Deep Ensemble Uncertainty Fails as Network Width Increases: Why, and How to Fix It

Lisa Schut [1] **Edward Hu** [2] **Greg Yang** [*2] **Yarin Gal** [*1]

Abstract

Deep ensembles can be seen as an empirical approximation of the distribution over trained neural networks, and deep ensemble uncertainty can be seen as approximating the variance of the distribution. This variance can be estimated analytically in the infinite width limit as the variance of a Gaussian process. Yang & Hu (2021) show that this limit does not allow for feature learning, throwing away the advantages of deep learning, and suggest instead a new parametrization that allows for feature learning. This new parametrization trades uncertainty quality for feature learning, rendering it unsuitable for uncertainty quantification as models become wider. In this work, we propose a modification to the parametrization and the training procedure that allows for both feature learning and uncertainty quantification via ensembling. Our preliminary results on MNIST suggest that our findings hold empirically.

1. Introduction

The neural tangent kernel (NTK) is a useful analytical tool for understanding the training dynamics of neural networks. Several works have considered the impact of the initialization of neural networks or learning rate on training (He et al., 2020; Chizat & Bach, 2018; Woodworth et al., 2020; Yang & Hu, 2021). Recently, Yang & Hu (2021) have shown that standard NTK parametrization does not allow for *feature learning*.

Feature learning is crucial for obtaining a strong performance and task-specific uncertainty. Yang & Hu (2021) introduce a new parametrization, known as μ-parametrization (μP), that does allow for feature learning. Informally, the parametrization changes the effective learning rate of each layer in the neural network, such that the features (pre-activation layer outputs) can be updated maximally. However, a consequence is that the learnt function f has zero variance in the infinite width limit. In this case, we cannot use deep ensembles (Lakshminarayanan et al., 2017) to obtain uncertainty estimates in the infinite width limit.

In this work, we propose a novel parametrization that allows for learning features and a non-zero variance function f in the infinite limit. While we derive these results for the infinite limit case, we demonstrate the same results empirically for neural networks with finite widths. Therefore, our derivations may also be of interest for practitioners interested in the scaling behavior of models.

2. Background Work

Before diving into the different parametrizations, we first recapitulate abc-parametrization (Yang & Hu, 2021), which we use to define the network parametrization. An abc-parametrization is characterized by three constant hyperparameters: a_l, b_l and c_l. Let W^l be a weight matrix in a L-layer network. Then, $W^l := n^{-a_l} w_l$, where $w_l \sim N(0, n^{-2b_l})$ is a trainable parameter. The third parameter c_l is the learning rate, defined as γn^{-c_l}, where γ is a constant. We make a minor change to the original definition of abc-parametrization introduced in (Yang & Hu, 2021) and define a layer-specific learning rate c_l, rather than c. It will become evident why in Section 3.

Through abc-parametrization, we can create an effective per-layer learning rate.[1] This allows us to ensure that we can maximally update both features as well as the function f. Using the abc-parametrization, the standard neural tangent kernel parametrization (SP) and μ-parametrization (μP) are summarized in Table 1. SP corresponds to the standard Kaiming normal initialization (He et al., 2015) in Pytorch.

The initial parametrization affects the function training dynamics. Specifically, it influences the norm of the updates in the weights and thereby the norm of the updates in the

[*]Equal contribution [1]OATML, University of Oxford, United Kingdom [2]Microsoft Research. Correspondence to: Lisa Schut <schut@robots.ox.ac.uk>.

Presented at the ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning., Copyright 2021 by the author(s).

[1]Note that setting $W^l := n^{-a_l} w_l$, where $w_l \sim N(0, n^{-2b_l})$, is not the same as using $W^l := w_l$, where $w_l \sim N(0, n^{-2b_l - 2a_l})$. This will become more evident in the derivations, but we also encourage the reader to look into Yang & Hu (2021).

Table 1. abc-parametrization of SP, μP and our parametrization. Formally, the parametrization we introduce is not an abc-parametrization as we modify the backward pass in SGD (as shown in Algorithm 1) but we slightly abuse the definition here.

	a_l	b_l	c_l
SP	0	$\begin{cases} 0, l=1 \\ \frac{1}{2}, l \geq 2 \end{cases}$	1
μP	$\begin{cases} -\frac{1}{2}, l=1, \\ 0, 2 \leq l \leq L, \\ \frac{1}{2}, l=L+1 \end{cases}$	$\frac{1}{2}$	0
OURS	$\begin{cases} -\frac{1}{2}, l=1 \\ 0, l \geq 2 \end{cases}$	$\frac{1}{2}$	$\begin{cases} 0, l \leq L, \\ 1, l=L+1 \end{cases}$

features, and the variance over the features and functions for a given input (due to the stochasticity in the initial weights). In this paper, we will focus on two key aspects: *feature learning* and whether the learnt function is *deterministic*.

We start by defining both terms. Following Yang & Hu (2021), we define feature learning as follows:

Definition 1 *Let x_l be the network features, i.e. pre-activation layer outputs. Then, feature learning occurs if x_l have an update of $\Theta(1)$,*

where the big-O notation denotes the scaling with respect to the network width n. Further, following Yang & Hu (2021), we define the big-O notation as

Definition 2 *A vector v is $O(n^a)$ iff $\sqrt{||v||^2/n}$ fluctuates on the order of $O(n^a)$, where n is the number of units in a hidden layer.*

We state that a function f is deterministic iff

Definition 3 $\lim_{n\to\infty} var(f_t) \to 0$*, where n is the number of units in a hidden layer.*

Given these definitions, we can discuss how different parametrizations affect feature learning and whether the learnt function f is deterministic. In SP, the learning rate must scale with $\Theta(1/n)$ (i.e., $c_l = 1$), or else training will blow up in a wide network. With such a learning rate, we enter the kernel training regime and are not able to obtain feature learning in the infinite width limit. However, the learnt function f is not deterministic, and therefore generally permits uncertainty estimation via ensembling.[2]

μP permits feature learning by re-scaling the layers. However, due to the down-scaling of the final layer (i.e., $a_{L+1} = 1/2$), f has a variance on the order of $\Theta(1/n)$. A network initialized with μP results in a deterministic function in the infinite limit. Consequentially,

[2]The SP modification introduced by He et al. (2020), so that deep ensembles have a GP posterior equivalent, has the same shortcoming as SP – it does not allow for not feature learning in the infinite limit.

Table 2. Do the parametrizations allow for feature learning and uncertainty (via ensembling) in the infinite width limit?

	FEATURE LEARNING	UNCERTAINTY
SP	$\times$	$\surd$
μP	$\surd$	$\times$
OURS	$\surd$	$\surd$

Corollary 1 *Assuming a fixed data ordering during training, an infinite width network with μP does not permit uncertainty quantification using ensembling.*

This corollary follows directly from the fact that f is deterministic, and therefore in the infinite width limit, there is no variance in the function predictions.

3. Modified μP

In the previous section, we determined that SP and μP are able to either permit feature learning or result in a deterministic function f. Our goal is to propose an initialization scheme such that we maintain *maximal updates* ($\Theta(1)$ updates in the features and trained function) while avoiding learning a deterministic function f, as in μP (Yang & Hu, 2021).

As such, we propose:

- in general, use μP to ensure that we obtain **maximal feature and function updates** during training.
- contrary to μP, do not downscale the weights in the final layer (i.e., use $a_{L+1} = 1/2$), to **avoid learning a deterministic function**.
- modify the backward pass, as described in the pseudocode in Algorithm 1. The main difference with normal SGD is that we set W_t by $\Delta W_t = W_t - W_0$.
- use a learning rate of γn^{-1} for the final layer, and γ for other layers.

The last two alterations prevent the network from blowing up during training, while maintaining the benefits of μP. In depth derivations can be found in Appendix A. Here, we derive the first few steps of SGD, to provide insight to how the modifications may enable learning features and a non-deterministic function.

Algorithm 1 Modified SGD

Input: data point (ξ, y), model f parametrized by weights $\{W^l\}_{l=1}^{L+1}$
Initialize weights, i.e. sample $W_i^l \overset{\text{iid}}{\sim} N(0, n^{-1}), \forall l$.
Store a copy of the weights in the final layer W_0^{L+1}
for $t = 1$ to T **do**
  Compute the forward pass as normal, i.e. $\hat{y} = f(\xi)$
  Set $W_{t-1}^{L+1} \leftarrow W_{t-1}^{L+1} - W_0^{L+1}$
  for $l = 1$ to $L + 1$ **do**
    Update the weights, i.e. $W_t^l \leftarrow W_{t-1}^l - \delta_t \frac{df}{dW_{t-1}^l}$
  end for
  Set $W_t^{L+1} \leftarrow W_t^{L+1} + W_0^{L+1}$
end for

3.1. Notation

We assume a single layer MLP network with a one-dimensional input and output ($\xi, y \in \mathbb{R}$) for simplicity. The derivations can be easily extended for multi-dimensional inputs and layers. The derivations for two layer networks can be found in Appendix A.

Let us assume we have a neural network $f(\xi)$ defined as

$$f(\xi) = Vx(\xi)$$
$$x(\xi) = \phi(h(\xi))$$
$$h(\xi) = U\xi$$

where $V_0 := v_0 \in \mathbb{R}^{1\times n}, (v_0)_i \overset{\text{iid}}{\sim} N(0, 1/n)$ and $U_0 := \sqrt{n}u_0 \in \mathbb{R}^{n\times 1}, (u_0)_i \overset{\text{iid}}{\sim} N(0, 1/n)$ are weights. n denotes the number of hidden units per layer, the subscript 0 denotes the values at initialization, and the subscript i distinguishes between different weights at a given time step. In general, subscript t denotes the value at the t-th iteration (not to be confused with the superscript $\cdot^t$ which denotes transpose). Therefore V_t denotes all weights at time step t, and $(V_t)_i$ denotes the i-th weight at time step t.

Our goal in the following derivations is to show that both f_t and h_t have $\Theta(1)$ updates; therefore, neither explode nor are unable to learn in the infinite width limit. Further, we want to show that the variance of f_t does not collapse in the infinite width limit.

First forward pass The first forward pass for an input ξ_1 proceeds as normal:

$$h_1 = \sqrt{n}u_0\xi_1$$
$$x_1 = \phi(h_1)$$
$$f_1 = V_0x_1$$

By the central limit theorem, h_1, x_1 and f_1 are approximately normally distributed with variance of $\Theta(1)$. This is a key difference with μP, where f_1 is $\Theta(1/n)$. This is important as it allows for uncertainty estimation via ensembling.

First backward pass In the backward pass, we compute the updates with $\Delta v_t = v_t - v_0$, where v_0 is the value of v at initialization. Therefore, the updates are:

$$\begin{aligned} v_1 &= v_0 - n^{-1}(d\ell/df) \cdot (df/dv) \\ &= v_0 - n^{-1}\delta_1 x_1^t \qquad (1) \\ u_1 &= u_0 - (d\ell/df) \cdot (df/du) \\ &= u_0 - \delta_1(\sqrt{n}\Delta V_0^t \odot \phi'(h_1)\xi_1) \\ &= u_0 - \delta_1(\sqrt{n} \odot 0 \odot \phi'(h_1)\xi_1) \\ &= u_0, \qquad (2) \end{aligned}$$

where $\delta_1 := d\ell_1/df$ is the derivative of the loss function at time step 1. Here we observe a key difference with a normal backpass: u_0 is not updated because $\Delta v_0 = 0$.

Second forward pass

$$\begin{aligned} h_2 &= \sqrt{n}u_1\xi_2 = \sqrt{n}u_0\xi_2 \quad \text{by Eq. 2} \\ x_2 &= \phi(h_2) \\ f_2 &= V_1x_2 = V_0x_2 - n^{-1}\delta_1 x_1^t x_2 \quad \text{by Eq. 1} \end{aligned}$$

Here, the update in f is $\Theta(1)$ as δ_1 and $x_i, i = 1, 2$ are $\Theta(1)$. In general, the variance of f_2 is $\Theta(1)$ as f_0 has a $\Theta(1)$ variance. A trivial example of a dataset for when this is the case is for $\xi_i = 0, \forall i$. This is an important difference with μP, where we would observe a $\Theta(1/n)$ variance.

Second backward pass This step differs from the first backward pass as Δv_t is no longer zero, and therefore we have a different update. Concretely,

$$\begin{aligned} V_2 &= V_1 - n^{-1}(d\ell/df) \cdot (df/dV) \\ &= V_1 - n^{-1}\delta_2 x_2^t \qquad (3) \\ u_2 &= u_1 - \delta_2 df/du \\ &= u_1 - \delta_2(-\sqrt{n}\Delta V_1^t \odot \phi'(h_2)\xi_2) \\ &= u_1 + n^{-1/2}\delta_1\delta_2 x_1 \odot \phi'(h_2)\xi_2 \qquad (4) \end{aligned}$$

The update in u_0 is $\Theta(n^{-1/2})$.

Third forward pass

$$\begin{aligned} h_3 &= \sqrt{n}u_2\xi_3 \\ &= \sqrt{n}u_1\xi_3 + \delta_1\delta_2 x_1 \odot \phi'(h_2)\xi_2\xi_3 \quad \text{by Eq. 4} \\ x_3 &= \phi(h_3) \\ f_3 &= V_2x_3 \\ &= V_1x_3 - n^{-1}\delta_2 x_2^t x_3 \quad \text{by Eq. 3} \end{aligned}$$

As u was updated in the last backward pass, the distributions of h and x have changed. Importantly, we have feature learning. δ_1, δ_2, x_1, $\phi'(h_2)$, ξ_1, and ξ_2 are $\Theta(1)$. Therefore, the update in h_3 is $\Theta(1)$. As before, the update in f is $\Theta(1)$.

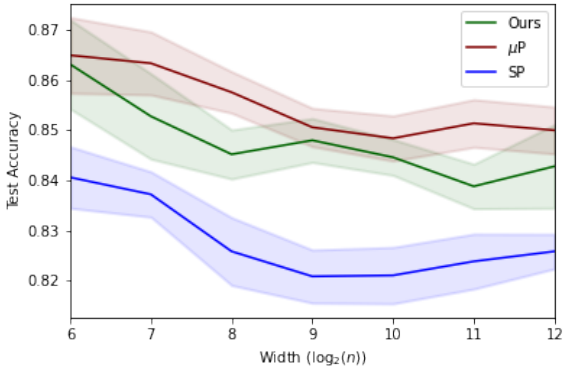

Figure 1. Feature learning: logistic regression performance using the top 15 principal components of features. The average over 10 seeds is reported. The shaded areas show the 95% confidence intervals.

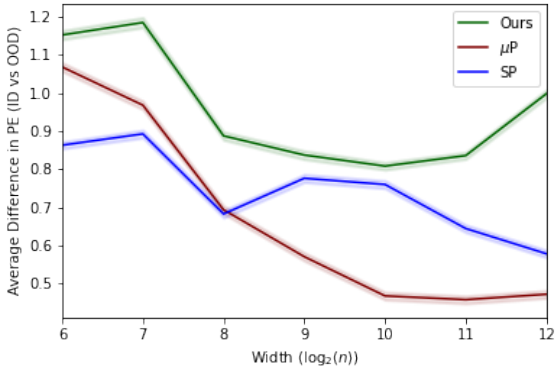

Figure 2. Average difference in predictive entropy for out-of-distribution data (FashionMNIST) versus in-distribution data (MNIST) as width increases for ensembles of neural networks with different parametrizations; higher is better. The average difference over $10,000$ examples is reported. The shaded areas show the 95% confidence intervals.

The above derivations provide some insight into the SGD updates in training a MLP. We have seen that [1] generally, $\text{var}(f_t)$ is $\Theta(1)$ and [2] the updates in the function f and features h are $\Theta(1)$. A more formal proof of this result can be derived by conditioning f on the initialization values, following the derivations Appendix D.1. in Yang & Littwin (2021).

4. Experiments

We verify our findings empirically on MNIST by evaluating the performance of MLPs of increasing widths on feature learning and uncertainty estimation. More extensive experiment details can be found in Appendix B.

To demonstrate feature learning, we extract the features (from the penultimate layer) and apply principal component analysis. We extract the top 15 principal components (where 15 was chosen using a scree plot) and use them as the input for a logistic classifier. We benchmark the performance of our parametrization against SP and μP.

Figure 1 shows the average accuracy of the logistic regression classification using the features extracted from neural networks of varying widths and parametrizations. Our findings for SP and μP for feature learning are in line with Yang & Hu (2021). We observe that as width increases, the performance of SP drops; this is expected as feature learning theoretically disappears in the infinite width limit. The performance of μP stays comparatively constant during training, which is expected as μP maintains feature learning in the infinite width limit. Our parametrization has a comparable (although for some widths, slightly worse) performance to μP. It performs significantly better than SP.

To evaluate uncertainty estimation performance, we estimate the average difference in predictive entropy between out-of-distribution data (FashionMNIST) and in-distribution data (MNIST). We estimate entropy by using an ensemble of 10 models.

Figure 2 shows the average difference in predictive entropy for in- and out-of-distribution data. A higher value is better. We observe that our parametrization is able to obtain better uncertainty estimation than the other parametrizations. As width increases, μP performs less well at distinguishing in-distribution versus out-of-distribution data using predictive entropy from ensembles. This is because the variance of f disappears as width increases. SP does relatively better, although we still observe a drop in performance.

5. Conclusion

We introduce a new neural network parametrization and training scheme, that allows for both feature learning and avoids function variance collapse in the infinite width limit. Our preliminary experiments show that the parametrization indeed improves uncertainty quantification for out-of-distribution detection. In future work, we hope to expand our analysis, including a more rigorous experimental evaluation to include different datasets and ablation studies. Further, we plan to further investigate some results, such as the observed dip in the performance of standard parametrization on uncertainty estimation as width increases.

6. Acknowledgements

L.S. is supported by EPSRC and DeepMind.

References

Chizat, L. and Bach, F. On the global convergence of gradient descent for over-parameterized models using optimal transport. In *Advances in neural information processing systems*, pp. 3036–3046, 2018.

He, B., Lakshminarayanan, B., and Teh, Y. W. Bayesian deep ensembles via the neural tangent kernel. *arXiv preprint arXiv:2007.05864*, 2020.

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

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles, 2017.

Woodworth, B., Gunasekar, S., Lee, J. D., Moroshko, E., Savarese, P., Golan, I., Soudry, D., and Srebro, N. Kernel and rich regimes in overparametrized models. *arXiv preprint arXiv:2002.09277*, 2020.

Yang, G. and Hu, E. J. Feature learning in infinite-width neural networks, 2021.

Yang, G. and Littwin, E. Tensor programs iib: Architectural universality of neural tangent kernel training dynamics. *arXiv preprint arXiv:2105.03703*, 2021.

A. Derivations

In the following section, we derive the first couple of steps for SGD. The main idea is to provide some intuition on how the parametrization scheme affects the network training dynamics.

A.1. Preliminary Notes

Before diving into the derivations, we state two useful properties, which we will use in the derivations.

Activations For the identity, relu and sigmoid activations, the size of $\phi'(h)$ is dictated by the size of the weight matrices. The activations themselves have a $\Theta(1)$ derivative.

Loss functions The derivative of the MSE loss, $\ell = (y - f)^2$, is $-2(y - f)$. Therefore the derivative is $\Theta(k)$ iff f is $\Theta(k)$.

A.2. One-layer Network

Let us assume we have a neural network $f(\xi)$ defined as

$$\begin{aligned} f(\xi) &= Vx(\xi) \\ x(\xi) &= \phi(h(\xi)) \\ h(\xi) &= U\xi \end{aligned}$$

where, $V_0 := v_0 \in \mathbb{R}^{1\times n} \sim N(0, 1/n)$ and $U_0 := \sqrt{n}u_0 \in \mathbb{R}^{n\times 1}, u_0 \sim N(0, 1/n)$ are weights. We assume n denotes the number of hidden units per layer.

Initial Distributions By initialization, V_0 has variance $\Theta(1/n)$ and U_0 has variance $\Theta(1)$. Due to the independence of $(U_0)_i$, $h_0(\xi)$ is normally distributed with a variance of $\Theta(1)$. Consequentially, $x_0(\xi)$ is $\Theta(1)$. Finally, as V_0 and $x_i(\xi)$ are independent, $f = \sum_{i=1}^{n}(V_0)_i(x_0(\xi))_i$ is approximately normally distributed with a variance of $\Theta(1)$ by the central limit theorem.

A.2.1. Partial Derivatives

Below, we explicitly derive the partial derivatives as they will be of use later. In our derivations, we drop the explicit dependence on (ξ) to lighten notation.

$$\begin{aligned} df/dx &= V^t \\ df/dV &= x^t \\ df/dh &= df/dx \odot \phi'(h) = V^t \odot \phi'(h) \\ df/du &= df/dh\sqrt{n}\xi = \sqrt{n}V^t \odot \phi'(h)\xi \\ df/d\xi &= df/dx\sqrt{n}u = \sqrt{n}(V^t \odot \phi'(h))^t u \end{aligned}$$

Above $\cdot^t$ denotes the transpose operation.

A.2.2. Forward and backward passes

In the next sections, we derive the forward and backward passes (for SGD) for the first couple of iterations. We assume that SGD with a learning rate of γ/n^{c_z} and a loss ℓ are used. For a general variable Z, the SGD update is given by $Z_t = Z_{t-1}\gamma n^{-c_z} d\ell/df \cdot df/dZ$, where $d\ell/df$ and df/dz are the partial derivatives of the loss ℓ with respect to the function output f and function f with respect to the variable z. We assume a learning rate of $\gamma = 1$, n^{-c_u} for parameters u and n^{-c_v} for parameters v. This is different from the main text, where we immediately assume $c_v = 1$ and $c_u = 0$. In the derivations below, we will show why this is necessary.

After the forward passes, we will derive the distributions of f, x, h. Our goal is to show that both f and h have $\Theta(1)$ updates (and that therefore neither explode or are unable to learn in the infinite limit).

After the backward passes, we compute the size of the updates in the parameters v, u. The backpasses allow us to see the effect of the learning rate and initialization scheme on the parameter updates. We explicitly derive the computations until the fourth iteration, as after this iteration the updates follow a generic form.

First forward pass The forward pass (with input ξ_1) proceeds as normal:

$$\begin{aligned} h_1 &= \sqrt{n} u_0 \xi_1 \\ x_1 &= \phi(h_1) \\ f_1 &= V_0 x_1 = \sum_{i=0}^{n} (V_0)_i (x_1)_i \end{aligned}$$

We obtain a loss of $\ell(y_1, f_1)$, where ℓ is the loss function, f_1 is the network output and y_1 is the true value.

Distributions after first forward pass We can argue that both h_1 and x_1 have roughly Gaussian co-ordinates by the central limit theorem. h_1 has a variance of $\Theta(1)$ (for reasoning, see "Initial Distributions" paragraph), and $x_1 = \phi(h_1)$ is $\Theta(1)$. As x_1 is $\Theta(1)$, V_0 is $\Theta(1/n)$, and x_1 and V_0 are independent, f_1 is approximately normal with variance of $\Theta(1)$ by the central limit theorem. This is a key difference with μP , where f_1 is $\Theta(1/n)$

First backward pass In the backward pass, we compute the updates with $\Delta v = v - v_0$, where v_0 is the value of v at initialization.

$$\begin{aligned} v_1 &= v_0 - n^{-c_v} (d\ell/df) \cdot (df/dv) \\ &= v_0 - n^{-c_v} \delta_1 x_1^t \qquad (1) \\ u_1 &= u_0 - n^{-c_u} (d\ell/df) \cdot (df/du) \\ &= u_0 - n^{-c_u} \delta_1 (\sqrt{n} \Delta v_0^t \odot \phi'(h_1) \xi_1) \\ &= u_0, \qquad (2) \end{aligned}$$

where $\delta_1 := d\ell/df$ is the derivative of the loss function at time step 1. Here we observe a key difference with "normal" a backpass: u_0 is not updated because $\Delta V_0 = 0$.

Distribution after first backward pass Assuming we use MSE, δ_1 is $\Theta(1)$ as f_1 is $\Theta(1)$ (see "Useful Notes" in Section A.1 for a more detailed explanation). Therefore, as x_1 is $\Theta(1)$, $V_1 - V_0$ is $\Theta(n^{-c_v})$.

Second forward pass

$$\begin{aligned} h_2 &= \sqrt{n} u_1 \xi_2 \\ &= \sqrt{n} u_0 \xi_2 \quad \text{by Equation 2} \\ x_2 &= \phi(h_2) \\ f_2 &= V_1 x_2 \\ &= V_0 x_2 - n^{-c_v} \delta_1 x_1^t x_2 \quad \text{by Equation 1} \\ &= \sum_{i=1}^{n} (V_0)_i (x_2)_i - n^{-c_v} \delta_1 (x_1)_i (x_2)_i \end{aligned}$$

Distribution after second forward pass h_2 follows almost the same distribution as h_1, as u has not been updated. Similarly, $x_2(\xi) \sim \phi(h_2)$, which is $\Theta(1)$.

Next, let's consider the f_2. Note that $f_2 = f_1 + (f_2 - f_1)$. We know the distribution of f_1, and therefore focus on the update $f_2 - f_1$. Focusing on $f_2 - f_1$ also makes explicit how much the function can change (and thereby "learn") at each step. As shown in "first forward pass", δ_1 is $\Theta(1)$. x_1 and x_2 are not independent – they both rely on U_0. Let's assume that ϕ is a linear activation. Then $E(x_1^t x_2) = \sum_i E((x_1)_i (x_2)_i) = n^2 \xi_1 \xi_2 E(u_0^2) = n^2 \xi_1 \xi_2 (1/n + 0) = n \xi_1 \xi_2$.[1] Therefore $x_1^t x_2$ is $\Theta(n)$. For other activations, such as ReLU, the same bound holds (see Section A.1 for further details). Therefore $f_2 - f_1$ is $\Theta(n^{-c_v+1})$. For $\Theta(1)$ updates, we need $c_v = 1$. As f_1 is $\Theta(1)$ and $f_2 - f_1$ is $\Theta(n^{-c_v+1})$, f_2 is $\Theta(\max(1, n^{-c_v+1}))$. From now onwards, we will use $c_v = 1$ to lighten notation.

[1] We treat the inputs as constants and use $u_0 \sim N(0, 1/n)$ in the derivation.

Second backward pass

$$
\begin{aligned}
v_2 &= v_1 - n^{-1}(d\ell/df) \cdot (df/dV) \\
&= v_1 - n^{-1}\delta_2 x_2^t \qquad (3) \\
u_2 &= u_1 - n^{-c_u}\delta_2 df/du \\
&= u_1 - n^{-c_u}\delta_2(-\sqrt{n}\Delta V_1^t \odot \phi'(h_2)\xi_2) \\
&= u_1 - n^{-c_u}\delta_2(-\sqrt{n}(n^{-1}\delta_1 x_1) \odot \phi'(h_2)\xi_2) \\
&= u_1 + n^{-c_u-1/2}\delta_1\delta_2 x_1 \odot \phi'(h_2)\xi_2 \qquad (4)
\end{aligned}
$$

Distribution by second backward pass For v_2, we can follow the same logic as in the first backward pass to show the update in $v_2 := v_2 - v_1$ is $\Theta(n^{-1/2})$.

The update for u_0 is different from before. We know that δ_1 is $\Theta(1)$. δ_2 is $\Theta(1)$ as f_2 is $\Theta(1)$, as shown in "Distribution after second forward pass". The input ξ_2 is independent of n and therefore $\Theta(1)$. x_1 is $\Theta(1)$. $\phi'(h)$ is $\Theta(1)$. Therefore, the update is $\Theta(n^{-c_u-1/2})$.

Third forward pass

$$
\begin{aligned}
h_3 &= \sqrt{n}u_2\xi_3 \\
&= \sqrt{n}u_1\xi_3 + n^{-c_u}\delta_1\delta_2 x_1 \odot \phi'(h_2)\xi_2\xi_3 \quad \text{by Equation 4} \\
x_3 &= \phi(h_3) \\
f_3 &= V_2 x_3 \\
&= V_1 x_3 - n^{-1}\delta_2 x_2^t x_3 \quad \text{by Equation 3} \\
&= \sum_i \left((V_1)_i (x_3)_i - n^{-1}\delta_2 (x_2)_i (x_3)_i\right)
\end{aligned}
$$

Distribution after third forward pass δ_1, δ_2, x_1, $\phi'(h_2)$, ξ_1, and ξ_2 are $\Theta(1)$. Therefore, the update in h_3 is $\Theta(n^{-c_u})$. $c_u = 0$ is required for h_3 to have an $\Theta(1)$ update.

Depending on the activation, $\phi(h_3)$ is either $\Theta(1)$ (if ϕ is sigmoid/tanh) or $\Theta(n^{(-c_u)})$ (if ϕ is linear/relu/identity). The update $h_3 - h_2$ is $\Theta(1)$ (if ϕ is sigmoid/tanh) or $\Theta(n^{(-c_u)})$ (if ϕ is linear/relu/identity). Again, if $c_v = 1$, then $c_u = 0$ is required for x_3 to have a $\Theta(1)$ update. From now onwards, we assume $c_u = 0$.

The update in f, i.e. $f_3(\xi_3) - f_2(\xi_3)$, is $\Theta(1)$. Let us decompose the updates in f. The left term in is $\sum_i (V_1)_i (x_3)_i$. Here, we have updated x_3, which creates a $\Theta(1)$ change. Therefore the change $f_3 - f_2$ due to the term $\sum_i (V_1)_i (x_3)_i$ due to $x_3 - x_2$ is $\Theta(1)$.

Next, let us focus on the right term $-n^{-1}\sum_i \sum_{j=1}^2 \delta_j (x_j)_i (x_3)_i$. Specifically, consider $x_2^t x_3$. Let us assume ϕ is linear. Then,

$$
x_2^t x_3 = \sum_{i=1}^n n(u_0)_i^2 \xi_2\xi_3 + (\sqrt{n}(u_0)_i\xi_2)(\delta_1\delta_2\phi'(h_2)\xi_2\xi_3) \qquad (5)
$$

From the forward pass, we know that δ_1, δ_2, ξ_j for $j = 1, 2, 3$ are $\Theta(1)$. Therefore, both the first term in the sum, $\sum_{i=1}^n n(u_0)_i^2\xi_2\xi_3$, and the second term, $\sum_{i=1}^n (\sqrt{n}(u_0)_i\xi_2)(\delta_1\delta_2\phi'(h_2)\xi_2\xi_3)$ are $\Theta(n)$. Therefore $x_2^t x_3$ is $\Theta(n)$ and $n^{-1}\sum_{i=1}^n \delta_2 (x_2)_i (x_3)_i$ is $\Theta(1)$. In conclusion, the update in f is $\Theta(1)$ if $c_v = 1, c_u = 0$.

Third backward pass

$$
\begin{aligned}
v_3 &= v_2 - n^{-1}\delta_3 x_3^t \\
&= v_0 - n^{-1}\sum_{j=1}^{3}\delta_j x_j^t \\
&= v_0 - n^{-1}\sum_{j=1}^{3}\delta_j x_j^t \\
u_3 &= u_2 - \delta_3 df/du \\
&= u_2 - \delta_3\sqrt{n}\Delta V_2^t \odot \phi'(h_3)\xi_3 \\
&= u_2 + \delta_3\sqrt{n}(n^{-c_v}\sum_{j=2}^{2}\delta_i x_i) \odot \phi'(h_3)\xi_3 \\
&= u_0 + \sum_{k=2}^{3} n^{-1/2}\delta_k(\sum_{j=1}^{k-1}\delta_j x_j) \odot \phi'(h_k)\xi_k \\
&= u_0 + n^{-1/2}\sum_{k=2}^{3}\delta_k(\sum_{j=1}^{k-1}\delta_j x_j) \odot \phi'(h_k)\xi_k
\end{aligned}
$$

Distribution by third backward pass From the previous backward pass, we know the distribution of v_2, which is $\Theta(n^{-1})$. For linear activations, x_3 is $\Theta(1)$ and δ_3 is $\Theta(1)$. Therefore $v_3 - v_2$ is $\Theta(1)$. $u_3 - u_2$ is $\Theta(n^{-1/2})$.

Fourth forward pass

$$
\begin{aligned}
h_4 &= \sqrt{n}u_3\xi_4 \\
&= \sqrt{n}u_2\xi_4 + \sqrt{n}(u_3 - u_2)\xi_4 \\
&= \sqrt{n}u_2\xi_4 + \delta_3(\sum_{j=2}^{2}\delta_i x_i) \odot \phi'(h_3)\xi_3\xi_4 \\
&= \sqrt{n}u_0\xi_4 + \sum_{k=2}^{3}\delta_k(\sum_{j=1}^{k-1}\delta_j x_j)\phi'(h_k)\xi_k\xi_4 \\
x_4 &= \phi(h_4) \\
f_4 &= V_3 x_4 \\
&= V_0 x_4 - n^{-1}\sum_{i=1}^{3}\delta_i x_i^t x_4 \\
&= \sum_i \left((V_0)_i (x_4)_i + n^{-1}\sum_{j=1}^{3}\delta_j (x_j)_i (x_4)_i \right)
\end{aligned}
$$

Distribution after fourth forward pass

We focus on the effect of the update of $u_3 - u_2$ to determine the order of the update $h_4 - h_3$. We know $u_3 - u_2$ is $\Theta(n^{-1/2})$. Therefore $\sqrt{n}(u_3 - u_2)\xi_4$ is $\Theta(1)$. Consequentially, h_4 and x_4 are $\Theta(1)$. Next, we look at the update in f. $V_3 - V_2$, $x_4 - x_3$ and x_4 are $\Theta(1)$. Therefore $f_4 - f_3 = (V_3 - V_2)x_4 = \sum(V_3 - V_2)_i(x_4)_i$ is $\Theta(1)$.

Fourth backward pass

$$
\begin{aligned}
v_4 &= v_3 - n^{-1} d\ell/df \cdot df/dV \\
&= v_3 - n^{-1}\delta_4 x_4^t \\
&= v_0 - n^{-1}\sum_{j=1}^{4}\delta_j x_j^t \\
u_4 &= u_3 - d\ell/df \cdot df/du \\
&= u_3 - \delta_4\sqrt{n}\Delta V_3 \odot \phi'(h_4)\xi_4 \\
&= u_3 + \delta_4(n^{-1/2}\sum_{j=1}^{3}\delta_j x_j) \odot \phi'(h_4)\xi_4 \\
&= u_0 + n^{-1/2}\sum_{k=2}^{4}\delta_k(\sum_{j=1}^{k-1}\delta_j x_j) \odot \phi'(h_k)\xi_k
\end{aligned}
$$

Distribution after fourth backward pass $v_4 - v_3 = n^{-1}\delta_4 x_4$. As δ_4, x_4 are $\Theta(1)$, the update is $\Theta(1/n)$. $u_4 - u_3$ is $\delta_4(n^{-1/2}\sum_{j=1}^{3}\delta_j x_j) \odot \phi'(h_4)\xi_4$, which is $\Theta(n^{-1/2})$.

A.2.3. General Forward/Backward Pass

t-th forward pass

$$
\begin{aligned}
h_t &= \sqrt{n}u_{t-1}\xi_t \\
&= \sqrt{n}u_{t-2}\xi_t + \delta_{t-1}(\sum_{j=1}^{t-2}\delta_j x_j)\phi'(h_{t-1})\xi_{t-1}\xi_t \\
&= \sqrt{n}u_0\xi_t + \sum_{k=2}^{t-1}\delta_k(\sum_{j=1}^{k-1}\delta_j x_j)\phi'(h_k)\xi_k\xi_t \\
x_t &= \phi(h_t) \\
f_t &= V_{t-1}x_t \\
&= V_{t-2}x_t - n^{-1}\delta_{t-1}x_{t-1}^t x_t \\
&= V_0 x_t - n^{-1}\sum_{i=1}^{t-1}\delta_i x_i^t x_t \\
&= \sum_i \left((V_0)_i (x_t)_i + n^{-1}\sum_{j=1}^{t-1}\delta_j (x_j)_i (x_t)_i \right)
\end{aligned}
$$

Distribution after t-th forward pass From the $t-1$th backward pass, we know $u_{t-1} - u_{t-2}$ is $\Theta(n^{-1/2})$. Therefore $\sqrt{n}(u_{t-1} - u_{t-2})\xi_t$ is $\Theta(1)$. Therefore, h_t, and consequentially x_t, are $\Theta(1)$.

Next, we look at the update in f. $V_{t-1} - V_{t-2}$, $x_t - x_{t-1}$ and x_t are $\Theta(1)$. Therefore $f_t - f_{t-1} = V_t - 1(x_t - x_{t-1}) + (V_{t-1}x_t - V_{t-2}x_{t-1})$ is $\Theta(1)$.

t-th backward pass

$$
\begin{aligned}
v_t &= v_{t-1} - n^{-1}\delta_t x_t \\
&= v_0 - n^{-1}\sum_{j=1}^{t}\delta_t x_t \\
u_t &= u_{t-1} - d\ell/df \cdot df/du \\
&= u_{t-1} - \delta_t\sqrt{n}\Delta V_{t-1}^t \odot \phi'(h_t)\xi_t \\
&= u_{t-1} + \delta_t\sqrt{n}(n^{-1}\sum_{j=1}^{t-1}\delta_j x_j) \odot \phi'(h_t)\xi_t \\
&= u_0 + n^{-1/2}\sum_{k=2}^{t}\delta_k(\sum_{j=1}^{k-1}\delta_j x_j) \odot \phi'(h_k)\xi_k
\end{aligned}
$$

Distribution by t-th backward pass In general, $v_t - v_{t-1} = n^{-1}\delta_t x_t$. As δ_t, x_t are $\Theta(1)$, the update is $\Theta(1/n)$. $u_t - u_{t-1}$ is $\delta_t\sqrt{n}(n^{-1}\sum_{j=1}^{t-1}\delta_j x_j) \odot \phi'(h_t)\xi_t$, which is $\Theta(n^{-1/2})$.

A.2.4. Proof General Updates

Proof: We can prove the form of the t-th update step and update sizes by induction.

Base case: We've shown that these updates hold for $t = 3, 4$.

Inductive step: We assume that the claims hold for the t-th forward and backward pass. We need to show that, the claims hold for the $t+1$-th step.

t+1-th forward pass

$$
\begin{aligned}
h_{t+1} &= \sqrt{n}u_t\xi_{t+1} \text{ by network definition} \\
&= \sqrt{n}u_{t-1}\xi_t + \delta_t(\sum_{j=1}^{t-1}\delta_j x_j)\phi'(h_t)\xi_t\xi_{t+1} \text{ by t-th backward pass} \\
&= \sqrt{n}u_0\xi_{t+1} + \sum_{k=2}^{t-1}\delta_k(\sum_{j=1}^{k-1}\delta_j x_j)\phi'(h_k)\xi_k\xi_{t+1} \text{ by t-th backward pass} \\
x_{t+1} &= \phi(h_{t+1}) \text{ by network definition} \\
f_{t+1} &= V_t x_{t+1} \text{ by network definition} \\
&= V_{t-1}x_{t+1} - n^{-1}\delta_t x_t^t x_{t+1} \text{ by t-th backward pass} \\
&= V_0 x_{t+1} - n^{-1}\sum_{i=1}^{t}\delta_i x_i^t x_{t+1} \text{ by t-th backward pass} \\
&= \sum_i \left((V_0)_i (x_{t+1})_i + n^{-1}\sum_{j=1}^{t}\delta_j (x_j)_i (x_{t+1})_i \right) \text{ re-writing the above equation}
\end{aligned}
$$

Distribution after t+1-th forward pass From the tth backward pass, we know $u_{t-1} - u_{t-2}$ is $\Theta(n^{-1/2})$. Therefore $\sqrt{n}(u_t - u_{t-1})\xi_t$ is $\Theta(1)$. Therefore, h_{t+1}, and consequentially x_{t+1}, are $\Theta(1)$. Next, we look at the update in f. $V_t - V_{t-1}$, $x_{t+1} - x_t$ and x_{t+1} are $\Theta(1)$. Therefore $f_{t+1} - f_t$ is $\Theta(1)$.

t+1-th backward pass

$$
\begin{aligned}
v_{t+1} &= v_t - n^{-1}\delta_{t+1}x_{t+1} \text{ by the definition of the network and SGD update rule} \\
&= v_0 - n^{-1}\sum_{j=1}^{t+1}\delta_t x_t \text{by the induction step} \\
u_{t+1} &= u_t - d\ell/df \cdot df/du \text{ by the definition of the SGD update rule} \\
&= u_t - \delta_{t+1}\sqrt{n}\Delta V_t^t \odot \phi'(h_{t+1})\xi_{t+1} \text{ by the update rule and the network definition} \\
&= u_t + \delta_{t+1}\sqrt{n}(n^{-1}\sum_{j=1}^{t}\delta_j x_j) \odot \phi'(h_{t+1})\xi_{t+1} \text{by the induction step} \\
&= u_0 + n^{-1/2}\sum_{k=2}^{t+1}\delta_k(\sum_{j=1}^{k-1}\delta_j x_j) \odot \phi'(h_k)\xi_k \text{by the induction step}
\end{aligned}
$$

Distribution by t-th backward pass In general, $v_{t+1} - v_t = n^{-1}\delta_{t+1}x_{t+1}$. As δ_{t+1}, x_{t+1} are $\Theta(1)$ (as shown in the forward pass), the update is $\Theta(1/n)$. $u_{t+1} - u_t$ is $\delta_t(n^{-1/2}\sum_{j=1}^{t}\delta_j x_j) \odot \phi'(h_{t+1})\xi_{t+1}$, which is $\Theta(n^{-1/2})$.

A.3. Two-layer Network

A.3.1. Network and Intialization

Setup We assume we have inputs $\xi, y \in \mathbb{R}$ (i.e., the input and output are of one-dimensional). We define our network as

$$
\begin{aligned}
f(\xi) &= V\bar{x}(\xi) \\
\bar{x}(\xi) &= \phi(\bar{h}(\xi)) \\
\bar{h} &= Wx \\
x(\xi) &= \phi(h(\xi)) \\
h(\xi) &= U\xi
\end{aligned}
$$

where

- $V_0 \in \mathbb{R}^{1\times n}$, $(V_0)_i \sim N(0, 1/n)$, $W \in \mathbb{R}^n$, $(W_0)_i \sim N(0, 1/n)$
- $U_0 := \sqrt{n}u_0 \in \mathbb{R}^{n\times 1}$, $(u_0)_i \sim N(0, 1/n)$. $(u_0)_i$ are updated during training.

The subscript 0 denote the values at initialization, and the subscript i distinguishes between different weights at a given time step. In general, subscript t denotes the value at the t-th iteration. Therefore V_t denotes all weights at time step t, and $(V_t)_i$ denotes the i-th weight at time step t.

Initial Distributions By initialization, V_0 has variance $\Theta(1/n)$, W_0 has has variance $\Theta(1/n)$, and U_0 has variance $\Theta(1)$. Due to the independence of $(U_0)_i$, $h_0(\xi)$ is normally distributed with a variance of $\Theta(1)$. Consequentially, $x_0(\xi)$ is $\Theta(1)$.

Similarly, due to the independence of $(W_0)_i$, $\bar{h}_i = W_i^t x$, where W_i is the i-th row of W, is normally distributed. Consequentially, $\bar{x}_0(\xi)$ is $\Theta(1)$.

Finally, as V_0 and $\bar{x}_i(\xi)$ are independent, $f = \sum_{i=1}^{n}(V_0)_i\bar{x}_0(\xi))_i$ is approximately normally distributed with a variance of $\Theta(1)$ by the central limit theorem.

A.3.2. Partial Derivatives

Below, we explicitly derive the partial derivatives as they will be of use later. In our derivations, we drop the explicit dependence on (ξ) to lighten notation.

$$
\begin{aligned}
df/d\bar{x} &= V^t \\
df/dV &= \bar{x}^t \\
df/d\bar{h} &= df/d\,\bar{\mathrm{x}} \odot \phi'(\bar{\mathrm{h}}) = V^t \odot \phi'(\bar{h}) \\
df/dx &= df/d\,\bar{\mathrm{h}} \cdot df/d\,\bar{\mathrm{x}} = V^t \odot \phi'(\bar{h}) W \\
df/dW &= df/d\,\bar{\mathrm{h}} \cdot df/dW = V^t \odot \phi'(\bar{h})\,\bar{\mathrm{x}}^t \\
df/du &= df/d\,\bar{\mathrm{h}} \cdot df/d\,\bar{\mathrm{x}} \cdot d\,\bar{\mathrm{x}}\,/du = \sqrt{n} V^t \odot \phi'(\bar{\mathrm{h}}) W \xi \\
df/d\xi &= df/d\,\bar{\mathrm{h}} \cdot df/d\,\bar{\mathrm{x}} \cdot d\,\bar{\mathrm{x}}\,/d\xi = \sqrt{n} V^t \odot \phi'(\bar{\mathrm{h}}) W u
\end{aligned}
$$

Above $\cdot^t$ denotes the transpose operation.

A.3.3. Forward and backward passes

First forward pass The forward pass (with input ξ_1) proceeds as normal:

$$
\begin{aligned}
f_1 &= V_0\,\bar{\mathrm{x}}_1 \\
\bar{\mathrm{x}}_1 &= \phi(\bar{\mathrm{h}}_1) \\
\bar{\mathrm{h}}_1 &= W_0 x_1 \\
x_1 &= \phi(h_1) \\
h_1 &= U_0 \xi_1
\end{aligned}
$$

Further, we obtain a loss of $\ell(y_1, f_1)$, where ℓ is the loss function, f_1 is the network output and y_1 is the true value.

Distributions after first forward pass Using similar arguments as for the single hidden layer setting, we can argue that h_1, x_1, $\bar{\mathrm{h}}_1$, $\bar{\mathrm{x}}_1$ and f_1 have roughly Gaussian co-ordinates by the central limit theorem with variance $\Theta(1)$.

First backward pass In the backward pass, we compute the updates with $\Delta V = V - V_0$, where V_0 is the value of V at initialization.

$$
\begin{aligned}
V_1 &= V_0 - n^{-c_v}(d\ell/df) \cdot (df/dV) \\
&= V_0 - n^{-c_v}\delta_1\,\bar{\mathrm{x}}_1^t \qquad (6) \\
w_1 &= w_0 - n^{-c_w}(d\ell/df) \cdot (df/dW) \\
&= w_0 - n^{-c_w}\delta_1 \Delta V_0^t \odot \phi'(\bar{h_1}) x_1^t \\
&= w_0 \qquad (7) \\
u_1 &= u_0 - n^{-c_u}(d\ell/df) \cdot (df/du) \\
&= u_0 - n^{-c_u}\delta_1 \sqrt{n} \Delta V_0^t \odot \phi'(\bar{h_1}) W_0 \xi_1 \\
&= u_0, \qquad (8)
\end{aligned}
$$

where $\delta_1 := d\ell/df$ is the derivative of the loss function at time step 1. Here we observe a key difference with "normal" a backpass: w_0 and u_0 are not updated because $\Delta V_0 = 0$.

Distribution after first backward pass By the same reasoning as in the single layer case, $V_1 - V_0$ is $\Theta(n^{-c_v})$.

Second forward pass

$$
\begin{aligned}
h_2 &= \sqrt{n} u_1 \xi_2 \\
&= \sqrt{n} u_0 \xi_2 \quad \text{by Equation 8} \\
x_2 &= \phi(h_2) \\
\bar{\mathrm{h}}_2 &= W_1 x_2 \\
&= W_0 x_2 \quad \text{by Equation 7} \\
\bar{\mathrm{x}}_2 &= \phi(\bar{\mathrm{h}}_2) \\
f_2 &= V_1 x_2 \\
&= V_0\, \bar{\mathrm{x}}_2 - n^{-c_v} \delta_1\, \bar{\mathrm{x}}_1^t\, \bar{\mathrm{x}}_2 \quad \text{by Equation 6} \\
&= \sum_{i=1}^{n} (V_0)_i (\bar{\mathrm{x}}_2)_i - n^{-c_v} \delta_1 (\bar{\mathrm{x}}_1)_i (\bar{\mathrm{x}}_2)_i
\end{aligned}
$$

Distribution after second forward pass $h_2, x_2, \bar{\mathrm{h}}_2,$ and $\bar{\mathrm{x}}_2$ all have similar distributions to the first forward pass as the weights have not been updated. Consequentially, they have a variance of $\Theta(1)$.

Next, we consider f_2. $V_0 x_2$ has roughly the same distribution as before as the distribution of x_2 has not changed. Therefore we focus on $n^{-c_v} \delta_1\, \bar{\mathrm{x}}_1^t\, \bar{\mathrm{x}}_2$. δ_1 is $\Theta(1)$ because f_1 is $\Theta(1)$. For simplicity, let us assume a linear activation ϕ. Then

$$
\begin{aligned}
E(\bar{\mathrm{x}}_1^t\, \bar{\mathrm{x}}_2) &= \sum_i E((\bar{\mathrm{x}}_1)_i (\bar{\mathrm{x}}_2)_i) \\
&= \sum_i E((\sum_j (W_0)_j U_0 \xi_1)_i (\sum_j (W_0)_j U_0 \xi_2)_i) \\
&= \xi_1 x_2 \sum_i \sum_j E((W_0)_{j,i}^2 (U_0)_{i,j}^2) \\
&= \xi_1 x_2 \sum_i \sum_j E((W_0)_{j,i}^2) E((U_0)_{i,j}^2) \quad \text{by independence} \\
&= \xi_1 x_2 \sum_i \sum_j (1/n)(1) \quad \text{by intialization} \\
&= n \xi_1 x_2
\end{aligned}
$$

Therefore the update in f is $\Theta(n^{-c_v+1})$. Therefore, for f to be updated maximally (without blowing up), it must hold that $c_v = 1$. From now onwards, we will use $c_v = 1$ to lighten notation.

Second backward pass

$$
\begin{aligned}
v_2 &= v_1 - n^{-1} (d\ell/df) \cdot (df/dv) \\
&= v_1 - n^{-1} \delta_2\, \bar{\mathrm{x}}_2^t \qquad (9) \\
w_2 &= w_1 - (d\ell/df) \cdot (df/dw) \\
&= w_1 - n^{-c_w} \delta_2 \Delta V_1^t \odot \phi'(\bar{\mathrm{h}}_2) x_2^t \\
&= w_1 + n^{-c_w} \delta_2 (n^{-1} \delta_1\, \bar{\mathrm{x}}_1) \odot \phi'(\bar{\mathrm{h}}_2) x_2^t \\
&= w_1 + n^{-c_w - 1} \delta_1 \delta_2\, \bar{\mathrm{x}}_1 \odot \phi'(\bar{\mathrm{h}}_2) x_2^t \qquad (10) \\
u_2 &= u_1 - n^{-c_u} (d\ell/df) \cdot df/du \\
&= u_1 - n^{-c_u} \delta_2 \sqrt{n} \Delta V_1^t \odot \phi'(\bar{h_2}) W_0 \xi_2 \\
&= u_1 - n^{-c_u} \delta_2 \sqrt{n} (n^{-1} \delta_1\, \bar{\mathrm{x}}_1) \odot \phi'(\bar{h_2}) W_0 \xi_2 \\
&= u_1 + n^{-c_u - 1/2} \delta_1 \delta_2\, \bar{\mathrm{x}}_1 \odot \phi'(h_2) W_0 \xi_2 \qquad (11)
\end{aligned}
$$

Distribution by second backward pass For v_2, we can follow the same logic as in the first backward pass to show the update in $v_2 := v_2 - v_1$ is $\Theta(n^{-1})$. Next, let us consider the update of w. Previously, we have shown that δ_1, δ_2 and $\phi'(h_2)$ are $\Theta(1)$. Therefore we focus on $\bar{\mathrm{x}}_1^t\, x_2$. $\bar{\mathrm{x}}_1$ and x_2 are $\Theta(1)$, therefore the sum is $\Theta(n)$. The update in W is $\Theta(n^{-c_w})$.

Lastly, we consider the update of u. As before, we know that δ_1, δ_2, $\phi'(h_2)$, and ξ_3 are $\Theta(1)$. Therefore, we focus on $\bar{\mathrm{x}}_1\, W_0$, which is approximately $\Theta(1)$. Therefore the update in u is $\Theta(n^{-c_u-1/2})$.

Third forward pass

$$
\begin{aligned}
h_3 &= \sqrt{n} u_2 \xi_3 \\
&= \sqrt{n} u_0 \xi_3 + n^{-c_u} \delta_1 \delta_2\, \bar{\mathrm{x}}_1 \odot \phi'(h_2) W_0 \xi_2 \xi_3 \quad \text{by Equation 11} \\
x_3 &= \phi(h_3) \\
\bar{\mathrm{h}}_3 &= W_2 x_3 \\
&= W_0 x_3 + n^{-c_w-1} \delta_1 \delta_2\, \bar{\mathrm{x}}_1 \odot \phi'(\bar{\mathrm{h}}_2) x_2^t \quad \text{by Equation 10} \\
\bar{\mathrm{x}}_3 &= \phi(\bar{\mathrm{h}}_3) \\
f_3 &= V_2\, \bar{\mathrm{x}}_3 \\
&= V_0\, \bar{\mathrm{x}}_3 - \sum_{j=1}^{2} n^{-c_v} \delta_j\, \bar{\mathrm{x}}_j^t\, \bar{\mathrm{x}}_3 \quad \text{by Equation 9} \\
&= \sum_{i=1}^{n} (V_0)_i (\bar{\mathrm{x}}_3)_i - \sum_{j=1}^{2} n^{-c_v} \delta_j (\bar{\mathrm{x}}_j)_i (\bar{\mathrm{x}}_3)_i
\end{aligned}
$$

The update in h_3 is $\Theta(n^{-c_u})$ (as the update in u is $\Theta(n^{-c_u-1/2})$ and ξ_3 is $\Theta(1)$. Therefore, for maximal feature learning we need $c_u = 0$. Consequentially, x_3 is $\Theta(1)$. The update in $\bar{\mathrm{h}}_3$ is $\Theta(n^{-c_w})$. Therefore, for maximal feature learning (without allowing the features to blow up) we need $c_w = 0$. Consequentially, $\bar{\mathrm{x}}_3$ is $\Theta(1)$.

Excluding pathological cases (where the covariance of the updates is strongly negatively correlated, leading to variance collapse) the update in f_3 and f_3 are $\Theta(1)$

Third backward pass

$$
\begin{aligned}
v_3 &= v_2 - n^{-1} (d\ell/df) \cdot (df/dV) \\
&= v_2 - n^{-1} \delta_3\, \bar{\mathrm{x}}_3^t \\
w_3 &= w_2 - (d\ell/df) \cdot (df/dW) \\
&= w_2 - n^{-c_w} \delta_3 \Delta V_2^t \odot \phi'(\bar{\mathrm{h}}_3) x_3^t \\
&= w_2 + n^{-c_w} \delta_3 (\sum_{k=1}^{2} n^{-1} \delta_k\, \bar{\mathrm{x}}_k) \odot \phi'(\bar{\mathrm{h}}_3) x_3^t \\
u_3 &= u_2 - n^{-c_u} (d\ell/df) \cdot df/du \\
&= u_2 - n^{-c_u} \delta_3 \sqrt{n} \Delta V_2^t \odot \phi'(\bar{h_3}) W_2 \xi_3 \\
&= u_2 - n^{-c_u-1/2} \delta_3 (\sum_{k=1}^{2} \delta_k\, \bar{\mathrm{x}}_k) \odot \phi'(\bar{h_3}) W_3 \xi_3
\end{aligned}
$$

Distribution by third backward pass From the previous backward pass, we know the distribution of v_2, which is $\Theta(n^{-1})$. The update in W is approximately $\Theta(1)$. The update in u is approximately $\Theta(1)$. The fourth passes follow a similar trend to the third passes. We can therefore move onto the general case.

t-th forward pass

$$
\begin{aligned}
h_t &= \sqrt{n} u_{t-1} \xi_t = \sqrt{n} u_{t-2} \xi_k + \delta_{t-1} (\sum_{k=1}^{t-2} n^{-1} \delta_k \, \bar{\mathrm{x}}_k) \odot \phi'(\bar{\mathrm{h}}_{t-1}) W_{t-1} \xi_{t-1} \xi_t \\
x_t &= \phi(h_t) \\
\bar{\mathrm{h}}_t &= W_{t-1} x_t = W_{t-1} x_t + \delta_{t-1} (\sum_{k=1}^{t-2} n^{-1} \delta_k \, \bar{\mathrm{x}}_k) \odot \phi'(\bar{\mathrm{h}}_{t-1}) x_{t-1}^t x_t \\
\bar{\mathrm{x}}_t &= \phi(\bar{\mathrm{h}}_t) \\
f_t &= V_{t-1} \, \bar{\mathrm{x}}_t = V_0 \, \bar{\mathrm{x}}_t - \sum_{j=1}^{t-1} n^{-c_v} \delta_j \, \bar{\mathrm{x}}_j^t \, \bar{\mathrm{x}}_t = \sum_{i=1}^{n} (V_0)_i (\bar{\mathrm{x}}_t)_i - \sum_{j=1}^{-1} n^{-1} \delta_j (\bar{\mathrm{x}}_j)_i (\bar{\mathrm{x}}_t)_i
\end{aligned}
$$

t-th backward pass

$$
\begin{aligned}
v_t &= v_{t-1} - n^{-1} (d\ell / df) \cdot (df / dV) \\
&= v_t - n^{-1} \delta_t \, \bar{\mathrm{x}}_t^t \\
w_t &= w_{t-1} - (d\ell / df) \cdot (df / dW) \\
&= w_{t-1} - \delta_t \Delta V_{t-1}^t \odot \phi'(\bar{\mathrm{h}}_t) x_t^t \\
&= w_{t-1} + \delta_t (\sum_{k=1}^{t-1} n^{-1} \delta_k \, \bar{\mathrm{x}}_k) \odot \phi'(\bar{\mathrm{h}}_t) x_t^t \\
u_t &= u_{t-1} - (d\ell / df) \cdot df / du \\
&= u_{t-1} - \delta_t \sqrt{n} \Delta V_{t-1}^t \odot \phi'(\bar{\mathrm{h}}_{t-1}) W_{t-1} \xi_t \\
&= u_{t-1} - n^{-1/2} \delta_t (\sum_{k=1}^{t-1} n^{-1} \delta_k \, \bar{\mathrm{x}}_k) \odot \phi'(\bar{\mathrm{h}}_{t-1}) W_{t-1} \xi_t
\end{aligned}
$$

B. Experiment Details

We train MLP classifiers on MNIST, with three layers of widths $(n, n, 10)$, with activations (relu, relu, softmax). We train the models using SGD (with batch size 64) and cross-entropy loss. We use early stopping and train for at most 300 epochs. For ensembles, we use seeds $0 - 9$.

We perform a grid search over models of width 64 for the hyper-parameters γ, the learning rates (one general learning rate, and one specifically for the last layer), and $\alpha = [\alpha_1, \alpha_2]$, constants which we multiply the first and last layer by (as in Yang & Hu (2021)). We consider γ in $2^{[-5,-4\ldots,-1,0,1\ldots5]}$ and α in $2^{[-6,-3,0,3,6,12]}$. We split the training set into a $80 - 20$ partition (with seed 0 in Pytorch) to create a validation set. We train all validations set models for 25 epochs. The final hyper-parameters are given in Table 1. Due to computational constraints, we use the hyper-parameters found for width 64 for larger width models.

Table 1. Hyper-parameters for different initializations

HYPERPARAMETER	SP	μP	OURS
γ	$[2.0, 2.0]$	$[0.03125, 0.03125]$	$[0.25, 0.03125]$
α	$[0.03125, 32]$	$[8.0, 64.0]$	$[512, 0.125]$

Figure 1 shows the average accuracy of the models. The training of our parametrization is unstable as can be seen by the high variance, and we plan to further investigate this. Nevertheless, for most larger widths, the performances do not differ significantly.

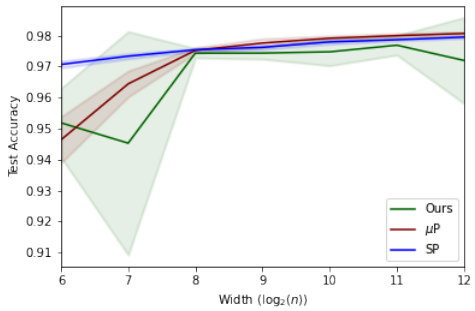

Figure 1. Average test accuracy of the networks of varying widths and parametrizations.

Feature Learning For the feature learning experiments, we apply principal component analysis (with whitening) to the features (i.e., outputs of the penultimate layer). Next, we embed the features using the top 15 features and use them as inputs to a logistic regression. For both, we use scikit-learn with default settings. We created a $4500/500$ train/test split on the test dataset for the logistic regression.

Uncertainty For the uncertainty experiments, we estimate the predictive entropy as in Smith & Gal (2018). We measure the average in-distribution (i.e. on the MNIST test set) predictive entropy and the average out-of-distribution entropy (i.e., on FashionMNIST test set) and plot the difference across various different widths.

References

Smith, L. and Gal, Y. Understanding measures of uncertainty for adversarial example detection, 2018.

Yang, G. and Hu, E. J. Feature learning in infinite-width neural networks, 2021.