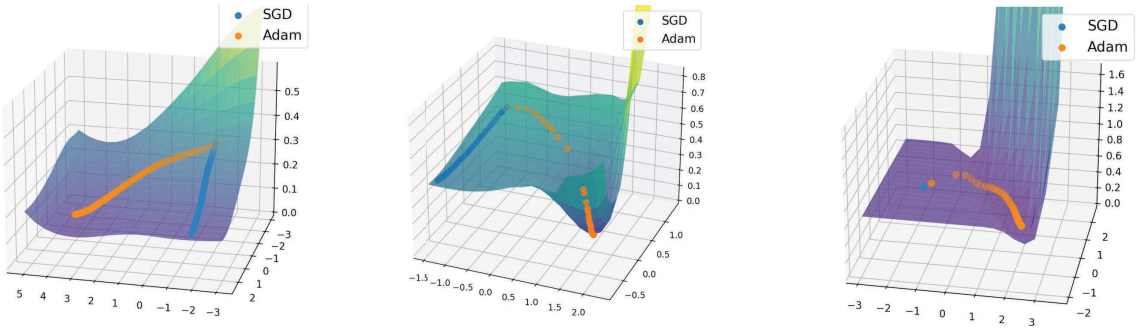


Figure 1: Optimization landscape after projection onto the two principal search directions of SGD and Adam. Fully connected ReLU networks on a binary classification problem (see App. A.1) with *He init*. Networks have 12 hidden units per layer and 1 (left), 32 (middle), 64 (right) hidden layers.

More precisely, we first show that networks that are deeper than wide suffer from *vanishing gradients at initialization even when following the established Xavier or He initialization schemes proposed in* [Glorot and Bengio, 2010, He et al., 2015]. This is somewhat surprising at first, since the analysis undertaken in these seminal works suggests stable gradient norms for any depth, in expectation. However, as we show in Section 3, the expectation analysis has an important shortcoming: when networks are deeper than wide, the initialization variance suggested to be optimal in order to stabilize the expected forward- and backpropagation norms yields exploding higher moments of these quantities. As a result, their distribution becomes fat tailed in depth and hence the expected value itself is increasingly unlikely to be observed. Thus, *we instead study the median of the propagation norms, which we find to indeed vanish in depth.*

Second, we reveal that, in the above settings, not only the gradient vanishes as depth increases, but also the entries of the Hessian become smaller and smaller. Hence, by Gershgorin's theorem [Gershgorin, 1931], the Hessian *eigenvalues shrink in depth.* As a result, random i.i.d. initialization ends up positioning optimization methods in a *very flat, saddle point like region* around the origin, where gradients are small and both negative as well as positive curvature exist. This is particularly *unfortunate for stochastic gradient descent* (SGD) as its saddle escaping time is inversely related to the magnitude of the most negative eigenvalue (see Prop. 7 as well as [Daneshmand et al., 2018, Fang et al., 2019]). This observation complements the vanishing gradient argument, whose implications for optimization were so far rather vague since small gradients on their own are not a-priori problematic for gradient-based learning: In fact, it is often the Lipschitz constant (i.e. curvature) and not the norm of the gradients that determines the speed of convergence of gradient descent [Nesterov et al., 2018]. Even around saddle points, sufficient negative curvature allows stochastic gradient descent to make fast progress despite small gradient norms.[1] Hence, vanishing gradients on their own fall short of a satisfactory description for the difficulty of training very deep networks with plain SGD.

We believe that it is precisely the vanishing curvature phenomenon, combined with the deficiencies of SGD in escaping flat saddles, that lead to developments on three fronts: (i) *modern architectural components*: batch normalization (BN) [Ioffe and Szegedy, 2015] and residual connections [He et al., 2016a], which collectively circumvent vanishing gradients/curvature even in non-linear networks (Fig. 9). (ii) *robust initialization schemes*: orthogonal- [Saxe et al., 2013], which is restricted to *linear* networks (Fig. 15) and Fixup/Skip initialization [Zhang et al., 2019a, De and Smith, 2020], which downscale the parametric branch of residual architectures proportionally to their depth (thus trivially yielding identity mappings in the limit). (iii) *Adaptive training algorithms*: notably several years before the emergence of BN and residual connections, so-called adaptive gradient methods [Duchi et al., 2011, Tieleman and Hinton, 2012] have shown remarkable success in training deep networks. We believe that their increased popularity is (at least partially) attributable to the fact that adaptive gradients methods are robust to this failure mode of random i.i.d. initialization because they naturally adapt to curvature, which allows them to escape saddles in time independent of their flatness (see Section 3 as well as [Dauphin et al., 2015, Staib et al., 2019]).

[1] To be more precise, the anisotropic noise of SGD has been shown to be aligned with negative curvature directions [Zhu et al., 2019, Daneshmand et al., 2018, Fang et al., 2019], which allows for a per-step progress of $\mathcal{O}(|\lambda^3|)$ where λ is the curvature along the current search direction [Curtis and Robinson, 2019].

In summary, our contribution is threefold:

- We discuss shortcomings of the analysis in Xavier [Glorot and Bengio, 2010] and He [He et al., 2015] initialization schemes and clarify when gradients vanish even with the initialization proposed in these seminal works.
- We enhance the understanding of the vanishing gradient phenomenon by showing that they co-occur with vanishing curvature, giving rise to flat plateaus at initialization. This effect gives a comprehensive understanding of the difficulty of training (classical) deep nets with (vanilla) SGD.
- Finally, we link the remarkable curvature adaptation capability of adaptive gradient methods to this phenomenon and show that it allows them to optimize networks of any depth.

2 Notation and setting

In our theoretical analysis, we consider the L2 loss associated with a multilayer perceptron (MLP)

$$\mathcal{L}(\mathbf{W}) = \frac{1}{2n}\sum_{i=1}^{n} \|\mathbf{y}_i - \mathbf{B}\mathbf{D}^L\mathbf{W}_\phi^{L:1}\mathbf{A}\mathbf{x}_i\|_2^2, \qquad \mathbf{W}_\phi^{L:1} := \mathbf{W}^L\mathbf{D}^{L-1}\mathbf{W}^{L-1}\cdots\mathbf{W}^2\mathbf{D}^1\mathbf{W}^1\mathbf{D}^0 \tag{1}$$

where $\mathbf{x}_i \in \mathbb{R}^{d_{in}}, \mathbf{y}_i \in \mathbb{R}^{d_{out}}$, $\mathbf{A} \in \mathbb{R}^{d\times d_{in}}$,$\mathbf{B} \in \mathbb{R}^{d_{out}\times d}$, and $\mathbf{W}^\ell \in \mathbb{R}^{d\times d}, \forall \ell = 1,\dots,L$. $\mathbf{D}^\ell$ is the diagonal matrix of activation gates w.r.t the non-linearity ϕ at layer ℓ, which we consider to be either $\phi(x) = x$ (linear networks) or $\phi(x) = \max\{x, 0\}$ (ReLU networks).

Notably, deep ReLU MLPs are not only the most widely studied networks in deep learning theory [Allen-Zhu et al., 2019, Jacot et al., 2018]), but they are also still of practical relevance for example in neural radiance fields [Mildenhall et al., 2020] and image classification [Tolstikhin et al., 2021, Touvron et al., 2021].

Assumption 1 (Random initialization)**.** *Each entry of $\mathbf{W}^\ell$ ($\ell = 1,\dots,L$) is initialized i.i.d. with some distribution $\mathcal{P}$ symmetric around zero with variance $\sigma^2 < \infty$ and fourth moment $\mu_4 < \infty$.*

For more than a decade, the standard choice of initialization variance was $\sigma^2 = \frac{1}{3d}$ [LeCun et al., 1998]. Motivated by repeated observations of the *vanishing gradient problem*, an improved initialization was suggested by [Glorot and Bengio, 2010] and [He et al., 2015]. The following theorem summarizes the reasoning in [He et al., 2015]. We define the parameter $p = 1$ for the linear case and $p = 1/2$ for the ReLU case

Proposition 2 ([Glorot and Bengio, 2010], [He et al., 2015])**.** *Under Assumption 1, the variance of the weight gradients $Var(\partial\mathcal{L}(\mathbf{W})/\partial\mathbf{W}^\ell)$ scales as $(pd\sigma^2)^L$ across all layers $\ell = 1,\dots,L$. When initializing with $\sigma^2 = \frac{1}{3d}$ (LeCun init.), this quantity vanishes in depth. Instead, choosing $\sigma^2 = 1/d$ in the linear case (Xavier init.), and $\sigma^2 = 2/d$ in the ReLU case (He init), stabilizes the variance.*

Proof. Let $\mathbf{a}^{\ell+1} = \mathbf{W}^\ell\mathbf{h}^\ell$ be the preactivation of layer $\ell+1$, computed using $\mathbf{h}^\ell = \mathbf{D}^\ell\mathbf{a}^\ell$, the activation at layer ℓ . Let $a^{\ell+1}$, w^ℓ and h^ℓ represent the random variables corresponding to each element in $\mathbf{a}^{\ell+1}$, $\mathbf{W}^\ell$ and $\mathbf{h}^\ell$ respectively. Since w^ℓ is zero mean, we have that $\mathrm{Var}[a^{\ell+1}] = d\cdot\mathrm{Var}[w^\ell]\cdot\mathbb{E}[(h^\ell)^2]$. Finally, since $\mathbb{E}[(h^\ell)^2] = p\mathrm{Var}[a^\ell]$, we end up with $\mathrm{Var}[a^{\ell+1}] = d\sigma^2 p\cdot\mathrm{Var}[a^\ell]$, which yields $\mathrm{Var}[a^{\ell+1}] = \mathrm{Var}[a^\ell]$ for $\sigma^2 = \frac{1}{dp}$. □

For example, for the uniform initialization $\mathcal{P} = \mathcal{U}[-\tau,\tau]$, we have $\sigma^2 = \tau^2/3$ and hence the "optimal" initialization range amounts to $\tau = \sqrt{3/d}$ in the linear - and $\tau = \sqrt{6/d}$ in the ReLU case.

3 Vanishing in neural chains and implications for optimization

To illustrate an important shortcoming in the analyses of [Glorot and Bengio, 2010] and [He et al., 2015], we consider a deep linear network of width one (henceforth called *neural chain*). While these networks are utterly useless for practical applications, they are sufficient to exhibit some critical properties of the loss landscape, that generalize to wider nets (see next section).

In the neural chain case, if $\mathbf{A} = \mathbf{B} = 1$, Eq.(1) simplifies to

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{n}(y_i - w_L...w_1x_i)^2/(2n).$$

We consider each $w_i \in \mathbb{R}$ to be drawn uniformly at random in $[-\tau,\tau]$.

Shortcomings of the expectation analysis. Prop. 2 suggests that both forward pass and gradient remain stable in magnitude when choosing $\tau = \sqrt{3}$. While this is true *in expectation*, it is not the case when initializing individual models, where the expected value becomes an increasingly atypical event (see Thm. 6) as the chain grows in depth (L). Indeed, in Fig. 2 we see that all quantities vanish under the "optimal" initialization. Perhaps the most intuitive indication for this pathological behavior comes from writing down the following population quantities for the absolute value of the input-output map.

Proposition 3 (Forward pass statistics chain)**.** *Consider the absolute value of a forward pass on the chain, i.e. the random variable* $v_{\tau,L} := \prod_{k=1}^{L}(\tau w_k)$*, with* $w_k \overset{iid}{\sim} \mathcal{U}(0,1]$*. Then,*

$$\mathbb{E}\left[v_{\tau,L}\right] = \left(\tfrac{\tau}{2}\right)^L, \mathbb{E}[v_{\tau,L}^2] = \left(\tfrac{\tau^2}{3}\right)^L, \mathbb{E}[v_{\tau,L}^3] = \left(\tfrac{\tau^3}{4}\right)^L \tag{2}$$

Clearly, $\tau = \sqrt{3}$ *(Xavier init.) leads to* $\mathbb{E}\left[v_{\tau,L}\right] \to 0$*,* $\mathbb{E}\left[v_{\tau,L}^2\right] = 1$ *and* $\mathbb{E}[v_{\tau,L}^3] \to \infty$*, as* $L \to \infty$*.*

The result follows directly from the moments of the uniform distribution. It might be tempting to conclude from Eq. 2 that picking $\tau = 2$ instead of $\sqrt{3}$ solves the problem. Yet, this is not the case since then $\mathbb{E}[v_{\tau,L}^2] \to \infty$ and by Mallows inequality [Mallows and Richter, 1969] the mean becomes an unreliable predictor for the median, as their difference is bounded by one standard deviation (exploding). In fact, the above proposition reveals that one cannot stabilize any pair of moments of $v_{\tau,L}$ simultaneously and hence in both cases $\tau = \sqrt{3}$ and $\tau = 2$, the distribution of $v_{\tau,L}$ becomes fat-tailed as $L \to \infty$, which leads to slow convergence of the central limit theorem[2]. As we show in Sec. 4, this basic moment trade off prevails in wider nets (see Eq. 5).

Median analysis. The above considerations suggest that one has to go beyond the population analysis in order to better understand this phenomenon. In a first step, we characterize the distribution of the magnitude of the input-output map in log scale.

Lemma 4 (Distribution of chain input-output)**.** *In the setting of Prop. 3,*

$$-\ln(v_{\tau,L}) = z - L\ln(\tau), \quad z \sim \text{Erlang}(L,1).$$

Hence $\Pr(-\ln(v_{\tau,L}) \le \zeta) = 1 - e^{-\xi}\sum_{k=0}^{L-1}\frac{\xi^k}{k!}$*,* $\xi := \zeta + L\ln\tau$*.*

Proof. Basic logarithm identities allow us to write

$$-\ln(v_{\tau,L}) = -\ln\left(\prod_{k=1}^{L}(\tau w_k)\right) = -L\ln(\tau) - \sum_{k=1}^{L}\ln(w_k).$$

Clearly, $-\ln(w_k))$ is exponentially distribution with parameter 1. Furthermore, if random variables $V_k \sim \text{Exp}(\lambda)$ are independent, then $\sum_{k=1}^{L} V_k \sim \text{Erlang}(L,\lambda)$ [Temme, 1996, Devroye, 2006]. The CDF follows from the properties of the Erlang distribution, which concludes the proof. □

[2]The speed of convergence in the CLT, as bounded by the Berry-Esseen inequality, is proportional to $\mathbb{E}[|v|^3]$.

This allows to characterize the median and provide an asymptotic[3] bound on the forward pass norm.

Proposition 5 (Expectation is not predictive for input-output map magnitude)**.** *We have*

$$\text{median}\left[v_{\tau,L}\right] = e^{L\ln(\tau) - \tilde{L}},$$

with $L - 1/3 \le \tilde{L} \le L - 1 + \ln(2)$*. Therefore, if* $\tau = 2$*,* $\text{median}\left[v_{\tau,L}\right] \to 0$ *while* $\mathbb{E}\left[v_{\tau,L}\right] = 1$ *and* $\mathbb{E}\left[v_{\tau,L}^2\right] \to \infty$*. For* $\tau = \sqrt{3}$*, also* $\text{median}\left[v_{\tau,L}\right] \to 0$*. Yet, the median is stabilized for* $\tau = e$*, since*

$$\lim_{L\to\infty}(v_{\tau,L})^{\frac{1}{L}} \overset{a.s.}{=} \tau/e,$$

which implies $v_{\tau,L} \sim \exp\left(-L(1-\ln\tau)\right)$*.*

Proof. The moments follow from Prop. 3. For the median, we solve $\Pr(-\ln v_{\tau,L} \le \zeta) = 1/2$ for ζ, which by Lemma 4 is equivalent to solving $1 - e^{-\xi}\sum_{k=0}^{L-1}\frac{\xi^k}{k!} = 1/2$ w.r.t. $\xi := \zeta + L\ln\tau$. The solution, termed $\tilde{L}$, is approximated with a Ramanujan formula (1913), as in [Choi, 1994]. Since $v_{\tau,L} = e^{L\ln\tau - z}$, then $(v_{\tau,L})^{\frac{1}{L}} = \tau e^{-z/L}$. We conclude with the strong law of large numbers. □

We now apply the idea behind the last result to analyze the first and second order partial derivatives.

Theorem 6 (Almost sure vanishing)**.** *Assume bounded data and* $w_i \sim \mathcal{U}[-\tau,\tau]$*, with fixed* τ*. For each* $k,\ell \le L$ *we asymptotically (as* $L \to \infty$*) have almost surely that*

$$\left|\frac{\partial\mathcal{L}_{chain}(\mathbf{w})}{\partial w_k}\right|, \left|\frac{\partial^2\mathcal{L}_{chain}(\mathbf{w})}{\partial w_k \partial w_{\ell\neq k}}\right| = \mathcal{O}\left(e^{-(L-1)(1-\ln\tau)}\right),$$
$$\left|\frac{\partial^2\mathcal{L}_{chain}(\mathbf{w})}{\partial w_k \partial w_k}\right| = \mathcal{O}\left(e^{-2(L-1)(1-\ln\tau)}\right).$$

In particular, as for $v_{\tau,L}$*, all these quantities asymptotically vanish if* $\tau < e$ *and explode if* $\tau > e$*.*
In the case of Xavier init. $\tau = \sqrt{3}$*, the Hessian vanishes in norm (hence eigenvalues vanish) and becomes hollow, i.e. diagonal elements become exponentially smaller than off-diagonal elements.*

The proof is presented in App. B.1. Furthermore, empirical simulations in Fig. 2 and Fig. 21 (top row) show that the result is very precise.

Implications on landscape and optimization. In narrow networks, our results show vanishing gradients and hollow Hessians with positive and negative eigenvalues of decreasing magnitude (also see Fig.

[3]In the context of asymptotic expansions, we write $f \sim g$ if $\lim_{L\to\infty} f(L)/g(L) = 1$.

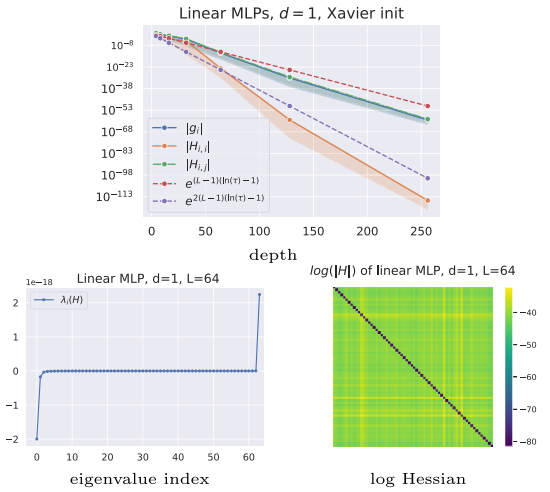


Figure 2: **Top:** Gradient and Hessian entry magnitudes for deep neural chains (Xavier init, Mean and 95% CI of 10 random seeds) **Bottom:** Eigenvalues and log Hessian entry maginute at init. for $L = 64$.

13). Hence, the initialization landscape constitutes a plateau that resembles a barely curved saddle (see Fig. 3). As discussed next, this is particularly bad for optimization with plain SGD but adaptive methods escape the plateau quickly due to a notable curvature adaptation capability.

To illustrate this point, we consider a single datapoint (x, y) with $x, y > 0$ and study the gradient flow on a neural chain of depth L with initialization $0 < w_1(0) = w_2(0) = \cdots = w_L(0) := w_0 \in \mathbb{R}$. The gradient

$$\nabla_{w_i} \mathcal{L}(\mathbf{w}) = x \prod_{j \neq i} w_j \left(\prod_r w_r x - y \right)$$

, is invariant w.r.t. any permutation of the w_i's. Hence, each coordinate of the gradient flow solution will satisfy $w_1(t) = w_2(t) = \cdots = w_L(t) =: w(t)$ and $w(t) \to w^* = (y/x)^{1/L}$ (as $L \to \infty$). The gradient flow is $\dot{w}(t) = -w(t)^{2L-1}x^2 + w(t)^{L-1}xy$. To simplify this we can take the special case $x = y$ (which leads to $w^* = 1$) and, drop the first term (negative). Hence we get an upper bounding solution (since $w(t)$ is increasing) which explodes in finite time t_e (see Fig. 3):

$$w(t) \leq [(L-2)(t_e - t)]^{-\frac{1}{L-2}}, \quad t_e = w_0^{2-L}/(L-2). \tag{3}$$

In this case, the upper bound for $w(t)$ reaches $w^* = 1$ at time $t^* = t_e - \frac{1}{L-2}$, which is exponential in the network depth L. This provides a proof for the following proposition.

Proposition 7 (Slow convergence of Gradient Flow on the chain)**.** *On neural chains, in the worst case, gradient flow takes exponential (in depth) time to reach an ϵ-neighbour of the solution.*

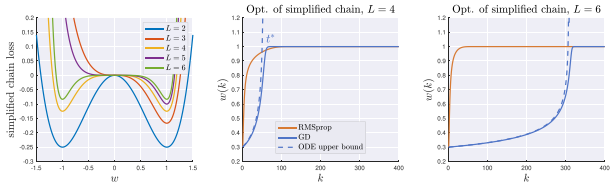


Figure 3: Chain setting of Prop. 7. *Fast convergence of RMSprop* with $\beta_2 = 0.9$ and stepsize decay. For GD, $\eta = 0.1$ is used since *bigger η leads to instability*). Plotted is also the loss corresponding to the integrated gradient $w^{2L-1} + w^{L-1}$. For the discretizing the bound, we use the equivalence $\eta k \equiv t$.

Curvature Adaptation of RMSprop. As can be seen in Fig. 3, RMSprop [Tieleman and Hinton, 2012] is able to optimize the neural chain, in a number of iterations independent of depth. Importantly, this finding is also observable in wider MLPs (Fig. 6) and deep convnets (Fig.7).

To provide some intuition around this phenomenon, we apply RMSprop to the neural chain gradient flow approximation $\dot{w}(t) = w(t)^{L-1}$. This gradient flow approximation is tight during the first steps of the optimizer if L is big. The RMSprop flow solves $\dot{w}(t) = w(t)^{L-1}/\sqrt{v(t)}$, where $v(t)$ is a low-pass filter on the approximate square gradient $w(t)^{2L-2}$. Since $w(t)^{L-1}$ is increasing, $v(t)$ is also increasing and the filter delay guarantees $\sqrt{v(t)} \leq w(t)^{L-1}$. It follows that $\dot{w}(t) > 1$ for t small, regardless of the network depth, which allows RMSprop to quickly escape the flat plateau.

In the vanishing curvature setting predicted by Thm. 6 and confirmed in Fig.2 & 3 (leftmost plot), this speedup is not extremely surprising. As we discuss thoroughly in App. C, many recent works report an improved curvature adaptation of adaptive methods compared to SGD [Dauphin et al., 2015, Kunstner et al., 2019]. For instance, [Staib et al., 2019] recently showed that RMSprop is provably faster than SGD around flat saddle points (see Section 5.2 of their paper). This result, *in combination with our findings on the flatness of the initialization landscape*, gives an explanation for the historical difficulties of training deep nets with SGD and for the success of adaptive methods.[4]

[4]While adaptive gradient methods are the go-to optimizer for transformers (e.g. Bert [Devlin et al., 2018], GPT-3 [Brown et al., 2020] & Vision Transformer [Dosovitskiy et al., 2020]) and MLPs (e.g. NeRF [Mildenhall et al., 2020] & MLP-Mixer [Tolstikhin et al., 2021]), their generalization performance within convnets has been questioned [Wilson et al., 2017]. Yet, particularly RMSProp is becoming increasingly popular again in computervision (e.g. EfficientNet [Tan and Le, 2019], MobileNet [Howard et al., 2019] & FBNet [Dai et al., 2020].)

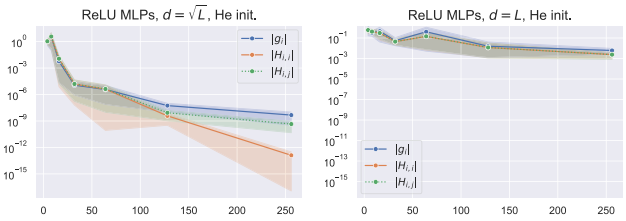


Figure 4: **Effect of width in ReLU MLPs:** Gradient and curvature scaling on Fashion-MNIST over depth. While quantities vanish on the left ($d = \sqrt{L}$), the right shows stable magnitudes. Mean and 95% CI of 15 runs.

Effect of noise. While it is known that the inherent sampling noise of SGD is anisotropic and in many settings aligned with negative curvature [Daneshmand et al., 2018, Zhu et al., 2018, Li et al., 2020], the saddle escape time still depends inversely on the magnitude of the smallest eigenvalue [Daneshmand et al., 2018, Curtis and Robinson, 2019]. As a result, similar to gradient flow on the chain, SGD is unable to train networks with vanishing gradients/curvature despite the presence of inherent noise (see Fig. 6 & 7). Another possibility is to directly add noise to the updates [Du et al., 2017, Du et al., 2019]. In Fig. 23-25, we provide evidence that noise can indeed accelerate GD on the chain, but it is still orders of magnitude slower than RMSprop, for any noise level and stable learning rate.

4 Vanishing in MLPs of arbitrary width

In analogy with Prop. 3, we first note (proof in App. B) the important fact that also in the general MLP case different population quantities cannot be jointly stabilized using standard i.i.d. initialization.

Proposition 8 (Forward pass statistics MLP)**.** *Let $\kappa = \mu_4/\sigma_4$ be the kurtosis (fourth standardized moment). Let $p = 1$ in the linear case and $p = 1/2$ in the ReLU case. Then we have*

$$\begin{aligned} \mathbb{E}\|\mathbf{W}_\phi^{k:1}\mathbf{A}\mathbf{x}\|_2^2 &= (d\sigma^2 p)^k \mathbb{E}\|\mathbf{A}\mathbf{x}\|_2^2. \\ \begin{pmatrix} \mathbb{E}\|\mathbf{W}_\phi^{k:1}\mathbf{z}\|_2^4 \\ \mathbb{E}\|\mathbf{W}_\phi^{k:1}\mathbf{z}\|_4^4 \end{pmatrix} &= \left(p^2 d\sigma^4\right)^k \mathbf{Q}^k \begin{pmatrix} \mathbb{E}\|\mathbf{A}\mathbf{x}\|_2^4 \\ \mathbb{E}\|\mathbf{A}\mathbf{x}\|_4^4 \end{pmatrix}, \\ \mathbf{Q} &:= \begin{pmatrix} d+2 & \frac{\kappa-3+(1-p)(d+2)}{p} \\ 3 & \frac{\kappa-3p}{p} \end{pmatrix}. \end{aligned} \tag{4}$$

For deep linear nets of arbitrary width d and Gaussian initialization ($\kappa = 3$) the above simplifies to

$$\begin{aligned} \mathbb{E}\|\mathbf{W}^{L:1}\mathbf{A}\mathbf{x}\|_2^2 &= (d\sigma^2)^L \mathbb{E}\|\mathbf{A}\mathbf{x}\|_2^2, \\ \mathbb{E}\|\mathbf{W}^{L:1}\mathbf{A}\mathbf{x}\|_2^4 &= \left(d\sigma^4\right)^L (d+2)^L \mathbb{E}\|\mathbf{A}\mathbf{x}\|_2^4. \end{aligned} \tag{5}$$

Hence, as for the neural chain (see Prop. 2), picking the Xavier initialization $\sigma^2 = \frac{1}{d}$ stabilizes $\mathbb{E}\|\mathbf{W}^{L:1}\mathbf{A}\mathbf{x}\|_2^2$, but $\mathbb{E}\|\mathbf{W}^{L:1}\mathbf{A}\mathbf{x}\|_2^4 = \left(\frac{d+2}{d}\right)^L \mathbb{E}\|\mathbf{A}\mathbf{x}\|_2^4$ explodes *unless* d grows faster than L. This points to an important shortcoming of the initialization proposed in [Glorot and Bengio, 2010] & [He et al., 2015], which — as we note next — is only guaranteed to prevent vanishing gradients and curvature in networks that are wider than deep. The next result is verified in Fig. 21 in the appendix.

Theorem 9. *The initialization in [Glorot and Bengio, 2010] & [He et al., 2015] is guaranteed to stabilize both the mean and the median of the squared forward pass norm for $d = \Omega(L)$.*

Proof. We carry out the proof for the Gaussian linear case, but the other settings are conceptually equivalent. First, in the case $\sigma^2 = 1/d$, we have

$$\begin{aligned} \mathrm{Var}\|\mathbf{W}^{L:1}\mathbf{A}\mathbf{x}\|_2^2 &= \mathbb{E}\|\mathbf{W}^{L:1}\mathbf{A}\mathbf{x}\|_2^4 - \left(\mathbb{E}\|\mathbf{W}^{L:1}\mathbf{A}\mathbf{x}\|_2^2\right)^2 \\ &\overset{Eq.\ (5)}{=} \left(\frac{d+2}{d}\right)^L - 1. \end{aligned} \tag{6}$$

By Mallows inequality [Mallows and Richter, 1969], the square root of this quantity provides a upper bound on $\left|\mathrm{median}\left(\|\mathbf{W}^{L:1}\mathbf{A}\mathbf{x}\|_2^2\right) - \mathbb{E}\|\mathbf{W}^{L:1}\mathbf{A}\mathbf{x}\|_2^2\right|$. If we want to guarantee a *non-vanishing median*, say in $[1-\alpha, 1+\alpha]$, for $1 > \alpha > 0$, we need to have d s.t. $\left(\frac{d+2}{d}\right)^L - 1 \leq \alpha^2$, which implies $d \geq \frac{2}{(\alpha^2+1)^{\frac{1}{L}}-1}$. A Maclaurin's series expansion gives $(\alpha^2+1)^{\frac{1}{L}} = 1 + \ln(\alpha^2+1)/L + \mathcal{O}(1/L^2)$, which concludes the proof for large enough L. □

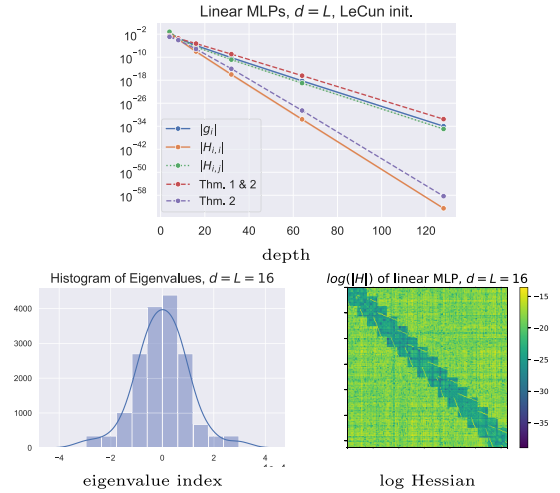


Figure 5: **Top:** Vanishing gradient and Hessian for deep linear MLPs with LeCun init. x-axis depicts depth *and* width. **Bottom**: log abs. Hessian entries and eigenvalues at random init. (see ReLU MLP in Fig. 4)

Before discussing the regime $d \ll L$, we consolidate our core claim about vanishing curvature by generalizing the results of [Glorot and Bengio, 2010, He et al., 2015] to second-order derivatives.

Theorem 10 (Gradient and Hessian in exp.)**.** *Under Ass. 1, the* ***expected*** *norm of any Hessian diag block* $\mathbb{E}[\|\frac{\partial^2\mathcal{L}(\mathbf{W})}{\partial \mathbf{W}_k \partial \mathbf{W}_k}\|_F]$ *in linear nets scales as* $\mathcal{O}\left((d\sigma^2)^L\right)$*, while off-diag. blocks* $\mathbb{E}[\|\frac{\partial^2\mathcal{L}(\mathbf{W})}{\partial \mathbf{W}_k \partial \mathbf{W}_\ell}\|_F]$ *and the gradient* $\mathbb{E}[\|\frac{\partial\mathcal{L}(\mathbf{W})}{\partial \mathbf{W}_k}\|_F]$ *scale as* $\mathcal{O}\left((d\sigma^2)^{\frac{L}{2}}\right)$*. In ReLU nets the scaling amounts to* $\mathcal{O}\left(\left(\frac{d}{2}\sigma^2\right)^L\right)$ *and* $\mathcal{O}\left(\left(\frac{d}{2}\sigma^2\right)^{\frac{L}{2}}\right)$ *respectively.*

This result is (to the best of our knowledge) the first to study the effects of depth on second-order derivatives at random initialization. Its proof, which mainly builds upon Prop. 8, can be found in App. B. A simple application of Gershgorin's theorem yields the following bound on the eigenvalues.

Corollary 11. *Under Assumption 1, the* ***expected*** *magnitude of the largest eigenvalue* $\lambda_{\max}$ *is upper bound as* $\mathbb{E}[|\lambda_{\max}^{linear}|] \leq Ld \cdot \mathcal{O}\left((d\sigma^2)^{\frac{L}{2}}\right)$ *and* $\mathbb{E}[|\lambda_{\max}^{ReLU}|] \leq Ld \cdot \mathcal{O}\left((\frac{d}{2}\sigma^2)^{\frac{L}{2}}\right)$ *respectively.*

Theorem 10, combined with Theorem 6 and Theorem 9 provides us with a very fundamental implication.

Theorem 12 (Main result)**.** *For* $d = \Omega(L)$ *the initialization in [Glorot and Bengio, 2010] & [He et al., 2015] is guaranteed to stabilize both the gradient and the Hessian* ***with high probability****. Instead, if* $d \ll \mathcal{O}(L)$*, then both these initializations can yield vanishing gradients/curvature almost surely.*

The theorem above simply illustrates that there must be a transition in behavior from the $d = 1$ case to the $d = \Omega(L)$ case. Arguably, the precise characterization of this transition is of little theoretical interest — as the fundamental insight is the existence of such transition (see also Table 1). Empirically, we observe that $d = \sqrt{L}$ still yields vanishing gradients.

Consequences for traditional LeCun init. The above results point to an important consequence of the standard way of initialization prior to [Glorot and Bengio, 2010]. When choosing $\sigma^2 = \frac{1}{3d}$ as in LeCun init. [LeCun et al., 1998], gradient- and Hessian off-diagonal norms in (e.g.) linear nets vanish as $\mathcal{O}((\frac{1}{3})^{L/2})$ and Hessian diagonal blocks vanish even faster, namely at $\mathcal{O}((\frac{1}{3})^L)$ (Fig. 5 & 14). This points to an important fact about the eigenspectrum. Since $\mathbb{E}\|\nabla^2\mathcal{L}(\mathbf{W})\|_F$ scales as $(d\sigma^2)^{\frac{L}{2}}$ by Thm. 10, one of the Ld^2 eigenvales must have a magnitude $(d\sigma^2)^{\frac{L}{2}}$ in expectation. Yet, the fast diagonal vanishing lets the trace scale as $(d\sigma^2)^L$. As a result, the sum of the eigenvalues is exponentially smaller than the maximum eigenvalue and hence there must be eigenvalues of opposite sign (see e.g. Fig. 5). In summary, as in neural chains, optimizers are initialized in a flat plateau with almost no gradient signal and both positive as well as negative, but very small eigenvalues.

Consequences for Xavier and He init. Thm. 9 & 10 provide simple theoretical grounding for the benefits of increasing width in random neural networks.[5] However, they also point out an important limitation in the analysis and applicability of [Glorot and Bengio, 2010, He et al., 2015]. What happens in cases where width scales sub-linearly with depth remains open for theoretical analysis.

	$d = 1$	$d = O(L)$
Ours	**vanish w.h.p. (Thm. 6)**	**stable w.h.p. (Thm. 9)**
[Glorot and Bengio, 2010, He et al., 2015]	stable in exp.	stable in exp.
[Hanin and Rolnick, 2018, Hanin and Nica, 2019]	-	stable w.h.p.

Table 1: Summary of existing claims for He/Xavier initialization for different scalings of width d w.r.t depth L.

Our study on the neural chain (Thm. 6) along with Thm. 9 suggests that no initialization variance σ^2 is capable of stabilizing the full *distribution* of all of activation-, gradient- and the Hessian norm when the width is small. Yet, characterizing the almost sure behaviour of the gradient and Hessian for networks with $d > 1$ is intricate since paths in fully connected networks overlap, such that one cannot treat them as a set of independent products of random variables, which would allow a straight forward generalization of Thm. 6 using Berry-Esseen inequality [Berry, 1941]. Similarly, deriving the distribution of the forward pass is very challenging. In fact, already in the scalar case (neural chain), product distributions for both uniform [Dettmann and Georgiou, 2009] and Gaussian initialization become very complex [Springer and Thompson, 1970]. We thus retreat to empirical simulations, but stress the fact that these are informative because they are undertaken in a controlled setting, where the only source of randomness (weight initialization) is well controlled by running multiple seeds.

Our empirical results highlight that $\sigma^2 = \frac{1}{d}$ is indeed not an optimal choice for narrow networks. Indeed, as can be seen in Fig.4, gradients and curvature vanish in narrow but stay stable in wide ReLU MLPs with He init. The same happens in linear networks with Xavier init (Fig. 16). Again, we find both negative and positive eigenvalues at initialization (Fig. 13). Interestingly, Fig. 21 depicts that the optimal initialization

[5] [Hanin and Nica, 2019, Allen-Zhu et al., 2019] come to similar conclusions albeit with more complex analysis.

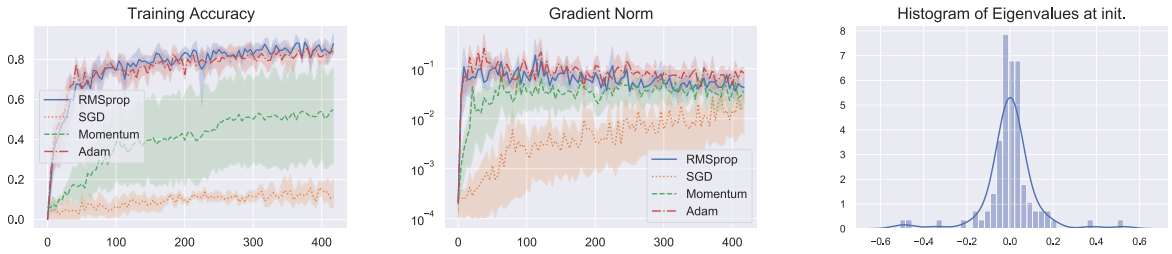


Figure 6: Fashion-MNIST on a narrow (32 hidden units) 128 layer ReLU MLP with *He init.* Mean and 95% CI of 10 random seeds. See App. A.1 for hyperparameters and Fig. 11 for test accuracy. Note that RMSProp successfully trains despite the fact that both gradients and curvature vanished (also compare Fig. 4). Yet, as can be deduced from how gradient norms evolve over time, SGD struggles to escape the flat plateau which is not surprising given that the negative eigenvalues are very small (compare Fig. 10 for a wide version of this network.)

variance σ^2 is a function $g(d, L)$ of the width to depth ratio. This function is highly non-trivial as it depends not only on the number of paths but also on their mutual overlap. Next, we show that these considerations extend to convolutional neural networks.

5 Vanishing in convolutional networks

Next, we discuss the effects of increasing width in convolutional architectures. For simplicity, we consider an image to image learning setting with images of resolution $r \times r$, using fully convolutional networks (FCNs) with $k \times k$ filters (where k is odd), c channels in each layer and a padding of $(k-1)/2$ (such that the resolution does not change over depth). We first note that width is not as straight-forward to define in CNNs as it is in MLPs. In Sec. 3, we argue that it is the number of paths leading to an output neuron (as well as their level of overlap) that determines its activation magnitude. While there are $|\Gamma| = d^L$ paths in MLPs, the FCN architecture yields $|\Gamma| = (k^2 c)^L$ (assuming non-zero padding). Consequently, we define the effective network width as $d := k^2 c$. Obviously, there are then two ways to increase width, namely via the filter size k and the number of channels c. As in prior theoretical [Arora et al., 2019] and practical works [Zagoruyko and Komodakis, 2016], we focus on the latter.

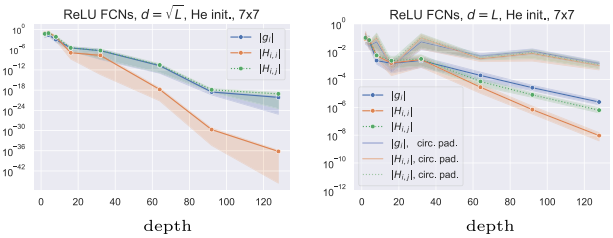


Figure 8: **Effect of width in CNNs:** Gradient/curvature on FCNs over depth. *Plots has different scales.* Mean and 95% CI of 15 runs. While increasing width helps for large resolution (see Fig. 17), for small resolutions such as 7x7 (above) vanishing occurs even when width scales as $\mathcal{O}(L)$ (albeit at slower rate (see y-axis)). Replacing zero- with circular padding mitigates this effect.

Empirical findings. We first consider the above introduced image to image learning setting with CIFAR-10 images as inputs and targets, once at the original 32×32 resolution and once downsized to 7×7. We fix the kernel size $k := 3$ and increase both the number of layers and channels. As can be seen in Fig.8, gradients/curvature vanish when depth grows faster than width and again the Hessian diagonal decreases fastest (top). Due to weight sharing, the effect is more pronounced as in MLPs (see App. D for discussion). When width grows linearly in depth (bottom), however, the beneficial effect is reduced when operating at small resolutions, where the pixel to padding ratio goes down such that zero-padding has a detrimental effect on the gradient/Hessian magnitude over depth. As indicated on the bottom right, this effect can be circumvented by opting for non-zero padding (circular in this case).

At first, 7×7 may seem unrealistically small but we remark that even when training on 224×224 ImageNet inputs, the last block of layers in ResNets indeed operates exactly at this scale [He et al., 2016a]. Consequently, vanishing also occurs in ResNet-type architectures, despite their large number of channels (Fig. 9). Here, similarly to [Yao et al., 2019], we feed images through ResNet architectures stripped of both batch normalization and residual connections, which we term *stripped_ResNets*. Interestingly, neither increasing the number of channels by two as in [Zagoruyko and Komodakis, 2016] (*wide_stripped_ResNets*) nor increasing the filter sizes from 1×1 to 3×3 and from 3×3 to 5×5 (*big_stripped_ResNets*) prevents gradient/curvature vanishing in deep *stripped ResNets*.

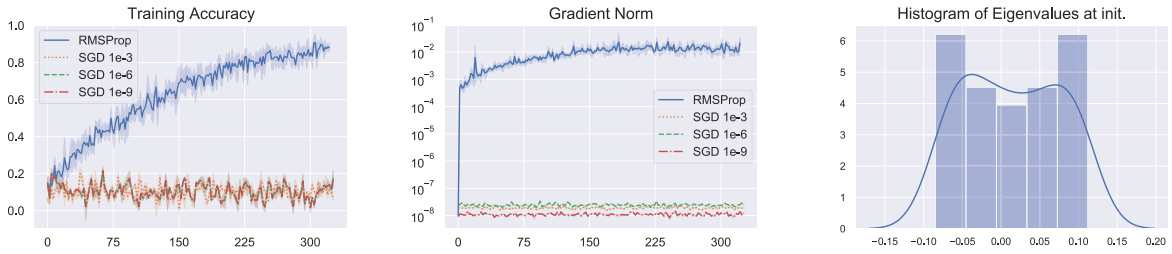


Figure 7: CIFAR-10 on a 500 layer *stripped_ResNet* with He init. Accuracy and gradient norm over epochs as well as eigenvalue histogram at initialization. Mean and 95% CI of 10 random seeds. See Appendix A.1 for hyperparameters and Figure 11 for test accuracy

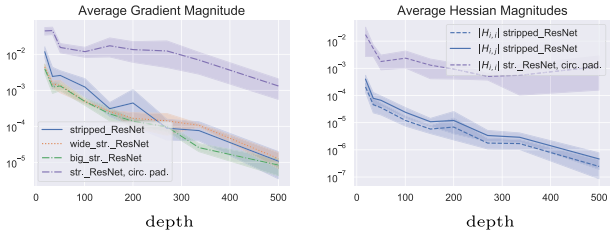


Figure 9: **Left:** Gradient in *stripped_ResNets* with He init. on CIFAR-10. Increasing number of channels (*wide_stripped_ResNet*) and kernel size (*big_stripped_ResNet*) does not help. Yet, as in Fig. 8, circular padding slows the effect down. **Right:** Average Hessian entry magnitudes in stripped ResNet (wide and big not computed due to memory limitations). Mean and 95% CI of 10 runs.

Architectural improvements As mentioned in the introduction, convolutional networks of depth 500 and more are not unheard-of. In fact, [He et al., 2016b] find that even going up to ResNet-1001 can improve test performance. Compared to the *stripped_ResNets* considered above, their architecture includes batch normalization and residual connections. Understanding the inner working of these components is an active area of research (e.g. [Hardt and Ma, 2016, Bjorck et al., 2018, Kohler et al., 2019, Zhang et al., 2019b]). Most related to our cases, [Labatie, 2019] suggests that only the combination of the two is effective in stabilizing information flow in the large depth limit, which is line with our finding that ResNets suffer from *exploding* gradients when taking either one of BN or residual connections out (Fig. 20).[6]

Algorithmic improvements. An obvious alternative is to generate robustness towards gradient/curvature vanishing directly in the training algorithm. As discussed in Sec. 3, adaptive methods (that were introduced about five years prior to batch normalization) are indeed able to escape flat initialization plateaus quickly. As a result, RMSprop can train a 500 layer *stripped_ResNet* without any normalization, skip-connections or learning-rate scheduling (Fig. 7).

[6]Figure 18 & 19 confirm this for MLPs and FCNs. See also [Yang et al., 2019] and [Zhang et al., 2019a]

6 Conclusion and outlook

Despite its long standing history, the phenomenon of vanishing and exploding gradients still lacks a comprehensive explanation. In this retrospective work we extended the current state of knowledge by: (i) showing that vanishing gradients co-occur with vanishing curvature, which clarifies the detrimental effect of depth on standard gradient descent at random initialization; (ii) examining when and why vanishing gradients occur in the first place. In this regard, we highlighted the role of effective width as well as the importance of combining Batch Normalization with residual connections; (iii) pointing to a remarkable curvature adaption property of adaptive gradients methods which allows them to train very deep networks despite the above mentioned issues.

Looking ahead, we regard investigations into quantifying the optimal variance for narrow MLPs as well as for un-normalized convnets an interesting follow-up. Furhtermore, clarifying the interplay of initialization and activation functions (Fig. 19) poses yet another interesting question. On a different route, we consider moving away from i.i.d. initialization and instead developing balanced initialization schemes that couple layers but preserve randomness a promising direction for designing simple deep neural network architectures that train and generalize well with gradient based methods.

References

[Allen-Zhu et al., 2019] Allen-Zhu, Z., Li, Y., and Song, Z. (2019). A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR.

[Arjevani et al., 2019] Arjevani, Y., Carmon, Y., Duchi, J. C., Foster, D. J., Srebro, N., and Wood-

worth, B. (2019). Lower bounds for non-convex stochastic optimization. *arXiv preprint:1912.02365*.

[Arora et al., 2019] Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R. R., and Wang, R. (2019). On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, pages 8141–8150.

[Balles and Hennig, 2018] Balles, L. and Hennig, P. (2018). Dissecting Adam: The sign, magnitude and variance of stochastic gradients. In *International Conference on Machine Learning*, pages 404–413. PMLR.

[Bengio et al., 1994] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

[Berry, 1941] Berry, A. C. (1941). The accuracy of the Gaussian approximation to the sum of independent variates. *Transactions of the American Mathematical Society*, 49(1):122–136.

[Biggio et al., 2021] Biggio, L., Bendinelli, T., Neitz, A., Lucchi, A., and Parascandolo, G. (2021). Neural symbolic regression that scales. *arXiv preprint arXiv:2106.06427*.

[Bjorck et al., 2018] Bjorck, J., Gomes, C., Selman, B., and Weinberger, K. Q. (2018). Understanding batch normalization. *arXiv preprint:1806.02375*.

[Brock et al., 2018] Brock, A., Donahue, J., and Simonyan, K. (2018). Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*.

[Brown et al., 2020] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *arXiv preprint:2005.14165*.

[Chen et al., 2020] Chen, J., Zhou, D., Tang, Y., Yang, Z., Cao, Y., and Gu, Q. (2020). Closing the generalization gap of adaptive gradient methods in training deep neural networks. In *IJCAI*.

[Choi, 1994] Choi, K. P. (1994). On the medians of gamma distributions and an equation of Ramanujan. *Proceedings of the American Mathematical Society*, 121(1):245–251.

[Curtis and Robinson, 2019] Curtis, F. E. and Robinson, D. P. (2019). Exploiting negative curvature in deterministic and stochastic optimization. *Mathematical Programming*, 176(1-2):69–94.

[Cutkosky and Orabona, 2019] Cutkosky, A. and Orabona, F. (2019). Momentum-based variance reduction in non-convex sgd. *Advances in neural information processing systems*, 32.

[Dai et al., 2020] Dai, X., Wan, A., Zhang, P., Wu, B., He, Z., Wei, Z., Chen, K., Tian, Y., Yu, M., Vajda, P., et al. (2020). Fbnetv3: Joint architecture-recipe search using neural acquisition function. *arXiv e-prints*, pages arXiv–2006.

[Daneshmand et al., 2020] Daneshmand, H., Kohler, J., Bach, F., Hofmann, T., and Lucchi, A. (2020). Batch normalization provably avoids ranks collapse for randomly initialised deep networks. *Advances in Neural Information Processing Systems*, 33.

[Daneshmand et al., 2018] Daneshmand, H., Kohler, J., Lucchi, A., and Hofmann, T. (2018). Escaping saddles with stochastic gradients. *arXiv preprint:1803.05999*.

[Dauphin et al., 2015] Dauphin, Y., de Vries, H., and Bengio, Y. (2015). Equilibrated adaptive learning rates for non-convex optimization. In *NIPS*.

[De and Smith, 2020] De, S. and Smith, S. (2020). Batch normalization biases residual blocks towards the identity function in deep networks. *Advances in Neural Information Processing Systems*, 33.

[Défossez et al., 2020] Défossez, A., Bottou, L., Bach, F., and Usunier, N. (2020). On the convergence of Adam and Adagrad. *arXiv preprint:2003.02395*.

[Dettmann and Georgiou, 2009] Dettmann, C. P. and Georgiou, O. (2009). Product of n independent uniform random variables. *Statistics & probability letters*, 79(24):2501–2503.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Devroye, 2006] Devroye, L. (2006). Nonuniform random variate generation. *Handbooks in operations research and management science*, 13:83–121.

[Dosovitskiy et al., 2020] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[Du et al., 2019] Du, S., Lee, J., Li, H., Wang, L., and Zhai, X. (2019). Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pages 1675–1685. PMLR.

[Du et al., 2017] Du, S. S., Jin, C., Lee, J. D., Jordan, M. I., Singh, A., and Poczos, B. (2017). Gradient descent can take exponential time to escape saddle points. In *Advances in neural information processing systems*, pages 1067–1077.

[Duchi et al., 2011] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

[Fang et al., 2019] Fang, C., Lin, Z., and Zhang, T. (2019). Sharp analysis for nonconvex sgd escaping from saddle points. *arXiv preprint:1902.00247*.

[Gemp and McWilliams, 2019] Gemp, I. and McWilliams, B. (2019). The unreasonable effectiveness of Adam on cycles. *NeurIPS Workshop on Bridging Game Theory and Deep Learning*.

[Gershgorin, 1931] Gershgorin, S. (1931). On the location of eigenvalues of a matrix. *Izv. Akad. Nauk SSSR, ser. Fiz. Mat. v6*, 749.

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

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

[Goodfellow et al., 2014] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. (2014). Generative adversarial nets. In *NIPS*.

[Hanin and Nica, 2019] Hanin, B. and Nica, M. (2019). Products of many large random matrices and gradients in deep neural networks. *Communications in Mathematical Physics*, pages 1–36.

[Hanin and Rolnick, 2018] Hanin, B. and Rolnick, D. (2018). How to start training: The effect of initialization and architecture. In *Advances in Neural Information Processing Systems*, pages 571–581.

[Hardt and Ma, 2016] Hardt, M. and Ma, T. (2016). Identity matters in deep learning. *arXiv preprint:1611.04231*.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

[He et al., 2016a] He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[He et al., 2016b] He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer.

[Hochreiter, 1991] Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1).

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Howard et al., 2019] Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. (2019). Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1314–1324.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint:1502.03167*.

[Jacot et al., 2018] Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*.

[Karras et al., 2020a] Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., and Aila, T. (2020a). Training generative adversarial networks with limited data. *arXiv preprint:2006.06676*.

[Karras et al., 2020b] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2020b). Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint:1412.6980*.

[Kohler et al., 2019] Kohler, J., Daneshmand, H., Lucchi, A., Hofmann, T., Zhou, M., and Neymeyr, K. (2019). Exponential convergence rates for batch normalization: The power of length-direction decoupling in non-convex optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 806–815. PMLR.

[Krizhevsky, 2009] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. *cs.toronto.edu*.

[Kunstner et al., 2019] Kunstner, F., Hennig, P., and Balles, L. (2019). Limitations of the empirical Fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems*, pages 4156–4167.

[Labatie, 2019] Labatie, A. (2019). Characterizing well-behaved vs. pathological deep neural networks. In *International Conference on Machine Learning*, pages 3611–3621. PMLR.

[LeCun et al., 1998] LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (1998). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer.

[Li et al., 2020] Li, X., Gu, Q., Zhou, Y., Chen, T., and Banerjee, A. (2020). Hessian based analysis of sgd for deep nets: Dynamics and generalization. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 190–198. SIAM.

[Liu et al., 2019] Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. (2019). On the variance of the adaptive learning rate and beyond. *arXiv preprint:1908.03265*.

[Loshchilov and Hutter, 2017] Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint:1711.05101*.

[Mallows and Richter, 1969] Mallows, C. L. and Richter, D. (1969). Inequalities of Chebyshev type involving conditional expectations. *The Annals of Mathematical Statistics*, 40(6):1922–1932.

[Martens, 2020] Martens, J. (2020). New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21:1–76.

[Mildenhall et al., 2020] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer.

[Nesterov et al., 2018] Nesterov, Y. et al. (2018). *Lectures on convex optimization*, volume 137. Springer.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

[Park et al., 2019] Park, T., Liu, M.-Y., Wang, T.-C., and Zhu, J.-Y. (2019). Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2337–2346.

[Pascanu et al., 2013] Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

[Pennington et al., 2018] Pennington, J., Schoenholz, S., and Ganguli, S. (2018). The emergence of spectral universality in deep networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1924–1932.

[Reddi et al., 2019] Reddi, S. J., Kale, S., and Kumar, S. (2019). On the convergence of Adam and beyond. *arXiv preprint:1904.09237*.

[Saxe et al., 2013] Saxe, A. M., McClelland, J. L., and Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint:1312.6120*.

[Schoenholz et al., 2016] Schoenholz, S. S., Gilmer, J., Ganguli, S., and Sohl-Dickstein, J. (2016). Deep information propagation. *arXiv preprint:1611.01232*.

[Springer and Thompson, 1970] Springer, M. D. and Thompson, W. E. (1970). The distribution of products of beta, gamma and Gaussian random variables. *SIAM Journal on Applied Mathematics*, 18(4):721–737.

[Staib et al., 2019] Staib, M., Reddi, S., Kale, S., Kumar, S., and Sra, S. (2019). Escaping saddle points with adaptive gradient methods. In *International Conference on Machine Learning*, pages 5956–5965. PMLR.

[Tan et al., 2019] Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le,

Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828.

[Tan and Le, 2019] Tan, M. and Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR.

[Telgarsky, 2016] Telgarsky, M. (2016). Benefits of depth in neural networks. *arXiv preprint:1602.04485*.

[Temme, 1996] Temme, N. M. (1996). *Special functions: An introduction to the classical functions of mathematical physics*. John Wiley & Sons.

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

[Tolstikhin et al., 2021] Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Keysers, D., Uszkoreit, J., Lucic, M., et al. (2021). Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*.

[Touvron et al., 2021] Touvron, H., Bojanowski, P., Caron, M., Cord, M., El-Nouby, A., Grave, E., Joulin, A., Synnaeve, G., Verbeek, J., and Jégou, H. (2021). Resmlp: Feedforward networks for image classification with data-efficient training. *arXiv preprint arXiv:2105.03404*.

[Wilson et al., 2017] Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., and Recht, B. (2017). The marginal value of adaptive gradient methods in machine learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4151–4161.

[Wolf et al., 2020] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

[Xiao et al., 2017] Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint:1708.07747*.

[Yang et al., 2019] Yang, G., Pennington, J., Rao, V., Sohl-Dickstein, J., and Schoenholz, S. S. (2019). A mean field theory of batch normalization. *arXiv preprint:1902.08129*.

[Yao et al., 2019] Yao, Z., Gholami, A., Keutzer, K., and Mahoney, M. (2019). Pyhessian: Neural networks through the lens of the hessian. *arXiv preprint:1912.07145*.

[Zagoruyko and Komodakis, 2016] Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv preprint:1605.07146*.

[Zhang et al., 2019a] Zhang, H., Dauphin, Y. N., and Ma, T. (2019a). Fixup initialization: Residual learning without normalization. *arXiv preprint:1901.09321*.

[Zhang et al., 2019b] Zhang, H., Yu, D., Yi, M., Chen, W., and Liu, T.-Y. (2019b). Convergence theory of learning over-parameterized resnet: A full characterization. *arXiv preprint:1903.07120*.

[Zhang et al., 2019c] Zhang, J., He, T., Sra, S., and Jadbabaie, A. (2019c). Why gradient clipping accelerates training: A theoretical justification for adaptivity. In *International Conference on Learning Representations*.

[Zhang et al., 2020] Zhang, J., Karimireddy, S. P., Veit, A., Kim, S., Reddi, S., Kumar, S., and Sra, S. (2020). Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33.

[Zhu et al., 2018] Zhu, Z., Wu, J., Yu, B., Wu, L., and Ma, J. (2018). The anisotropic noise in stochastic gradient descent: Its behavior of escaping from minima and regularization effects.

[Zhu et al., 2019] Zhu, Z., Wu, J., Yu, B., Wu, L., and Ma, J. (2019). The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In *ICML*, pages 7654–7663.

[Zou et al., 2020] Zou, D., Cao, Y., Zhou, D., and Gu, Q. (2020). Gradient descent optimizes over-parameterized deep ReLU networks. *Machine Learning*, 109(3):467–492.

A Additional experimental details and results

A.1 Experimental setup

All experiments are conducted in PyTorch 1.7 [Paszke et al., 2019]. We run experiments on up to 8 Tesla V100 GPUs with 32 GB memory.

Figure 1 We draw samples $\mathbf{X} \in \mathbb{R}^{n \times d}, \mathbf{y} \in \{0,1\}^n$ from two interleaving half circles using *sklearn.datasets.make_moons* [Pedregosa et al., 2011], choosing $n = 1000$ and $d = 2$. In order to classify these samples, we train ReLU MLPs of width 2 (with biases) and increasing depth using SGD and Adam with learning rates $\eta = 0.1$ and $\eta = 0.001$ respectively and batch size $bs = 100$. Both methods use $\beta = 0.9$ momentum. Simply increasing the learning rate for SGD won't solve the problem since this yields instabilities in the long run due to sharply increasing gradient magnitudes outside the plateau. The loss landscape is computed for the full batch. The projections were computed using a PCA on the visited parameters of both optimizers.

Figure 5 & 14. We draw samples $\mathbf{X} \in \mathbb{R}^{n \times d}$ from a multivariate Gaussian distribution $\mathcal{N}(0, \mathbf{I}_d)$, choosing $n = 100$ and d equal to the network width (x-axis). Targets $\mathbf{Y} \in \mathbb{R}^{n \times d}$ are generated form a 1-hidden-layer network of width as on the x-axis. We use as mean-squared-error loss. The networks are simple MLPs with weight matrices in $d \times d$. As initialization we use LeCun uniform, i.e. $W^\ell_{i,j} \sim \mathcal{U}\left[-\frac{1}{\sqrt{d}}, \frac{1}{\sqrt{d}}\right]$. As long as memory is sufficient we compute the full hessian, above we randomly sub-sample hessian blocks (layers).

Figure 4 & 18. As in Figure 5 & 14 but using Xavier uniform initialization $W^\ell_{i,j} \sim \mathcal{U}\left[-\sqrt{\frac{3}{d}}, \sqrt{\frac{3}{d}}\right]$ [Glorot and Bengio, 2010] for linear- and He uniform initialization $W^\ell_{i,j} \sim \mathcal{U}\left[-\sqrt{\frac{6}{d}}, \sqrt{\frac{6}{d}}\right]$ [He et al., 2015] for ReLU networks. The networks are simple MLPs with weight matrices in $\left\lceil\sqrt{d}\right\rceil \times d$ in the top and $d \times d$ in the bottom row.

In Figure 18 we add residual connections and batch normalization. Notably, the residual connections skip a set of *three* layers.

Figure 6. We train Fashion-MNIST [Xiao et al., 2017] with the given train-test split, on a 32 hidden unit, 128 hidden layer MLP with ReLU activations. All optimizers are depicted with (individually) grid-searched learning rate (in terms of training accuracy) in the set $1e-3, 5e-4, 1e-4, 5e-5, 1e-5, 5e-6, 1e-6, 5e-7, 1e-7, 5e-8, 1e-8$. Depicted are SGD and Momentum with learning rate $1e-4$ and RMSprop as well as Adam with learning rate $1e-5$. Batch size is 128 for all optimizers. The momentum factor for SGD was set to 0.9.

Figure 10. Same setting as in Fig. 6 but with network width equal to network depth (128).

Figure 8 & 19. We here consider a fully convolutional image to image learning setting where each layer has c kernels of size 3×3 that operate with a padding of 1. As a result, the image resolution does not change over depth. As inputs, we use a batch of 32 CIfAR-10 images [Krizhevsky, 2009]. We feed them trough the networks once at the original 32×32 resolution and once down-sampled to 7×7 images and compute a mean-squared-error loss at the end using the input image as target. We show plots for ReLU nets but note that the general picture is the same for linear networks (vanishing happens just a bit slower, compare MLPs).

In Figure 19 we add residual connections and batch normalization. Notably, the residual connections skip a set of *three* layers (similarly to the ResNet). In fact, we found exploding gradients/curvature when residual connections only skip one layer.

Figure 9 & 20. In these figures we feed CIFAR-10 images [Krizhevsky, 2009] at the original 32×32 resolution through convolutional networks that resemble the ResNet architecture but omit both Batch Normalization and residual connections. These networks have 4 main blocks of layers which operate at image resolutions: 56×56, 28×28, 14×14 and 7×7 and with 64, 128, 256 and 512 channels. Each of these blocks consists of 3 convolutional layers. We depict result on networks of depth $18, 34, 50, 101, 152, 200, 270, 336, 500$ which have the following block

configurations

$$[2,2,2,2],[2,4,4,2],[3,4,6,3],[3,4,23,3],[3,8,36,3]$$
$$[3,24,36,3],[3,36,48,3],[3,44,62,3],[3,70,90,3].$$

Unless stated differently, these networks use ReLU activations.

When neither Batch Norm nor residual connections are present we called the network *stripped_resnet*. The *wide_stripped_resnet* has twice the number of channels in each block compared to the ones stated in the previous paragraph. The *big_stripped_resnet* has the original number of channels but all 3×3 filters are replaced by 5×5 - and all 1×1 filters are replaced by 3×3 filters.

Figure 7. We train CIFAR-10 on a *stripped_resnet* with 500 layers. All optimizers are depicted with (individually) grid-searched learning rate (in terms of training accuracy) in the set $1e-3, 5e-4, 1e-4, 5e-5, 1e-5, 5e-6, 1e-6, 5e-7, 1e-7, 5e-8, 1e-8$. Batch size is 64 for all optimizers. For RMSprop, the best found learning rate was $1e-6$.

A.2 Additional results

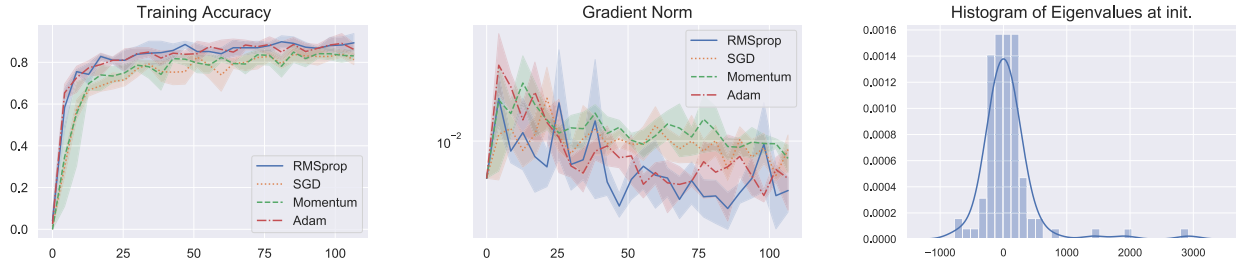


Figure 10: Addendum to Fig. 6: Training **Fashion-MNIST** on a ***wide*** (128 units per layer) **128 layer ReLU MLP** (He init.) with batch size 64 and grid-searched learning rates. Training accuracy and gradient magnitude over epochs as well as eigenvalues at initialization. Mean and 95% CI of 10 random seeds. Clearly, increased width prevents vanishing and thus allows SGD to train.

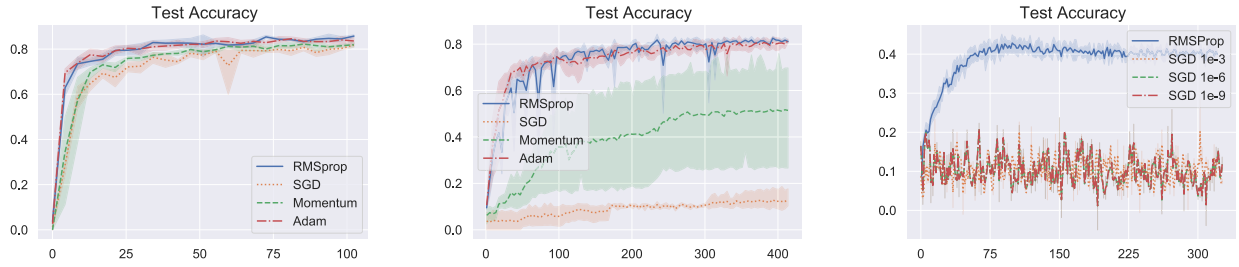


Figure 11: **Test set performance** of the wide- and narrow Fashion-MNIST ReLU MLP (left and middle) as well as on the CIFAR-10 stripped ResNet500 (right). It becomes evident that RMSProp heavily overfits in the stripped ResNet, which is not surprising given that the network has no regularization what so ever and memorizing CIFAR-10 is comparatively easy for such large convnets.

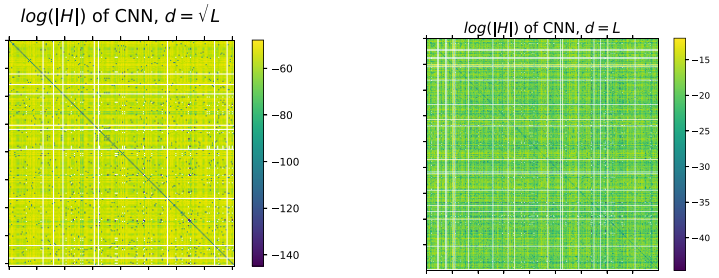


Figure 12: Addendum to Fig. 8. **Hessians of fully convolutional ReLU networks** of depth $L = 128$ on downscaled CIFAR-10 samples at random initializatio (He init.). Contrary to the MLP case in Fig. 5, the Hessians are no longer approximately block-hollow but approximately hollow. Furthermore, as expected, the hollowness decreases in network width (left to right).

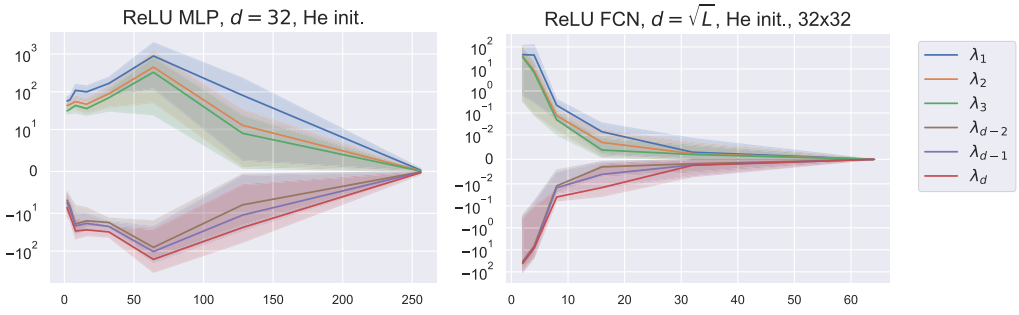


Figure 13: **Eigenvalues over depth:** Largest $(\lambda_1, \lambda_2, \lambda_3)$ and smallest $(\lambda_{d-2}, \lambda_{d-1}, \lambda_d)$ three eigenvalues on a symlog scale over depth. Left: ReLU MLPs of width 32. Right: ReLU FCNs of $d = \sqrt{L}$. Both on Fashion-MNIST. Mean and 95% CI of 5 random seeds.

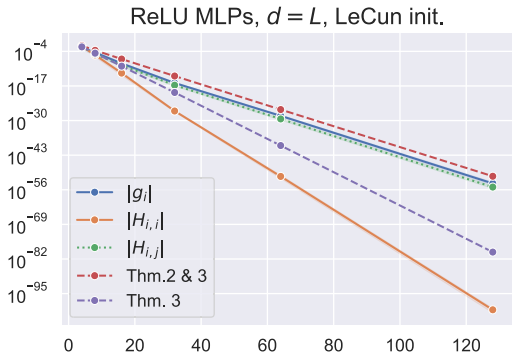


Figure 14: Addendum to Fig. 5. Vanishing gradient and Hessian for deep **ReLU MLPs with LeCun init.**. x-axis depicts depth *and* width of the networks. Mean and 95% CI of 10 random seeds.

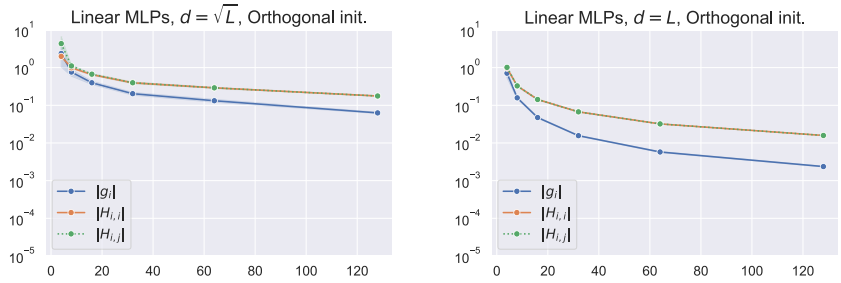


Figure 15: Addendum to Fig. 4: Gradient/curvature on **linear MLPs** over depth initialized with **orthogonal initialization** [Saxe et al., 2013]. This strategy is very robust towards gradient/curvature vanishing on linear MLPs but quickly yields absolute zeros on ReLU MLPs (PyTorch implementation [Paszke et al., 2019], not shown). Mean and 95% CI of 15 runs.

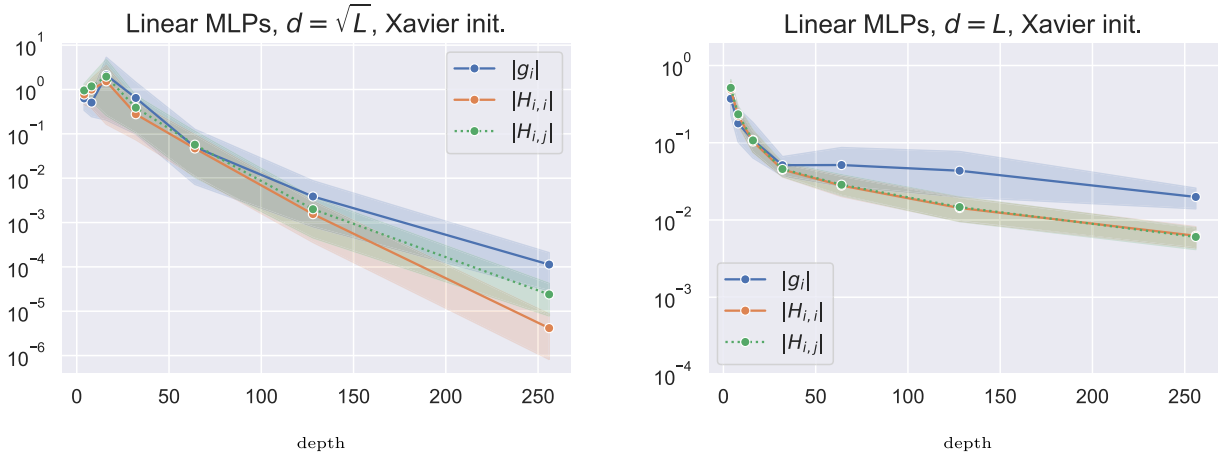


Figure 16: Addendum to Fig. 4: Effect of width in **linear** MLPs: Gradient and curvature scaling on Fashion-MNIST over depth. While quantities vanish on the left, where the width scales only as square-root of depth, the right shows stable behaviour. Mean and 95% CI of 15 runs.

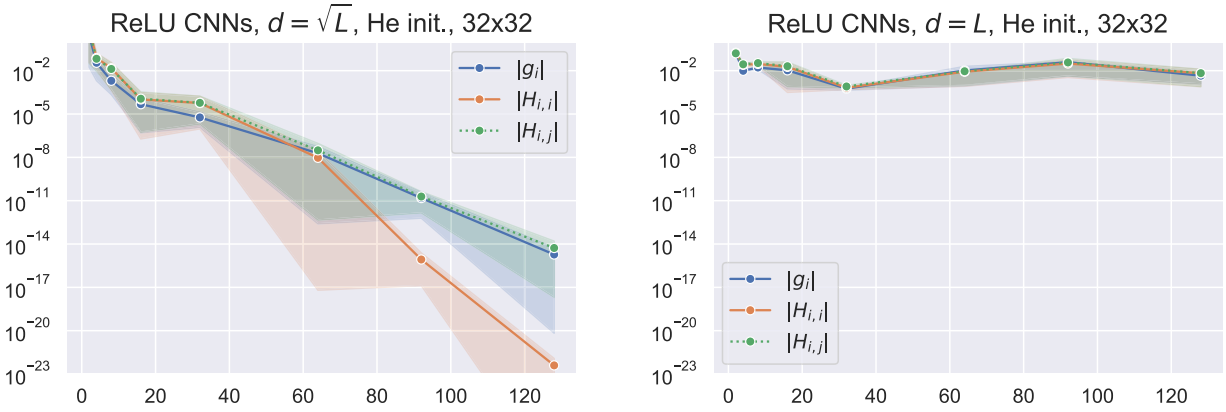


Figure 17: Addendum to Figure 8: Gradient/curvature on **FCNs** over depth at large resolution (32x32). Mean and 95% CI of 15 runs.

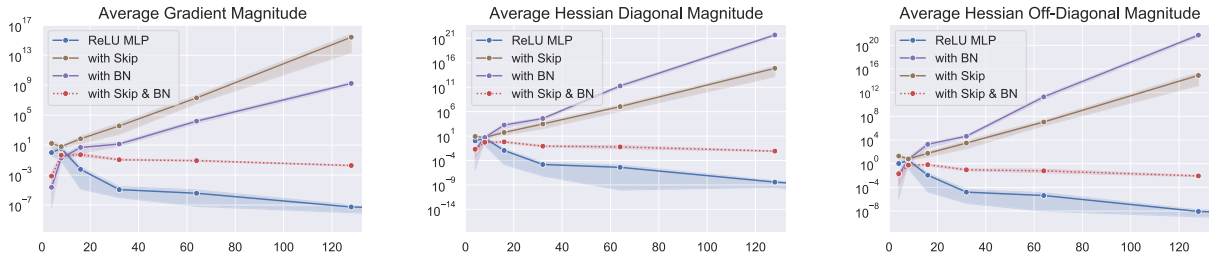


Figure 18: Addendum to Fig. 4: Gradient and Hessian magnitudes in **ReLU MLPs** over depth. Depicted is the ReLU MLP with $d = \sqrt{L}$ and He init. Here, we add Batch Normalization and skip connections to the network.

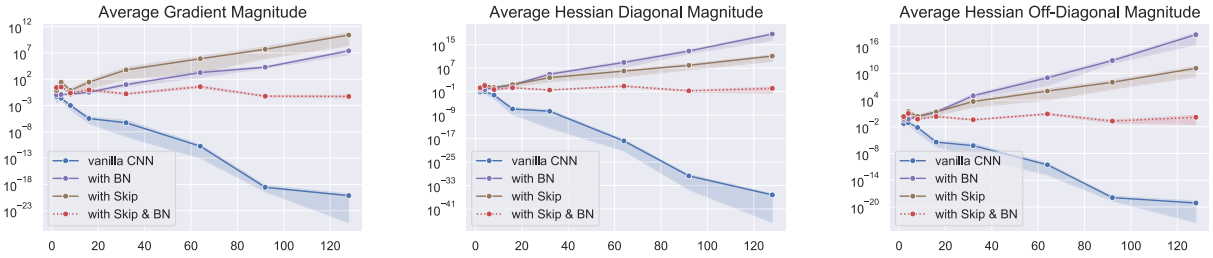


Figure 19: Addendum to Fig. 8: Gradient and Hessian magnitudes in **FCNs** over depth. Depicted is the ReLU FCN with $d = \sqrt{L}$ at 7×7 resolution initialized with He init. (Compare 8 top right). Here, we add Batch Normalization and skip connections to the network.

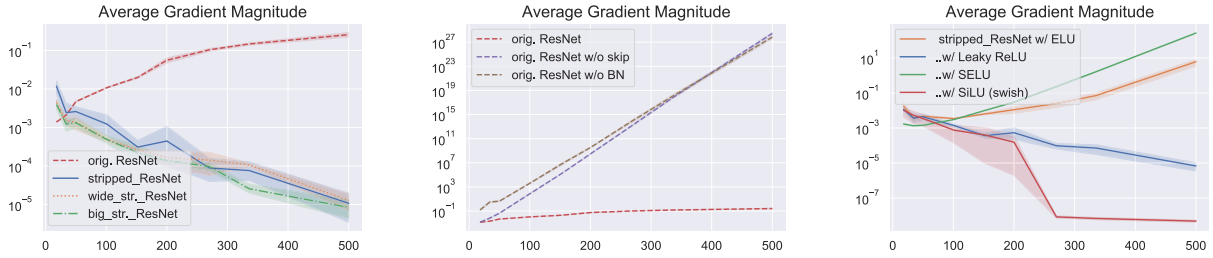


Figure 20: Addendum to Fig. 9: **Left: Including original ResNet** architecture (i.e. with BN and skip connections). **Middle:** original ResNet plus version with just BN or Skip connections.**Right:** Effect of different activations functions in **stripped ResNets**.

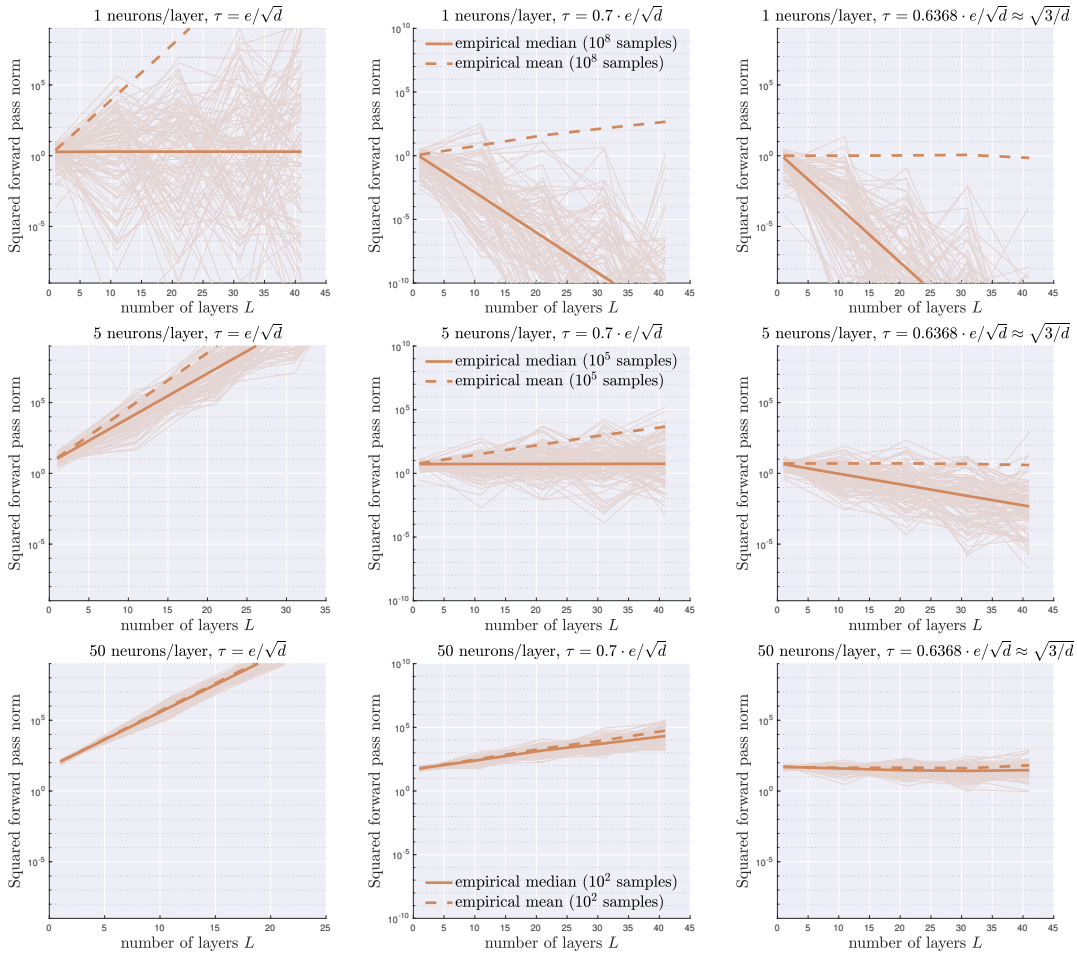


Figure 21: Behavior of the variable $\|\mathbf{W}^L\mathbf{W}^{L-1}\cdots\mathbf{W}^1\mathbf{1}\|_2^2$. Entries randomly sampled $\mathcal{U}([-\tau,\tau])$. Only 50 samples are shown, but 10^6 are used to approximate population quantities. The expectation is by rare events, and is drastically different from the median if $d \ll L$, as shown in Thm. 9.

B Analysis of deep linear and ReLU Networks

Note: the fundamental result of this Section (Thm. 18) is checked numerically in Figure 22.

B.1 Proof Theorem 6 (neural chains)

Proof:

We focus on the derivatives with respect to the parameters w_1 and w_2 since the argument can be repeated for any parameter w_i. Denote by $\mathbf{w}$ the set of all parameters $\{w_i\}_{i=1}^L$. The derivatives of the chain loss are

$$|\nabla_{w^1}\mathcal{L}(\mathbf{w})| = |yw^L\cdots w^2x - (w^L)^2\cdots(w^2)2^2(w^1)x| \tag{7}$$

$$|\nabla^2_{w^1,w^2}\mathcal{L}(\mathbf{w})| = |yw^L\cdots w^3x - 2(w^L)^2\cdots(w^3)^2w^2w^1x| \tag{8}$$

and

$$\left|\nabla^2_{w^1,w^1}\mathcal{L}(\mathbf{w})\right| = \left|(w^L)^2...(w^2)^2x\right| \tag{9}$$

It can be seen that the gradient as well as the Hessian off-diagonals scale similarly in depth. Let us first consider these two. Clearly,

$$\begin{aligned}|\nabla_{w^1}\mathcal{L}(\mathbf{w})| &\leq |yw^L...w^2x| + |(w^L)^2...(w^2)^2w^1x| \\ &\leq 2\cdot\max\{|yw^L...w^2x|, |(w^L)^2...(w^2)^2w^1x|\}\end{aligned} \tag{10}$$

For the first term in the max, we note that $\ln\left(\prod_{k=2}^L |w^k|\right) = \sum_{k=2}^L \ln(|w^k|)$ and since $|w^k| \sim \mathcal{U}[0,\tau]$, we have $\mathbb{E}\left[\ln(|w^k|)\right] = \ln(\tau) - 1$. Thus, the strong law of large numbers yields that with probability one

$$\prod_{k=2}^{L} |w^k| \to \exp\left((L-1)\left(\ln\left(\tau\right) - 1\right)\right) \tag{11}$$

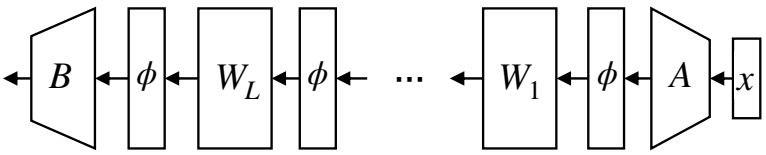
For the second term, we have $\ln\left(w^1\prod_{k=2}^L (w^k)^2\right) = \ln(w^1) + \sum_{k=2}^L \ln((w^k)^2)$ and $\mathbb{E}\left[\ln((w^k)^2)\right] = 2\left(\ln(\tau) - 1\right)$. Thus, the strong law of large numbers yields that with probability one $\ln\left(w^1\prod_{k=2}^L (w^k)^2\right) \to (2L-1)\left(\ln(\tau) - 1\right)$. Hence

$$w^1\prod_{k=2}^{L}(w^k)^2 \to \exp\left((2L-1)\left(\ln\left(\tau\right) - 1\right)\right). \tag{12}$$

For large L, Eq.11 clearly dominates Eq.12, which proves the first statement. The second statement follows similarly to Eq. 12.

□

B.2 Notation and fundamental properties



Notation. In this section, we use neural network notations similar to the one in [Allen-Zhu et al., 2019]. In particular,

$$F(\mathbf{x}) := \mathbf{B}\mathbf{D}^L\mathbf{W}_\phi^{L:1}\mathbf{A}\mathbf{x}, \quad \mathbf{W}_\phi^{L:1} := \mathbf{W}^L\mathbf{D}^{L-1}\mathbf{W}^{L-1}\cdots\mathbf{W}^2\mathbf{D}^1\mathbf{W}^1\mathbf{D}^0, \tag{13}$$

where $F(\mathbf{x})$ constitutes the forward pass of a given input $\mathbf{x} \in \mathbb{R}^{d_{in}}$, $\mathbf{A} \in \mathbb{R}^{d\times d_{in}}$,$\mathbf{B} \in \mathbb{R}^{d_{out}\times d}$, and $\mathbf{W}^\ell \in \mathbb{R}^{d\times d}, \forall \ell = 1,\dots,L$. $\mathbf{D}^\ell$ is the diagonal matrix of activation gates w.r.t the non-linearity ϕ at layer ℓ, which we consider to be either $\phi(x) = x$ (linear networks) or $\phi(x) = \max\{x, 0\}$ (ReLU networks). Finally, we denote by $\mathbf{a}^\ell$ and $\mathbf{h}^\ell$ the pre- and post activations in layer ℓ respectively, i.e. for example $\mathbf{a}^1 = \mathbf{A}\mathbf{x}$ and $\mathbf{h}^1 = \mathbf{D}^0\mathbf{a}^1$.

Fundamental properties of activations and preactivations at each layer

Lemma 13 (Fundamental properties of activations and preactivations at each layer)**.** *Let the entries of* $\mathbf{A}$*,* $\mathbf{B}$ *and each* $\mathbf{W}^\ell$ *be i.i.d. samples from a symmetric distribution around 0 with finite moments, and variance* σ^2*. Then for any fixed input* $\mathbf{x}$*, we have*

1. *At each layer, entries of the preactivation vector are integrable and have a distribution symmetric around zero.*
2. *For ReLU networks, at each layer, the entries of the activation vector are non-zero with probability* $1/2$*.*
3. *At each layer, in both the ReLU and the linear case, the preactivation and the activation vectors have uncorrelated squared entries.*

Proof. Recall that the preactivation at each layer is $\mathbf{a}^{\ell+1} = \mathbf{W}^\ell\mathbf{h}^\ell$. Clearly, since $\mathbf{W}^\ell = -\mathbf{W}^\ell$ in distribution, $\mathbf{a}^{\ell+1} = -\mathbf{a}^{\ell+1}$ in distribution. From this follows also that, if a ReLU is applied, $\mathbf{h}^{\ell+1} > 0$ with probability $1/2$. The last property to show is that squared entries of activations and preactivations are uncorrelated. Let's drop the layer index ℓ and pick two neurons $i \neq j$, then

$$\mathbb{E}[(g_i)^2(g_j)^2] = \sum_{r,s,u,v} \mathbb{E}\left[w_{ir}w_{is}\right]\mathbb{E}\left[w_{ju}w_{jv}\right]\mathbb{E}\left[h_rh_sh_uh_v\right] = \sigma^4\sum_{r,u}\mathbb{E}\left[(h_r)^2(h_u)^2\right]. \tag{14}$$

Instead, for the single squared variables we have

$$\mathbb{E}[(g_i)^2] = \sum_{r,s} \mathbb{E}\left[w_{ir}w_{is}\right]\mathbb{E}\left[h_rh_s\right] = \sigma^2\sum_{r}\mathbb{E}\left[(h_r)^2\right]. \tag{15}$$

Therefore $\mathbb{E}[(g_i)^2(g_j)^2] = \mathbb{E}[(g_i)^2]\mathbb{E}[(g_j)^2]$ if and only if $(h_i)^2$ and $(h_j)^2$ are uncorrelated. As $\mathbf{x}$ is fixed, this is the case at the input layer and we conclude the proof by induction on ℓ.

Last, we show the same properties for the activations in the ReLU case. Let $d_i = \phi(g_i)/g_i = \mathbb{1}(g_i > 0)$ and consider the new activation $h_i^+ = d_ig_i$. We start by applying the law of total expectation:

$$\begin{aligned}\mathbb{E}[(h_i^+)^2(h_j^+)^2] &= \mathbb{E}\left[(d_i)^2(d_j)^2(g_i)^2(g_j)^2\right] = \frac{1}{4}\mathbb{E}\left[(g_i)^2(g_j)^2|d_i, d_j = 1\right] \\ &= \frac{1}{4}\mathbb{E}\left[(g_i)^2(g_j)^2\right] = \frac{1}{2}\mathbb{E}\left[(g_i)^2\right]\frac{1}{2}\mathbb{E}\left[(g_j)^2\right] = \mathbb{E}[(h_i^+)^2]\mathbb{E}[(h_j^+)^2],\end{aligned} \tag{16}$$

where the third and the last equalities follow from the fact that the value of the squared preactivation is independent on the sign of the preactivation. □

Statistics for the propagation through one layer

Lemma 14 (Statistics after activation function)**.** *Let* $\mathbf{x}$ *be a symmetric random vector with uncorrelated squared entries. Let* $\phi(\mathbf{x}) = \mathbf{D}_\mathbf{x}\mathbf{x}$*. We have*

$$\mathbb{E}\|\mathbf{D}_\mathbf{x}\mathbf{x}\|_2^2 = p\ \mathbb{E}\|\mathbf{x}\|_2^2; \tag{17}$$

$$\mathbb{E}\|\mathbf{D}_\mathbf{x}\mathbf{x}\|_2^4 = p^2\mathbb{E}\|\mathbf{x}\|_2^4 + (p-p^2)\mathbb{E}\|\mathbf{x}\|_4^4; \tag{18}$$

$$\mathbb{E}\|\mathbf{D}_\mathbf{x}\mathbf{x}\|_4^4 = p\ \mathbb{E}\|\mathbf{x}\|_4^4. \tag{19}$$

where $p = 1$ *for linear nets and* $p = 1/2$ *for ReLU nets.*

Proof. The first property is based on the fundamental idea in [He et al., 2015]. Let d_i be the entry (i,i) of $\mathbf{D}_\mathbf{x}$. Then, d_i is independent from x_i^2. Hence, also noting that $(d_i)^2 = d_i$, we have

$$\mathbb{E}\|\mathbf{D}_\mathbf{x}\mathbf{x}\|_2^2 = \sum_i \mathbb{E}\left[d_i^2 x_i^2\right] = \sum_i \mathbb{E}\left[d_i\right]\mathbb{E}\left[x_i^2\right] = p\ \mathbb{E}\|\mathbf{x}\|_2^2. \tag{20}$$

where $p = \mathbb{E}[d_i]$, which is $1/2$ for ReLU nets and 1 for linear nets, as shown in Lemma 13. The last property can be proved in the same way by noting that d_i is independent from x_i^4

$$\mathbb{E}\|\mathbf{D}_\mathbf{x}\mathbf{x}\|_4^4 = \sum_i \mathbb{E}\left[d_i^4 x_i^4\right] = \sum_i \mathbb{E}\left[d_i\right]\mathbb{E}\left[x_i^4\right] = p\ \mathbb{E}\|\mathbf{x}\|_4^4. \tag{21}$$

The second property is a bit more involved to prove.

$$\mathbb{E}\|\mathbf{D}_\mathbf{x}\mathbf{x}\|_2^4 = \mathbb{E}\left(\sum_{i=1}^d (d_i x_i)^2\right)^2 = \sum_{i=1}^d \mathbb{E}\left[(d_i x_i)^4\right] + \sum_{i\neq j}\mathbb{E}\left[(d_i x_i)^2 (d_j x_j)^2\right]. \tag{22}$$

From Lemma 13, third point, we have $\mathbb{E}\left[(d_i x_i)^2 (d_j x_j)^2\right] = \mathbb{E}\left[(d_i x_i)^2\right]\mathbb{E}\left[(d_j x_j)^2\right]$ (compare Eq.(16)). Hence, noting again that $d_i = d_i^2$,

$$\mathbb{E}\|\mathbf{D}_\mathbf{x}\mathbf{x}\|_2^4 = \sum_{i=1}^d \mathbb{E}\left[d_i\right]\mathbb{E}\left[x_i^4\right] + \sum_{i\neq j}\mathbb{E}\left[d_i\right]\mathbb{E}\left[d_j\right]\mathbb{E}\left[x_i^2 x_j^2\right] \tag{23}$$

$$= p\ \mathbb{E}\|\mathbf{x}\|_4^4 + p^2\mathbb{E}\left[\|\mathbf{x}\|_2^4 - \|\mathbf{x}\|_4^4\right]. \tag{24}$$

This concludes the proof. □

The following corollary is of fundamental importance of understanding the properties of ReLU nets: if the input of the net is modified (say from $\mathbf{x}$ to α, the ReLU gates $\mathbf{D}_\mathbf{x}$ act as purely random Bernoulli gates. This comment can be also found in the proof of Lemma A.8 in [Zou et al., 2020].

Corollary 15 (Statistics after activation function with changed input)**.** *In the context of Lemma 14, let* $\boldsymbol{\alpha}$ *be a fixed vector. we have*

$$\mathbb{E}\|\mathbf{D}_\mathbf{x}\boldsymbol{\alpha}\|_2^2 = p\ \|\boldsymbol{\alpha}\|_2^2; \tag{25}$$

$$\mathbb{E}\|\mathbf{D}_\mathbf{x}\boldsymbol{\alpha}\|_2^4 = p^2\|\boldsymbol{\alpha}\|_2^4 + (p-p^2)\|\boldsymbol{\alpha}\|_4^4; \tag{26}$$

$$\mathbb{E}\|\mathbf{D}_\mathbf{x}\boldsymbol{\alpha}\|_4^4 = p\ \|\boldsymbol{\alpha}\|_4^4. \tag{27}$$

where $p = 1$ *for linear nets and* $p = 1/2$ *for ReLU nets.*

Proof. Just note that since $\mathbf{D}_\mathbf{x}$ and $\boldsymbol{\alpha}$ are independent we can basically follow the proof of Lemma 14, but simplified:

$$\mathbb{E}\|\mathbf{D}_\mathbf{x}\boldsymbol{\alpha}\|_m^m = \sum_i \mathbb{E}\left[d_i^m \alpha_i^m\right] = \sum_i \mathbb{E}\left[d_i\right]\alpha_i^m = p\ \|\boldsymbol{\alpha}\|_m^m. \tag{28}$$

The second property is also easy to show compared to Lemma 14:

$$\mathbb{E}\|\mathbf{D_x}\boldsymbol{\alpha}\|_2^4 = \mathbb{E}\left(\sum_{i=1}^d (d_i\alpha_i)^2\right)^2 \tag{29}$$

$$= \sum_{i=1}^d \mathbb{E}\left[(d_i\alpha_i)^4\right] + \sum_{i\neq j}\mathbb{E}\left[(d_i\alpha_i)^2(d_j\alpha_j)^2\right] \tag{30}$$

$$= \sum_{i=1}^d \mathbb{E}\left[d_i\right]\alpha_i^4 + \sum_{i\neq j}\mathbb{E}[d_i]\mathbb{E}[d_j]\alpha_i^2\alpha_j^2. \tag{31}$$

This concludes the proof. □

Next, we study the change in statistics after multiplication with a random matrix.

Lemma 16 (Statistics after multiplication with a random matrix)**.** *Let* $\mathbf{W}$ *be an iid random matrix, with zero mean entries that have variance* σ^2 *and kurtosis* κ*. Let* $\boldsymbol{\xi}\in\mathbb{R}^d$ *be an arbitrary random vector (not necessarily symmetric or with uncorrelated squared entries). Then*

$$\mathbb{E}\|\mathbf{W}\boldsymbol{\xi}\|_2^2 = d\sigma^2\mathbb{E}\|\boldsymbol{\xi}\|^2; \tag{32}$$

$$\mathbb{E}\|\mathbf{W}\boldsymbol{\xi}\|_2^4 = d(d+2)\sigma^4\mathbb{E}\|\boldsymbol{\xi}\|_2^4 + (\kappa-3)d\sigma^4\mathbb{E}\|\boldsymbol{\xi}\|_4^4; \tag{33}$$

$$\mathbb{E}\|\mathbf{W}\boldsymbol{\xi}\|_4^4 = 3d\sigma^4\mathbb{E}\|\boldsymbol{\xi}\|_2^4 + (\kappa-3)d\sigma^4\mathbb{E}\|\boldsymbol{\xi}\|_4^4. \tag{34}$$

Proof. The first property is easy to show:

$$\mathbb{E}\|\mathbf{W}\boldsymbol{\xi}\|_2^2 = \mathbb{E}\sum_r\left(\sum_{s,u} w_{rs}\xi_s w_{ru}\xi_u\right) = \mathbb{E}\sum_r\left(\sum_s w_{rs}^2\xi_s^2\right) = d\sigma^2\mathbb{E}\|\boldsymbol{\xi}\|^2. \tag{35}$$

The second and the third properties need computations.

$$\|\mathbf{W}\boldsymbol{\xi}\|_2^4 = \Big(\sum_i\Big(\sum_r w_{ir}\xi_r\Big)^2\Big)^2 \tag{36}$$

$$= \sum_{i,j}\sum_r w_{ir}\xi_r\sum_s w_{is}\xi_s\sum_u w_{ju}\xi_u\sum_v w_{jv}\xi_v, \tag{37}$$

$$\|\mathbf{W}\boldsymbol{\xi}\|_4^4 = \sum_i\Big(\sum_r w_{ir}\xi_r\Big)^4 \tag{38}$$

$$= \sum_i\sum_r w_{ir}\xi_r\sum_s w_{is}\xi_s\sum_u w_{iu}\xi_u\sum_v w_{iv}\xi_v. \tag{39}$$

Taking expectations, yields

$$\frac{\mathbb{E}\|\mathbf{W}\boldsymbol{\xi}\|_4^4}{\sigma^4} = \underbrace{\kappa\sum_i}_{\kappa d}\mathbb{E}\|\boldsymbol{\xi}\|_4^4 + \underbrace{3\sum_i}_{3d}\underbrace{\sum_{r\neq s}\mathbb{E}\left[\xi_r^2\xi_s^2\right]}_{\mathbb{E}\|\boldsymbol{\xi}\|_2^4-\mathbb{E}\|\boldsymbol{\xi}\|_4^4} = 3d\,\mathbb{E}\|\boldsymbol{\xi}\|_2^4 + (\kappa-3)d\,\mathbb{E}\|\boldsymbol{\xi}\|_4^4; \tag{40}$$

$$\frac{\mathbb{E}\|\mathbf{W}\boldsymbol{\xi}\|_2^4}{\sigma^4} = \underbrace{\sum_{i\neq j}}_{d(d-1)}\underbrace{\mathbb{E}\left[\Big(\sum_{r=s}\xi_r^2\Big)\Big(\sum_{u=v}\xi_u^2\Big)\right]}_{\mathbb{E}\|\boldsymbol{\xi}\|_2^4} + \underbrace{3\sum_{i=j}}_{3d}\underbrace{\sum_{r\neq s}\mathbb{E}\left[\xi_r^2\xi_s^2\right]}_{\mathbb{E}\|\boldsymbol{\xi}\|_2^4-\mathbb{E}\|\boldsymbol{\xi}\|_4^4} + \underbrace{\kappa\sum_i}_{\kappa d}\underbrace{\sum_{r=s}\mathbb{E}[\xi_r^4]}_{\mathbb{E}\|\boldsymbol{\xi}\|_4^4} \tag{41}$$

$$= d(d+2)\mathbb{E}\|\boldsymbol{\xi}\|_2^4 + (\kappa-3)d\,\mathbb{E}\|\boldsymbol{\xi}\|_4^4.$$

The factor 3 appears because, if 4 indices are paired in groups of two, we have a total of 3 disjoint choices: $\{i=j,u=v\}$, $\{i=u,j=v\}$, $\{i=v,j=u\}$. □

Corollary 17 (Preactivations after one (ReLU) layer)**.** *Let* $\mathbf{x}$ *be a symmetric random vector with uncorrelated squared entries and* $\mathbf{W}$ *be an iid random matrix, entries having mean zero, variance* σ^2 *and kurtosis* κ*. Then, the following formulas hold:*

$$\mathbb{E}\|\mathbf{W}\mathbf{D}_{\mathbf{x}}\mathbf{x}\|_2^2 = d\sigma^2 p\mathbb{E}\|\mathbf{x}\|_2^2; \tag{42}$$

$$\mathbb{E}\|\mathbf{W}\mathbf{D}_{\mathbf{x}}\mathbf{x}\|_2^4 = \left[d(d+2)\sigma^4p^2\right]\mathbb{E}\|\mathbf{x}\|_2^4 + \left[d\sigma^4p\left(\kappa-3+(1-p)(d+2)\right)\right]\mathbb{E}\|\mathbf{x}\|_4^4; \tag{43}$$

$$\mathbb{E}\|\mathbf{W}\mathbf{D}_{\mathbf{x}}\mathbf{x}\|_4^4 = \left[3d\sigma^4p^2\right]\mathbb{E}\|\mathbf{x}\|_2^4 + \left[d\sigma^4p(\kappa-3p)\right]\mathbb{E}\|\mathbf{x}\|_4^4. \tag{44}$$

Proof. Follows directly from Lemma 16 by plugging in $\boldsymbol{\xi} = \mathbf{D}_{\mathbf{x}}\mathbf{x}$. The effect of the potential ReLU is then integrated out using Lemma 14. □

B.3 Forward pass

The next theorem, which is the fundamental tool for our analysis, is checked numerically in Figure 22.

Theorem 18 (Forward pass statistics)**.** *Let* $\mathbf{z} = \mathbf{A}\mathbf{x}$*, which is, for* $\mathbf{x}$ *fixed, symmetrically distributed and with uncorrelated squared entries. Then,*

$$\mathbb{E}\|\mathbf{W}_\phi^{k:1}\mathbf{z}\|_2^2 = (d\sigma^2p)^k\mathbb{E}\|\mathbf{z}\|_2^2; \tag{45}$$

$$\begin{pmatrix}\mathbb{E}\|\mathbf{W}_\phi^{k:1}\mathbf{z}\|_2^4\\ \mathbb{E}\|\mathbf{W}_\phi^{k:1}\mathbf{z}\|_4^4\end{pmatrix} = \left(p^2d\sigma^4\right)^k\mathbf{Q}^k\begin{pmatrix}\mathbb{E}\|\mathbf{z}\|_2^4\\ \mathbb{E}\|\mathbf{z}\|_4^4\end{pmatrix}, \quad \mathbf{Q} := \begin{pmatrix}d+2 & \frac{\kappa-3+(1-p)(d+2)}{p}\\ 3 & \frac{\kappa-3p}{p}\end{pmatrix}. \tag{46}$$

Proof. From Lemma 13 we know that at initialization the entries of layer preactivations are symmetrically distributed and with uncorrelated squared entries. Hence, we can apply Corollary 17 recursively to preactivations.

$$\mathbb{E}\|\mathbf{W}_\phi^{k:1}\mathbf{z}\|_2^2 = d\sigma^2p\mathbb{E}\|\mathbf{W}_\phi^{k-1:1}\mathbf{z}\|_2^2; \tag{47}$$

$$\mathbb{E}\|\mathbf{W}_\phi^{k:1}\mathbf{z}\|_2^4 = \left[d(d+2)\sigma^4p^2\right]\mathbb{E}\|\mathbf{W}_\phi^{k-1:1}\mathbf{z}\|_2^4 \tag{48}$$

$$+\left[d\sigma^4p\left(\kappa-3+(1-p)(d+2)\right)\right]\mathbb{E}\|\mathbf{W}_\phi^{k-1:1}\mathbf{z}\|_4^4; \tag{49}$$

$$\mathbb{E}\|\mathbf{W}_\phi^{k:1}\mathbf{z}\|_4^4 = \left[3d\sigma^4p^2\right]\mathbb{E}\|\mathbf{W}_\phi^{k-1:1}\mathbf{z}\|_2^4 + \left[d\sigma^4p(\kappa-3p)\right]\mathbb{E}\|\mathbf{W}_\phi^{k-1:1}\mathbf{z}\|_4^4. \tag{50}$$

The formula for $\mathbb{E}\|\mathbf{W}_\phi^{k:1}\mathbf{z}\|_2^2$ directly follows, while the other two statistics evolve as a coupled dynamical system. □

Simple take away. Consider a Gaussian initialization $\kappa = 3$ in linear neural networks ($p = 1$). Then the recursion matrix $\mathbf{Q}$ in Eq.(46) simplifies to

$$\mathbf{Q} := \begin{pmatrix}d+2 & 0\\ 3 & 0\end{pmatrix}, \tag{51}$$

from which follows that $\mathbb{E}\|\mathbf{W}_\phi^{k:1}\mathbf{z}\|_2^4 = \left(d\sigma^4\right)^k(d+2)^k\,\mathbb{E}\|\mathbf{z}\|_4^2$. Hence, similar to the case of the neural chain depicted in Eq. 2 the pair of moments $\mathbb{E}\|\mathbf{W}_\phi^{k:1}\mathbf{A}\mathbf{z}\|_2^2$ and $\mathbb{E}\|\mathbf{W}_\phi^{k:1}\mathbf{z}\|_2^4$ cannot be stabilized jointly. Hence, as shown in Figure 21, the mean is allowed to be very different from the median by Mallows inequality [Mallows and Richter, 1969] — see also Thm. 9. Finally, we note that the effect we just discussed grows stronger if $d \ll L$. This is also confirmed by the simulation in Figure 22.

Corollary 19 (Asymptotic forward pass statistics)**.** *In the context of Theorem 18, as* $d \to \infty$

$$\mathbb{E}\|\mathbf{W}_\phi^{k:1}\mathbf{A}\mathbf{x}\|_2^4 \lesssim (d\sigma^2p)^{2k}, \tag{52}$$

$$\mathbb{E}\|\mathbf{W}_\phi^{k:1}\mathbf{A}\mathbf{x}\|_4^4 \lesssim (d\sigma^2p)^{2k}, \tag{53}$$

where "$\lesssim$*" denotes "asymptotically less of equal than a multiple of" (same as* $\mathcal{O}$*).*

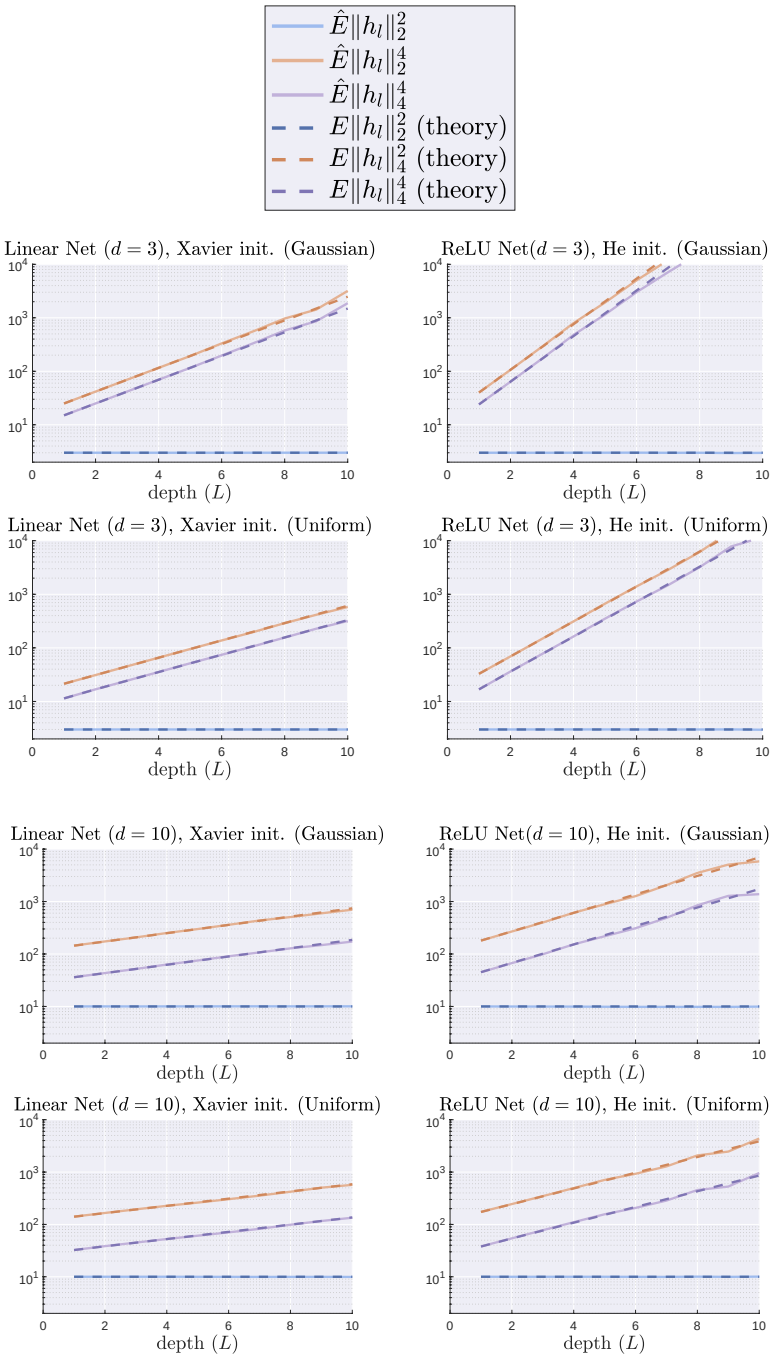


Figure 22: Numerical validation of Theorem 18 using the classical stabilizing initializations [Glorot and Bengio, 2010, He et al., 2015]. The variance of the weights is set to $1/\sqrt{d}$ for linear nets and $2/\sqrt{d}$ for ReLU nets (here, we used $d = 3, 10$). We consider a random Gaussian input and initialization of the weights with either Gaussian or uniform distribution. The theory matches the experiment (empirical mean denoted as $\hat{\mathbb{E}}$ — $1e5$ runs for $d = 10$, $1e7$ runs for $d = 3$). The results for the two initializations are similar, yet Gaussian case explodes a bit faster due to the effect of the kurtosis, which for the Gaussian is 3 while for the uniform is $3 - \frac{6}{5}$. The formula in Thm 18 also perfectly predicts this tiny shift in the population quantities, *confirming the correctness of our calculations.*

Proof. For the linear and the ReLU case we respectively have

$$\mathbf{Q}_{\text{linear}} = \begin{pmatrix} d+2 & \kappa-3 \\ 3 & \kappa-3 \end{pmatrix}, \quad \mathbf{Q}_{\text{ReLU}} = \begin{pmatrix} d+2 & d+2\kappa-4 \\ 3 & 2\kappa-3 \end{pmatrix}. \tag{54}$$

As $d \to \infty$, these matrices behave as follows:

$$\mathbf{Q}^{\infty}_{\text{linear}} = \begin{pmatrix} \mathcal{O}(d) & \mathcal{O}(1) \\ \mathcal{O}(1) & \mathcal{O}(1) \end{pmatrix}, \quad \mathbf{Q}^{\infty}_{\text{ReLU}} = \begin{pmatrix} \mathcal{O}(d) & \mathcal{O}(d) \\ \mathcal{O}(1) & \mathcal{O}(1) \end{pmatrix}. \tag{55}$$

The result follows. □

Proposition 20 (Forward pass form a different vector)**.** *Assume that, instead of* $\mathbf{x}$*, we feed into the random net (with activation gates computed using* $\mathbf{x}$*) a different vector* $\boldsymbol{\alpha}$*. Then, all the forward pass statistics still hold.*

Proof. Direct consequence of Corollary 15, when applied iteratively as done above. □

Statistics for network-dependent matrices from the forward pass

Proposition 21 (From network matrices to propagations of canonical vectors)**.** *Let* $\mathbf{e}_1$ *be the first vector in the canonical basis. We have*

$$\mathbb{E}\|\mathbf{W}_{\phi}^{\ell:k+1}\|_F^2 = d\ \mathbb{E}\|\mathbf{W}_{\phi}^{\ell:k+1}\mathbf{e}_1\|_2^2 \sim (d\sigma^2 p)^{\ell-k}; \tag{56}$$

$$\mathbb{E}\|\mathbf{W}_{\phi}^{\ell:k+1}\|_F^4 = d^2\ \mathbb{E}\|\mathbf{W}_{\phi}^{\ell:k+1}\mathbf{e}_1\|_2^4 \sim (d\sigma^2 p)^{2(\ell-k)}. \tag{57}$$

where "$\lesssim$*" denotes "asymptotically less of equal than a multiple of" (same as* $\mathcal{O}$*).*

Proof. By direct calculation

$$\mathbb{E}\|\mathbf{W}_{\phi}^{\ell:k+1}\|_F^2 = \mathbb{E}\left[\sum_{i=1}^{d}\|\mathbf{W}_{\phi}^{\ell:k+1}\mathbf{e}_i\|_2^2\right] = d\ \mathbb{E}\|\mathbf{W}_{\phi}^{\ell:k+1}\mathbf{e}_1\|_2^2 \sim (d\sigma^2 p)^{\ell-k}. \tag{58}$$

The second last equality follows from the isotropic structure of $\mathbf{W}_{\phi}^{\ell:k}$ and the last from Prop. 20.

$$\mathbb{E}\|\mathbf{W}_{\phi}^{\ell:k+1}\|_F^4 = \mathbb{E}\left[\left(\sum_{i=1}^{d}\|\mathbf{W}_{\phi}^{\ell:k+1}\mathbf{e}_i\|_2^2\right)^2\right] = \mathbb{E}\left[\sum_{ij}\|\mathbf{W}_{\phi}^{\ell:k+1}\mathbf{e}_i\|_2^2\|\mathbf{W}_{\phi}^{\ell:k+1}\mathbf{e}_j\|_2^2\right]. \tag{59}$$

Since $\|\mathbf{W}_{\phi}^{\ell:k+1}\mathbf{e}_j\|_2^2 = \|\mathbf{W}_{\phi}^{\ell:k+1}\mathbf{e}_i\|_2^2$ in distribution, $\mathbb{E}\|\mathbf{W}_{\phi}^{\ell:k+1}\|_F^4 = d^2\mathbb{E}\|\mathbf{W}_{\phi}^{\ell:k+1}\mathbf{e}_i\|_2^4$. The asymptotic statements hold thanks to Proposition 20. □

B.4 Gradient

We consider the loss

$$\mathcal{L}_{\mathbf{x},\mathbf{y}}(\mathbf{W}) = \frac{1}{2}\|\mathbf{y} - \mathbf{B}\mathbf{D}^L\mathbf{W}_{\phi}^{L:1}\mathbf{A}\mathbf{x}\|^2. \tag{60}$$

As in [Allen-Zhu et al., 2019], by noting $\tilde{\mathbf{B}} := \mathbf{B}\mathbf{D}^L$ and $\mathbf{z} = \mathbf{A}\mathbf{x}$ we get

$$\frac{\partial\mathcal{L}}{\partial\mathbf{W}^k} = \mathbf{W}_{\phi}^{k+1:L}\tilde{\mathbf{B}}^\top[\tilde{\mathbf{B}}\mathbf{W}_{\phi}^{L:1}\mathbf{z} - \mathbf{y}]\mathbf{z}^\top\mathbf{W}^{1:k-1} \tag{61}$$

$$= \underbrace{\mathbf{W}_{\phi}^{k+1:L}\tilde{\mathbf{B}}^\top\tilde{\mathbf{B}}\mathbf{W}_{\phi}^{L:1}\mathbf{z}\mathbf{z}^\top\mathbf{W}^{1:k-1}}_{\partial\mathcal{L}_k^1} - \underbrace{\mathbf{W}_{\phi}^{k+1:L}\tilde{\mathbf{B}}^\top\mathbf{y}\mathbf{z}^\top\mathbf{W}^{1:k-1}}_{\partial\mathcal{L}_k^2}, \tag{62}$$

with $\mathbf{W}_\phi^{k+1:L} := \left(\mathbf{W}_\phi^{L:k+1}\right)^\top$. By the triangle inequality, we have

$$\left\| \frac{\partial \mathcal{L}}{\partial \mathbf{W}^k} \right\|_F \leq \|\boldsymbol{\partial} \mathcal{L}_k^1\|_F + \|\boldsymbol{\partial} \mathcal{L}_k^2\|_F, \tag{63}$$

therefore we can bound each term individually. Let $\mathbb{E}^p[\cdot]$ be the p-th power of $\mathbb{E}[\cdot]$.

Proposition 22 (Bounding gradients with forward passes)**.** *We have*

$$\mathbb{E}\|\boldsymbol{\partial} \mathcal{L}_k^1\|_F \leq \mathbb{E}^{1/2}\left[\|\tilde{\mathbf{B}}\mathbf{W}_\phi^{L:k+1}\|_2^2\right] \mathbb{E}^{1/4}\left[\|\tilde{\mathbf{B}}\mathbf{W}_\phi^{L:1}\mathbf{z}\|_2^4\right] \mathbb{E}^{1/4}\left[\|\mathbf{W}_\phi^{k-1:1}\mathbf{z}\|_2^4\right]; \tag{64}$$

$$\mathbb{E}\|\boldsymbol{\partial} \mathcal{L}_k^2\|_F \leq \mathbb{E}\left[\|\mathbf{y}^\top\tilde{\mathbf{B}}\mathbf{W}_\phi^{L:k+1}\|_2^2\right] \mathbb{E}\left[\|\mathbf{W}_\phi^{k-1:1}\mathbf{z}\|_2^2\right]. \tag{65}$$

Proof. The bounds follow from submultiplicativity of Frobenius norm and Cauchy-Schwarz inequality — applied possibly twice.

$$\begin{aligned}
\mathbb{E}\|\boldsymbol{\partial} \mathcal{L}_k^1\|_F &= \mathbb{E}\|\mathbf{W}_\phi^{k+1:L}\tilde{\mathbf{B}}^\top\tilde{\mathbf{B}}\mathbf{W}_\phi^{L:1}\mathbf{z}\mathbf{z}^\top\mathbf{W}_\phi^{1:k-1}\|_F && (66)\\
&\leq \mathbb{E}\left[\|\mathbf{W}_\phi^{k+1:L}\tilde{\mathbf{B}}^\top\|_F\|\tilde{\mathbf{B}}\mathbf{W}_\phi^{L:1}\mathbf{z}\|_F\|\mathbf{z}^\top\mathbf{W}_\phi^{1:k-1}\|_F\right] && (67)\\
&= \mathbb{E}\left[\|\tilde{\mathbf{B}}\mathbf{W}_\phi^{L:k+1}\|_F\|\tilde{\mathbf{B}}\mathbf{W}_\phi^{L:1}\mathbf{z}\|_F\|\mathbf{W}_\phi^{k-1:1}\mathbf{z}\|_F\right] && (68)\\
&\leq \mathbb{E}^{1/2}\left[\|\tilde{\mathbf{B}}\mathbf{W}_\phi^{L:k+1}\|_F^2\right] \mathbb{E}^{1/2}\left[\|\tilde{\mathbf{B}}\mathbf{W}_\phi^{L:1}\mathbf{z}\|_F^2\|\mathbf{W}_\phi^{k-1:1}\mathbf{z}\|_F^2\right] && (69)\\
&\leq \mathbb{E}^{1/2}\left[\|\tilde{\mathbf{B}}\mathbf{W}_\phi^{L:k+1}\|_F^2\right] \mathbb{E}^{1/4}\left[\|\tilde{\mathbf{B}}\mathbf{W}_\phi^{L:1}\mathbf{z}\|_F^4\right] \mathbb{E}^{1/4}\left[\|\mathbf{W}_\phi^{k-1:1}\mathbf{z}\|_F^4\right]. && (70)
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}\|\boldsymbol{\partial} \mathcal{L}_k^2\|_F &= \mathbb{E}\|\mathbf{W}_\phi^{k+1:L}\tilde{\mathbf{B}}^\top\mathbf{y}\mathbf{z}^\top\mathbf{W}_\phi^{1:k-1}\|_F && (71)\\
&\leq \mathbb{E}\left[\|\mathbf{W}_\phi^{k+1:L}\tilde{\mathbf{B}}^\top\mathbf{y}\|_F\|\mathbf{z}^\top\mathbf{W}_\phi^{1:k-1}\|_F\right] && (72)\\
&\leq \mathbb{E}^{1/2}\left[\|\mathbf{W}_\phi^{k+1:L}\tilde{\mathbf{B}}^\top\mathbf{y}\|_F^2\right] \mathbb{E}^{1/2}\left[\|\mathbf{z}^\top\mathbf{W}_\phi^{1:k-1}\|_F^2\right] && (73)\\
&= \mathbb{E}^{1/2}\left[\|\mathbf{y}^\top\tilde{\mathbf{B}}\mathbf{W}_\phi^{L:k+1}\|_F^2\right] \mathbb{E}^{1/2}\left[\|\mathbf{W}_\phi^{k-1:1}\mathbf{z}\|_F^2\right]. && (74)
\end{aligned}$$

We conclude by noting that the Frobenius norm is the 2-norm for vectors. □

Proposition 23 (Bounding gradients with forward passes, wide net)**.** *As $d \to \infty$,*

$$\mathbb{E}\|\boldsymbol{\partial} \mathcal{L}_k^1\|_F \lesssim (p\sigma^2 d)^{\frac{2L-1}{2}}, \tag{75}$$

$$\mathbb{E}\|\boldsymbol{\partial} \mathcal{L}_k^2\|_F \lesssim (p\sigma^2 d)^{\frac{L-1}{2}}, \tag{76}$$

where "$\lesssim$" denotes "asymptotically less of equal than a multiple of" (same as $\mathcal{O}$). Hence, we have

$$\mathbb{E}\left\| \frac{\partial \mathcal{L}}{\partial \mathbf{W}^k} \right\|_F \lesssim (p\sigma^2 d)^{\frac{L-1}{2}} \qquad \text{if} \quad (p\sigma^2 d) \leq 1 \quad \textit{(vanishing-stable regime).} \tag{77}$$

$$\mathbb{E}\left\| \frac{\partial \mathcal{L}}{\partial \mathbf{W}^k} \right\|_F \lesssim (p\sigma^2 d)^{\frac{2L-1}{2}} \qquad \text{if} \quad (p\sigma^2 d) \geq 1 \quad \textit{(exploding regime).} \tag{78}$$

Proof. Simple application of Corollary 19 and Proposition 21 to the bounds in Proposition 22. □

B.5 Hessian

The Hessian of a linear DNN can be split into two block matrices, where each block has a Kronecker product structure. We can apply the product rule to the gradient and consider $\mathbf{W}^\ell$ and $\mathbf{W}^{\ell\top}$ ($\ell > k$) as two distinct

matrices: the block (k, ℓ) of the Hessian matrix is:

$$\underbrace{\frac{\partial^2 \mathcal{L}}{\partial \mathbf{W}^k \partial \mathbf{W}^\ell}}_{\mathbf{H}_1^{k\ell}} + \underbrace{\frac{\partial^2 \mathcal{L}}{\partial \mathbf{W}^k \partial \mathbf{W}^{\ell\top}}}_{\mathbf{H}_2^{k\ell}} \underbrace{\frac{\partial (\mathbf{W}^{\ell\top})}{\partial \mathbf{W}^\ell}}_{\mathbb{T}},$$

where $\mathbb{T}$ is the matrix transpose tensor. Recall that

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^k} = \mathbf{W}_\phi^{k+1:L} \tilde{\mathbf{B}}^\top [\tilde{\mathbf{B}} \mathbf{W}_\phi^{L:1} \mathbf{z} - \mathbf{y}] \mathbf{z}^\top \mathbf{W}^{1:k-1}. \tag{79}$$

By using the simple rule $\frac{\partial \mathbf{EWF}}{\partial \mathbf{W}} = \mathbf{F}^\top \otimes \mathbf{E}$, we get

$$\mathbf{H}_1^{k\ell} = \mathbf{W}_\phi^{k-1:1} \mathbf{z}\mathbf{z}^\top \mathbf{W}_\phi^{1:\ell-1} \ \otimes \ \mathbf{W}_\phi^{k+1:L} \tilde{\mathbf{B}}^\top \tilde{\mathbf{B}} \mathbf{W}_\phi^{L:\ell+1}; \tag{80}$$

$$\begin{aligned} \mathbf{H}_2^{k\ell} &= \mathbf{W}_\phi^{k-1:1} \mathbf{z} \left(\mathbf{z}^\top \mathbf{W}_\phi^{1:L} \tilde{\mathbf{B}}^\top - \mathbf{y}^\top \right) \tilde{\mathbf{B}} \mathbf{W}_\phi^{L:\ell+1} \otimes \mathbf{W}_\phi^{k+1:\ell-1} \\ &= \underbrace{\mathbf{W}_\phi^{k-1:1} \mathbf{z}\mathbf{z}^\top \mathbf{W}_\phi^{1:L} \tilde{\mathbf{B}}^\top \tilde{\mathbf{B}} \mathbf{W}_\phi^{L:\ell+1} \otimes \mathbf{W}_\phi^{k+1:\ell-1}}_{\mathbf{H}_{21}^{k\ell}} - \underbrace{\mathbf{W}_\phi^{k-1:1} \mathbf{z}\mathbf{y}^\top \tilde{\mathbf{B}} \mathbf{W}_\phi^{L:\ell+1} \otimes \mathbf{W}_\phi^{k+1:\ell-1}}_{\mathbf{H}_{22}^{k\ell}}, \end{aligned} \tag{81}$$

Note that if instead $k = \ell$, $\mathbf{H}^{kk} = \mathbf{H}_1^{kk}$.

Proposition 24 (Bounding the Hessian with forward passes)**.** *It is possible to bound the Hessian with statistics only on the forward pass.*

Proof. We simply apply the Cauchy-Schwarz inequality twice for each term, using also the Frobenius norm formula for the Kronecker product and norm submultiplicativity.

$$\mathbb{E}\|\mathbf{H}_1^{k\ell}\|_F = \mathbb{E} \left\| \mathbf{W}_\phi^{k-1:1} \mathbf{z}\mathbf{z}^\top \mathbf{W}_\phi^{1:\ell-1} \ \otimes \ \mathbf{W}_\phi^{k+1:L} \tilde{\mathbf{B}}^\top \tilde{\mathbf{B}} \mathbf{W}_\phi^{L:\ell+1} \right\|_F \tag{82}$$

$$\leq \mathbb{E}^{1/2} \left\| \mathbf{W}_\phi^{k-1:1} \mathbf{z}\mathbf{z}^\top \mathbf{W}_\phi^{1:\ell-1} \right\|_F^2 \mathbb{E}^{1/2} \left\| \mathbf{W}_\phi^{k+1:L} \tilde{\mathbf{B}}^\top \tilde{\mathbf{B}} \mathbf{W}_\phi^{L:\ell+1} \right\|_F^2 \tag{83}$$

$$\leq \mathbb{E}^{1/4} \left\| \mathbf{W}_\phi^{k-1:1} \mathbf{z} \right\|_F^4 \mathbb{E}^{1/4} \left\| \mathbf{W}_\phi^{\ell-1:1} \mathbf{z} \right\|_F^4 \mathbb{E}^{1/4} \left\| \tilde{\mathbf{B}} \mathbf{W}_\phi^{L:\ell+1} \right\|_F^4 \mathbb{E}^{1/4} \left\| \tilde{\mathbf{B}} \mathbf{W}_\phi^{L:k+1} \right\|_F^4. \tag{84}$$

$$\mathbb{E}\|\mathbf{H}_{22}^{k\ell}\|_F \leq \mathbb{E}^{1/2} \left\| \mathbf{W}_\phi^{k-1:1} \mathbf{z}\mathbf{y}^\top \tilde{\mathbf{B}} \mathbf{W}_\phi^{L:\ell+1} \right\|_F^2 \mathbb{E}^{1/2} \left\| \mathbf{W}_\phi^{k+1:\ell-1} \right\|_F^2 \tag{85}$$

$$\leq \mathbb{E}^{1/4} \left\| \mathbf{W}_\phi^{k-1:1} \mathbf{z} \right\|_F^4 \mathbb{E}^{1/4} \left\| \mathbf{y}^\top \tilde{\mathbf{B}} \mathbf{W}_\phi^{L:\ell+1} \right\|_F^4 \mathbb{E}^{1/2} \left\| \mathbf{W}_\phi^{\ell-1:k+1} \right\|_F^2. \tag{86}$$

$$\mathbb{E}\|\mathbf{H}_{21}^{k\ell}\|_F \leq \mathbb{E}^{1/2} \left\| \mathbf{W}_\phi^{k-1:1} \mathbf{z}\mathbf{z}^\top \mathbf{W}_\phi^{1:L} \tilde{\mathbf{B}}^\top \tilde{\mathbf{B}} \mathbf{W}_\phi^{L:\ell+1} \right\|_F^2 \mathbb{E}^{1/2} \left\| \mathbf{W}_\phi^{k+1:\ell-1} \right\|_F^2 \tag{87}$$

$$\leq \mathbb{E}^{1/4} \left\| \mathbf{W}_\phi^{k-1:1} \mathbf{z}\mathbf{z}^\top \mathbf{W}_\phi^{1:\ell} \mathbf{W}_\phi^{\ell+1:L} \tilde{\mathbf{B}}^\top \tilde{\mathbf{B}} \mathbf{W}_\phi^{L:\ell+1} \right\|_F^2 \mathbb{E}^{1/4} \left\| \mathbf{W}_\phi^{\ell-1:k+1} \right\|_F^2. \tag{88}$$

Note that the last term is not simplified completely, but unfortunately a simple iterated Cauchy-Schwarz splitting would lead to quantities with high exponents (eighth moment). Hence, we need to take a more complex approach. First, we split between terms which do not share weights.

$$\mathbb{E} \left[\left\| \mathbf{W}_\phi^{k-1:1} \mathbf{z}\mathbf{z}^\top \mathbf{W}_\phi^{1:\ell} \right\|_F^2 \left\| \mathbf{W}_\phi^{\ell+1:L} \tilde{\mathbf{B}}^\top \tilde{\mathbf{B}} \mathbf{W}_\phi^{L:\ell+1} \right\|_F^2 \right]. \tag{89}$$

Using the law of total expectation, the last expression becomes

$$\mathbb{E} \left[\ \mathbb{E} \left[\ \left\| \mathbf{W}_\phi^{k-1:1} \mathbf{z}\mathbf{z}^\top \mathbf{W}_\phi^{1:\ell} \right\|_F \left\| \mathbf{W}_\phi^{\ell+1:L} \tilde{\mathbf{B}}^\top \tilde{\mathbf{B}} \mathbf{W}_\phi^{L:\ell+1} \right\|_F^2 \ \middle| \ \mathcal{F}_\ell \ \right] \ \right], \tag{90}$$

where $\mathcal{F}_\ell$ is the information until layer ℓ. Using the same reasoning as Corollary 15 and Proposition 20, it is easy to realize that fixing the preactivation a separate integration of the second term in the product. In particular, the expression becomes

$$\mathbb{E}\left[\left\|\mathbf{W}_\phi^{k-1:1}\mathbf{z}\mathbf{z}^\top\mathbf{W}_\phi^{1:\ell}\right\|_F\right] \cdot \mathbb{E}\left[\left\|\mathbf{W}_\phi^{\ell+1:L}\tilde{\mathbf{B}}^\top\tilde{\mathbf{B}}\mathbf{W}_\phi^{L:\ell+1}\right\|_F^2 \;\middle|\; \mathcal{F}_\ell \right]. \tag{91}$$

As usual, we drop the filtration notation and plug this back into the expression for $\mathbb{E}\|\mathbf{H}_{21}^{k\ell}\|_F$:

$$\begin{aligned}\mathbb{E}\|\mathbf{H}_{21}^{k\ell}\|_F &\leq \mathbb{E}^{1/2}\left\|\mathbf{W}_\phi^{k-1:1}\mathbf{z}\mathbf{z}^\top\mathbf{W}_\phi^{1:\ell}\right\|_F^2 \mathbb{E}^{1/2}\left\|\mathbf{W}_\phi^{\ell+1:L}\tilde{\mathbf{B}}^\top\tilde{\mathbf{B}}\mathbf{W}_\phi^{L:\ell+1}\right\|_F^2 \mathbb{E}^{1/2}\left\|\mathbf{W}_\phi^{\ell-1:k+1}\right\|_F^2 \\ &\leq \mathbb{E}^{1/4}\left\|\mathbf{W}_\phi^{k-1:1}\mathbf{z}\right\|_F^4 \mathbb{E}^{1/4}\left\|\mathbf{W}_\phi^{\ell:1}\mathbf{z}\right\|_F^4 + \mathbb{E}^{1/2}\left\|\tilde{\mathbf{B}}\mathbf{W}_\phi^{L:\ell+1}\right\|_F^4 \mathbb{E}^{1/2}\left\|\mathbf{W}_\phi^{\ell-1:k+1}\right\|_F^2.\end{aligned}$$

□

Proposition 25 (Bounding Hessians with forward passes, wide net)**.** *As $d\to\infty$, for $k\neq\ell$*

$$\mathbb{E}\|\mathbf{H}^{k\ell}\|_F \lesssim (p\sigma^2 d)^{\frac{L-2}{2}} + (p\sigma^2 d)^{L-1}, \tag{92}$$

$$\mathbb{E}\|\mathbf{H}^{kk}\|_F \lesssim (p\sigma^2 d)^{L-1}, \tag{93}$$

where "$\lesssim$" denotes "asymptotically less of equal than a multiple of" (same as $\mathcal{O}$). Hence, we have

$$\begin{aligned}&\mathbb{E}\|\mathbf{H}^{k\ell}\|_F \lesssim (p\sigma^2 d)^{\frac{L-2}{2}}, \quad \mathbb{E}\|\mathbf{H}^{kk}\|_F \lesssim (p\sigma^2 d)^{L-1} \qquad \text{if} \quad (p\sigma^2 d)\leq 1 \quad \textit{(vanishing regime).}\\ &\mathbb{E}\|\mathbf{H}^{k\ell}\|_F \lesssim (p\sigma^2 d)^{L-1}, \quad \mathbb{E}\|\mathbf{H}^{kk}\|_F \lesssim (p\sigma^2 d)^{L-1} \qquad \text{if} \quad (p\sigma^2 d)\geq 1 \quad \textit{(exploding regime).}\end{aligned}$$

Proof. Follows from the triangle inequality. The only element left to bound is the transpose tensor $\mathbb{T}$, which however has only polynomial frobwnius norm in d and L. □

C Curvature Adaption of RMSProp

C.1 Literature review

Role of adaptive methods in modern-day deep learning. Adam and RMSprop (as well as explicitly regularized variants such as AdamW [Loshchilov and Hutter, 2017]) are known to perform extremely well compared to SGD when training attention models [Zhang et al., 2020, Liu et al., 2019, Wolf et al., 2020, Brown et al., 2020, Biggio et al., 2021], generative models [Karras et al., 2020a] and RNNs [Hochreiter and Schmidhuber, 1997]. In convolutional neural networks, many works [Loshchilov and Hutter, 2017, Balles and Hennig, 2018, Chen et al., 2020, Liu et al., 2019] show that slight variations of adaptive methods (e.g. AdamW) can close a suspected generalization gap [Wilson et al., 2017] and recently, [Tan and Le, 2019, Tan et al., 2019] achieved state of the art on ImageNet classification training with RMSprop (Adam with $\beta_1 = 0$). Finally, in the context of generative adversarial nets [Goodfellow et al., 2014], RMSprop is often preferred over Adam, leading to fast convergence [Park et al., 2019, Brock et al., 2018, Karras et al., 2020b], as opposed to SGD with momentum (see e.g. [Gemp and McWilliams, 2019] and references within).

The stochastic non-convex optimization approach to Adam. The first correct[7] proof of convergence for (a modified variant of) Adam in the stochastic nonconvex case was given in [Reddi et al., 2019], subject to a few assumptions (e.g. bounded gradients, decreasing stepsizes) and under the framework of online optimization. Perhaps the most recent simple, up-to-date, complete, and elegant proof of convergence of Adam was recently given in [Défossez et al., 2020], where the authors show that a rate $\mathcal{O}(\log(k)/\sqrt{k})$ can be achieved in expectation with iterate averaging yet no additional assumption on the maximum gradient norm (as opposed to most previous work). The same exact rate is also achieved by vanilla SGD [Ghadimi and Lan, 2013], which is well-known to be optimal for non-convex stochastic programming with bounded variance [Arjevani et al., 2019], if one does not rely on variance reduction (else, one can achieve slightly faster convergence [Cutkosky and Orabona, 2019]). Hence, it is clear that worst-case first-order complexity bounds which can be derived using the standard non-convex optimization methodology (at least for first order stationary points) are not yet able to explain the superiority of Adam in the context of optimization of deep neural networks. However, for Padam [Chen et al., 2020] a slight variation of Adam, it is possible to show a better dependency on the problem dimension, compared to (the known upper bound for) SGD.

Geometry adaptation, noise rescaling, gradient clipping and other conjectures. Some papers on Adam do not take the standard non-convex optimization approach discussed above. Chronologically, the first was [Balles and Hennig, 2018], that "dissects" Adam highlighting a variance-dependent preconditioning effect. This insight was taken one step further by [Staib et al., 2019], that shows how the variance-adaptation of *Adam effectively scales the gradient variance to be isotropic*; the authors claim this effect leads to fast escape from saddle points, since all eigendirections are equally excited by the stochastic perturbation.

Other papers take instead a geometric approach and motivate how Adam and RMSprop provide a cheap diagonal approximation of the empirical Fisher preconditioner, which is sometimes related to the Hessian [Martens, 2020]. As a result, *Adam can be thought of as an approximate Gauss-Newton method* [Nocedal and Wright, 2006]. While this interpretation could in principle explain the fast convergence of Adam, it was recently shown [Kunstner et al., 2019] that very often the Adam preconditioner can be very far away from the true Hessian, which is instead provably related to the non-empirical (a.k.a. true) Fisher. However, in [Dauphin et al., 2015], the authors show that RMSprop effectively regularizes the landscape, making it more "equilibrated" (well-conditioned).

Finally, [Zhang et al., 2019c] relate the success of adaptive methods to the underlying gradient clipping effect: if sporadic big stochastic gradients are encountered, those are smoothed out by the effect of the variables m and v. The authors are able to show that clipped SGD, under quite uncommon assumptions on the cost function, is able to slightly improve (by a constant factor) over the rate of SGD, in some particular cases. While this does provide a quantitative improvement on SGD (not the case e.g. for the rates in [Défossez et al., 2020]), the result is arguably not strong enough to motivate the success of adaptive methods.

[7] Quite interestingly, the original proof in [Kingma and Ba, 2014] contains a few mistakes due to the problematic non-monotonic decrease of the effective stepsize.

C.2 Behaviour of RMSprop on the chain

We empirically study the behavior of RMSprop on the chain loss with one data point $(x, y) = (1, 1)$:

$$\mathcal{L}_{\text{chain}}(\mathbf{w}) = \frac{1}{2}(1 - w_L w_{L-1} \cdots w_1)^2. \tag{94}$$

While this cost function is very simple, it showcases the adaptiveness of RMSprop in an extremely clean and concrete way. We consider $L = 10$ and initialize each weight to $w_i(0) \sim \mathcal{U}[-0.2, 0.2]$, to induce vanishing gradients (order 10^{-8}) and curvature (order order 10^{-7}), as it can be seen from Figure 23 and is predicted by Theorem 6 and Corollary 11. This initialization is close to $\mathbf{w} = \mathbf{0}$, which is clearly a saddle point because all partial derivatives vanish, but there exist directions of both increases and decreases: increasing all w_i simultaneously to $\epsilon > 0$ makes the loss decrease, while increasing half (i.e. five) of them to ϵ and decreasing the other half to $-\epsilon$ makes the loss increase.

We show results for 4 different seeds, 3 different noise injection levels and 3 different GD stepsizes. Results are shown in Figure 23& 24: while perturbed GD is slow to escape the saddle for any noise injection level and any stepsize, RMSprop is able to adapt to curvature and quickly escapes the saddle. This result validates the claim in Sec. 3 of the main paper (Prop. 7). In addition, Figure 25 shows that, if the initialization is such that the gradients at initialization are bigger, i.e. $w_i(0) \sim \mathcal{U}[-1, 1]$, then the performance of perturbed gradient descent get can get closer to the one of RMSprop.

Notation. We denote by $v(t)$ the exponential moving average (with parameter $\beta_2 = 0.9$) of the squared gradients at iteration t and by $\Lambda(t)$ the maximum Hessian eigenvalue at iteration t.

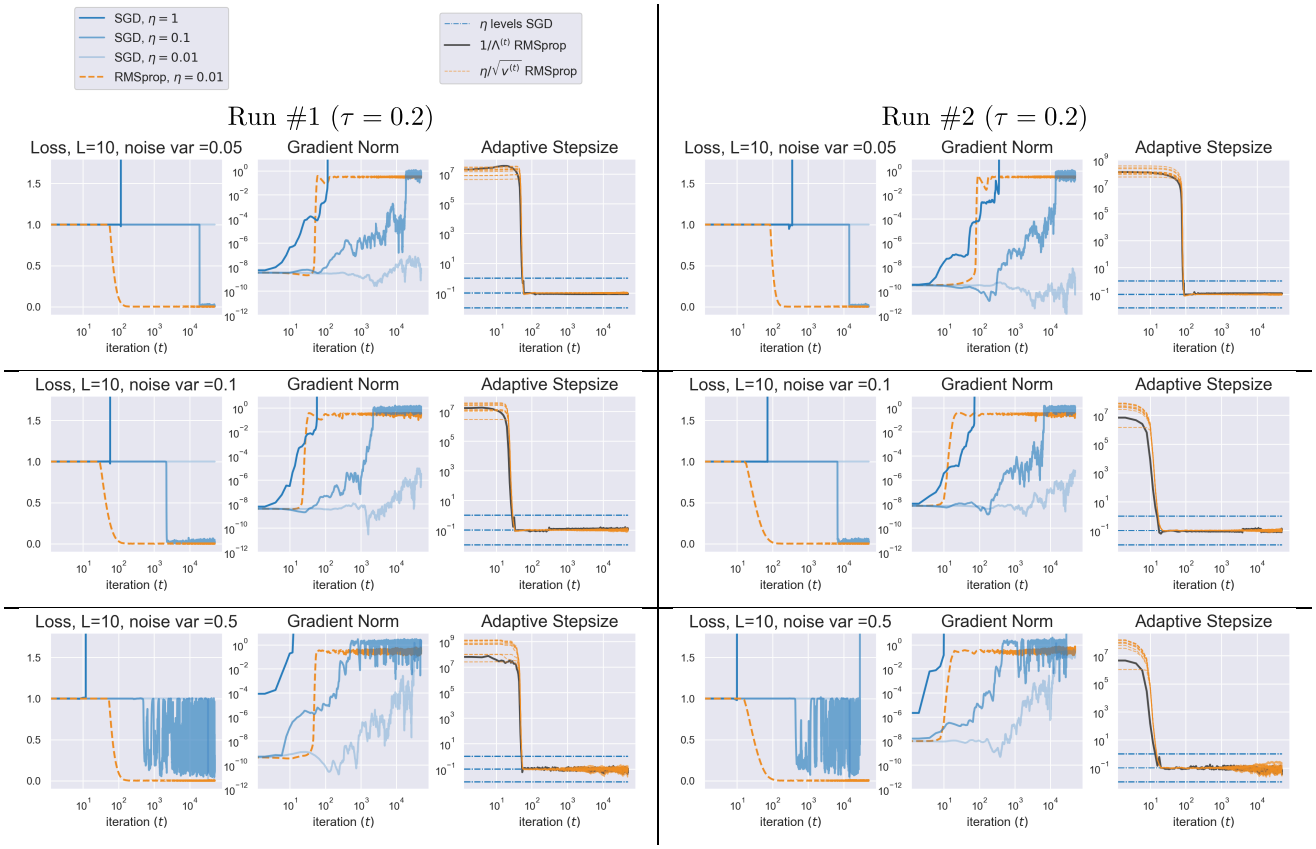


Figure 23: Optimization of a ten-dimensional chain, $w_i(0) \sim \mathcal{U}[-0.2, 0.2]$. Injected is an isotropic Gaussian noise of standard deviations $0.05, 0.1, 0.5$ (two additional runs in Fig. 24). While moderate noise helps GD (blue), no choice of stepsize is able to provide a performance comparable to RMSprop, which escapes after less than 100 iterations. Notably, the effective RMSprop stepsize matches the inverse curvature and is robust to noise.

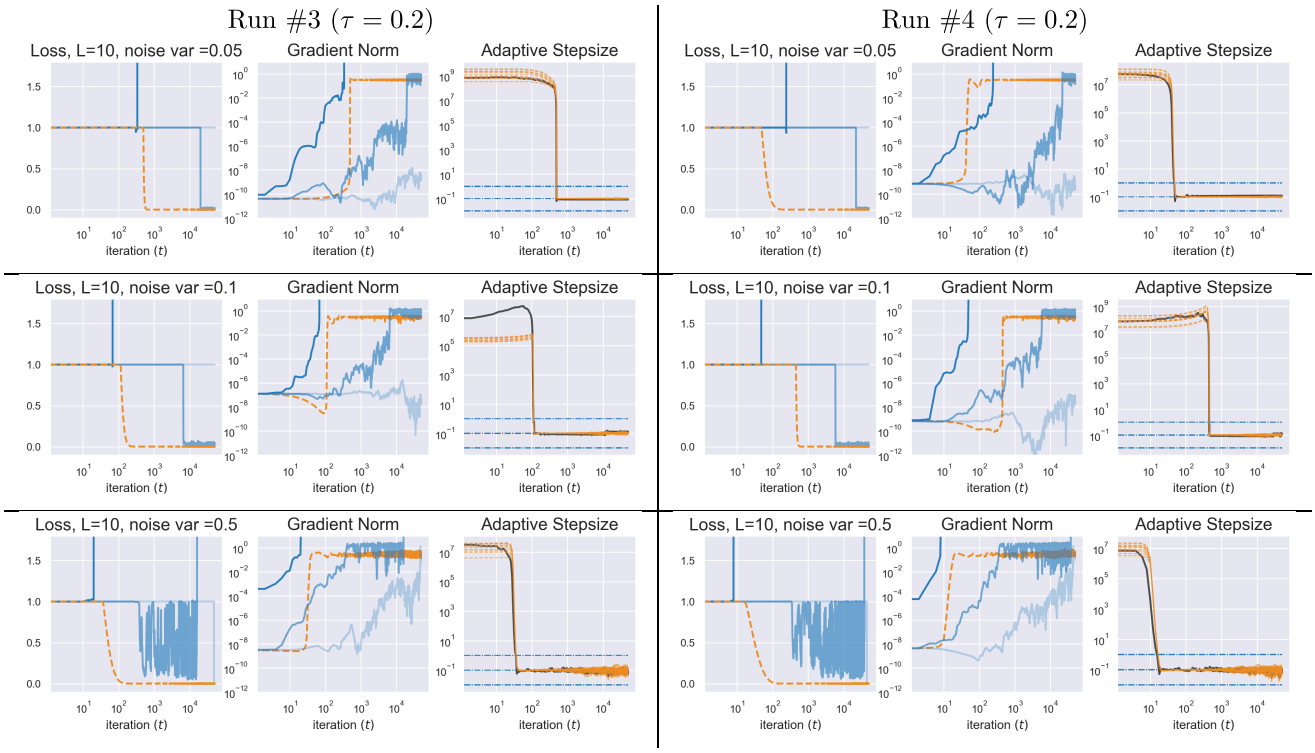


Figure 24: Two additional runs, same settings and legend as Figure 23.

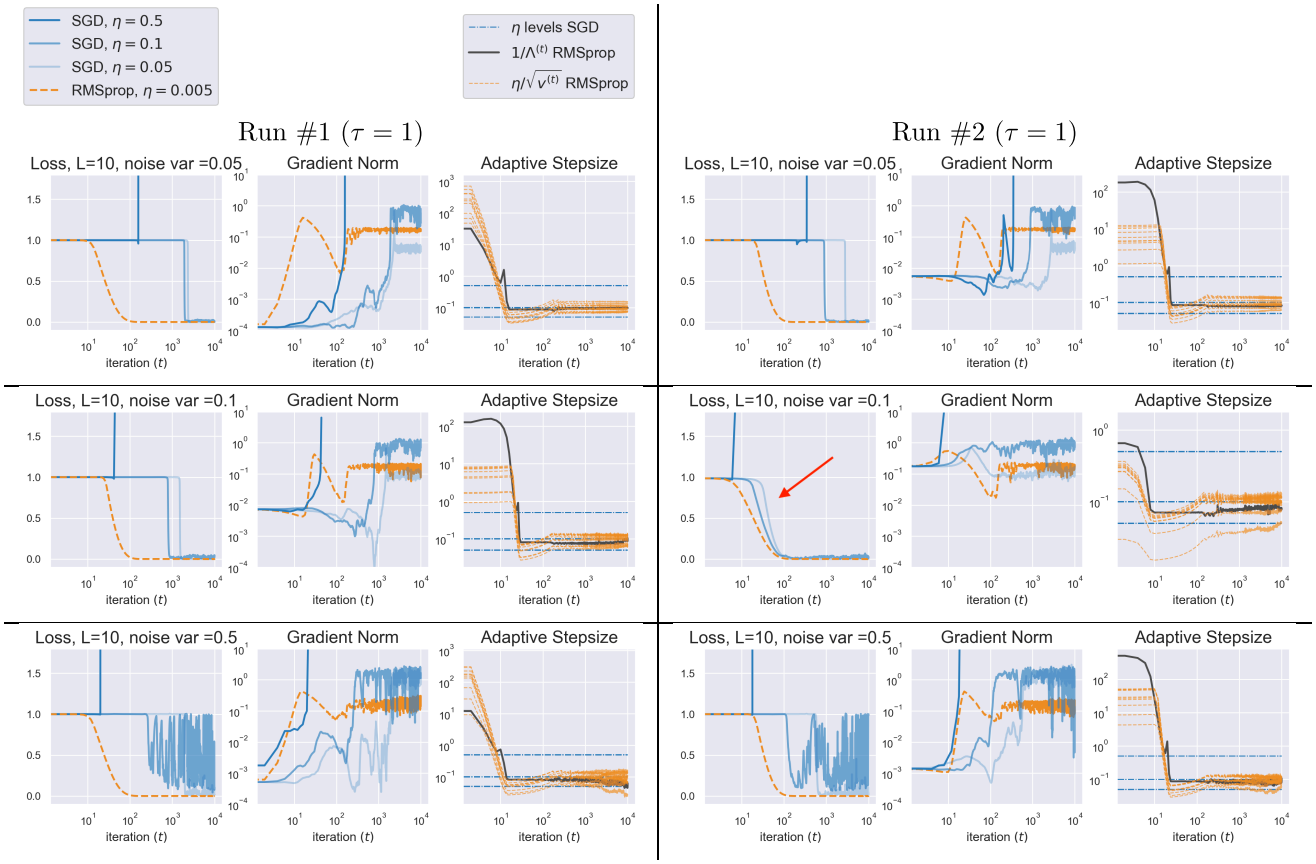


Figure 25: Here we instead consider $w_i(0) \sim \mathcal{U}[-1, 1]$, which induces less curvature-gradient vanishing compared to Fig. 23& 24. If the initial gradient norm is high, Perturbed gradient descent gets now closer in performance to RMSprop. However, the gradient/Hessian norm at initialization has a considerable variance, which makes the performance less predictable compared to Fig. 23& 24. This fact also makes the curvature adaptation feature of RMSprop less apparent.

D Paths in CNNs

Contrary to the fully connected architectures discussed above, convolutional neural networks are only sparsely connected and yield a large amount of weight sharing. It is thus not surprising to see that, given an MLP and a CNN of the same width to depth ratio (where the CNN width is defined as in Section 5, i.e. $d = k^2 c$), the CNN yields much smaller gradient (and Hessian) magnitudes than the MLP (compare Figure 4 and 8 when $d = \sqrt{L}$). This is so, because the paths going into each output neuron share most parameters and thus there is less probability for a path being atypically large.

The following Figure illustrates this for a CNN with one dimensional convolutions[8]. To be precise, we consider an input $x \in \mathbb{R}^3$, convolutional kernels $h^l \in \mathbb{R}^3$ and opt for circular padding. In this case one can write the CNN forward pass as $\mathbf{K}^L \dots \mathbf{K}^1 \mathbf{x}$ and the MLP forward pass as$\mathbf{W}^L \dots \mathbf{W}^1 \mathbf{x}$, where

$$\mathbf{K}^l = \begin{bmatrix} h_2^l & h_3^l & h_1^l \\ h_1^l & h_2^l & h_3^l \\ h_3^l & h_1^l & h_2^l \end{bmatrix} \quad \text{and} \quad \mathbf{W}^l = \begin{bmatrix} w_1^l & w_2^l & w_3^l \\ w_4^l & w_5^l & w_6^l \\ w_7^l & w_8^l & w_9^l \end{bmatrix}.$$

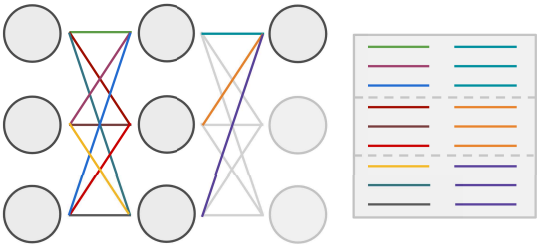


Figure 26: * MLP

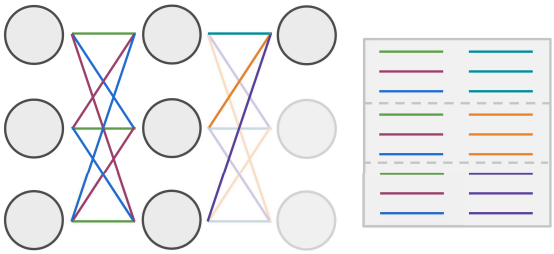


Figure 27: * CNN

Figure 28: Paths in MLP and CNN of same width.

As can be seen from the above figure, the number of i.i.d. weights w_i^l that contribute to an output neuron in the MLP grows *exponentially*, i.e. for each of the $d = 3$ weights in layer L there are another d weights contribution in layer $L - 1$ and so on. In the CNN case, this growth is (despite the existence of the same number of paths) only *additive* since all neuron in a given layer l share their incoming weights. As a result, it is no surprising to see that when comparing narrow CNNs to narrow MLPs, the vanishing gradients/curvature is much more pronounced in the former (compare Fig. 8 and 4).

[8]For simplicity of presentation, the logic directly generalized to higher dimensional convolutions