Non-Gaussian Tensor Programs

Eugene Golikov
École Polytechnique Fédérale de Lausanne
Lausanne, Switzerland
evgenii.golikov@epfl.ch

Greg Yang
Microsoft Research
Seattle, USA
gregyang@microsoft.com

Abstract

The Tensor Programs framework has produced a series of powerful results by 1) expressing any deep learning computation of concern as a principled composition of elementwise nonlinearities and matrix multiplication, and 2) inductively reasoning about the program behavior as the sizes of the matrices in the program tend to infinity. For example, this framework showed that infinitely wide neural networks of *any architecture* exhibit Gaussian process behavior at initialization and evolve like a kernel model during training in the so-called NTK parametrization [29, 30, 34]. Moreover, this framework yielded a novel parameterization, coined μP [32], that for the first time enabled hyperparameter tuning for enormous networks too expensive to train more than once [33]. However, this framework has so far been limited to Gaussian initialized weights, while uniform or truncated Gaussian distributions are more prevalent in practice. This work extends Tensor Programs to general non-Gaussian initializations, thus recovering all of the above results in all practical settings.[1]

1 Introduction

Universality in Neural Network Initialization Arguably the most prevalent initialization scheme used to train neural networks today is the one proposed by [10].[2] This scheme defines a specific way of scaling initial layer weights with layer sizes, however, it does not put any preference on the initial weight distribution. While some works [12, 11] use a Gaussian distribution, some practitioners advocate for a uniform one since it is possible to sample very large weights from a Gaussian, causing numerical instabilities.

However, a theorist, especially with a background on physics or probability theory, would suspect that the precise choice of distribution does not matter as the neural network's width tends to infinity. This is a belief in the general concept of *universality* — large systems display a consistent behavior at a macroscopic scale regardless of the microscopic details.[3]

This consideration has convinced deep learning practitioners that while the scale of initial weights is important, the initial weight distribution is largely a matter of personal choice.[4] However, the situation appears to be more subtle: roughly speaking, distribution universality holds for hidden weights but does not hold for input and output weights. We formulate the precise *universality principle* (Principle 2) for neural network random intialization in Section 2. We do so in plain

[1] We refer the reader to the arXiv version for the latest version of the manuscript.

[2] Note, however, other meaningful initialization schemes, e.g. mean-field [24, 3, 26] which is a special case of μP [32].

[3] The simplest example of universality: taking the average of many iid copies of a random variable always yield its mean, regardless of whether the variable is Gaussian, uniform, Laplace, etc.

[4] See e.g. a discussion in https://github.com/fchollet/keras/issues/52.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

English, but to deduce this principle requires some fundamental advancements in the theory of *Tensor Programs.* Below, we briefly review this theory before describing our contributions to it.

Tensor Programs Just like autograd [25] empirically automates the calculation of chain rule of arbitrary computation graphs, Tensor Programs (TP) [28] has automated the theoretical calculation of infinite-width limits of the same (where width of a computation graph corresponds to the size of matrices). What previously were difficult limits to calculate, now becomes routine via TP. For example, [17] in 1994 showed that randomly initialized wide shallow neural networks are Gaussian Processes (which is called the Neural Network-Gaussian Process Correspondence, or NNGP Correspondence), but only recently this has been extended to deep perceptrons [15, 16] and more advanced architectures such as convolutional neural networks [5, 18], and each such extension requires painstaking calculations and careful applications of Law of Large Numbers and Central Limit Theorem. But with Tensor Programs, one can show that NNGP Correspondence holds for any architecture all at once [29]. Similarly, in a certain parametrization, a wide multi-layer perceptron (MLP) evolves like a linear model during training [13], but showing this for advanced architectures was very difficult. TP [30, 34] again was able to prove this behavior for any architecture. Finally, TP gave rise to the Dynamical Dichotomy Theorem [32], a classification of all natural infinite-width limits of neural networks, and led to the discovery of Maximal Update Parametrization, or μP. These results underlie the hyperparameter transfer technology that for the first time enabled the hyperparameter tuning of enormous neural networks too expensive to train more than once [33].

Distributional Universality However, these results were only proven for computational graphs whose matrices and vectors are random Gaussians (which, in the special case of graphs pertaining to neural network training, means random Gaussian initialization of NN). What happens when they take more general distributions?

In this work, we show that any program with non-Gaussian matrices and vectors also have infinite-width limits under mild conditions. In fact, they often coincide with the limits of Gaussian samplings: we shall formulate a general *distributional universality principle* for Tensor Programs (Principle 3), from which the universality principle (Principle 2) for NN random intialization (mentioned in the beginning of this section) follows as a special case.

For those familiar with the language of Tensor Programs, the principle can be summarized simply: all programs have the same limit if their Gaussian matrices are swapped out for non-Gaussian matrices with the same variances; however, this is not true in general for their initial vectors.

Applications to Random Matrix Theory The universality principle also makes the proof of the Semicircle and Marchenko-Pastur Laws in [31] automatically valid for non-Gaussian random matrix ensembles. Likewise, it shows that the asymptotic singular value distribution of the input-output Jacobian of a random neural network does not depend on the distribution of random initialization beyond its variance.

2 Distributional Universality in Words

In this section, we state, at a level understandable to practitioners, where the sampling distribution in random initialization matters (beyond the first two moments). We begin with the simple example of MLPs.

Motivating Example: MLP Let us first consider the case of a simple biasless multilayer perceptron (MLP) $f(\xi)$ with L hidden layers and nonlinearity ϕ:

$$f(\xi) = W^{L+1}\phi(W^L\phi(\cdots\phi(W^1\xi)\cdots)). \tag{1}$$

Here $W^2, \ldots, W^L \in \mathbb{R}^{n\times n}$, $W^1 \in \mathbb{R}^{n\times d_{\text{in}}}$ and $W^{L+1} \in \mathbb{R}^{d_{\text{out}}\times n}$, where d_{in} (resp. d_{out}) is the input (resp. output) dimension, and we call n the *width* of f. Then we can formulate the following *universality* principle:

Principle 1 (Universality in MLP Initialization)**.** *As width tends to infinity, two different iid random initializations of a biasless multilayer perceptron (MLP) induce* identical training behavior *as long as*

1) they sample input and output weights the same way, and 2) they sample hidden weights with the same mean and variance.[5]

Here "identical training behavior" means that for *any* sequence of batches of data, performing SGD[6] from either random initialization would see the same function after any number steps of training.[7]

The key point in Principle 1 is that hidden weights are not sensitive to the exact distribution but the input and output weights are. We can demonstrate the sensitivity of the input and output weights on a simple example. Consider the case of the 1-hidden-layer (i.e. $L = 1$) version of Eq. (1) with $d_{\text{in}} = d_{\text{out}} = 1$, and furthermore with input and output weights tied, which we name $U = W^1 = W^{2\top} \in \mathbb{R}^n$ (where again n is width):

$$f(\xi) = \frac{1}{n} U^\top \phi(U\xi). \tag{2}$$

Here, the additional $\frac{1}{n}$ factor compared to Eq. (1)[8] is just a normalization so that the network output will not blow up to infinity as width n becomes large; it's convenient but not essential for what we will discuss. Consider two alternatives (G) and (R) for sampling U:

$$\text{(R)}\ \ U_\alpha = \pm 1 \text{ with prob. } 1/2 \quad \text{or} \quad \text{(G)}\ \ U_\alpha \sim \mathcal{N}(0,1).$$

Both methods have variance 1. Now suppose the nonlinearity ϕ is 0 outside the interval $[-1/2, 1/2]$ and otherwise it is 1. Then

$$f(1) = 0 \text{ with init (R)} \quad \text{but} \quad f(1) \to \mathbb{E}_z z\phi(z) > 0 \text{ with init (G)}$$

as $n \to \infty$, where $z \sim \mathcal{N}(0,1)$. Thus (G) and (R) definitely do not induce identical training behaviors — they are not even identical at initialization!

This example can be generalized to deep biasless MLPs to show that input and output weights are sensitive to the sampling distribution. Conversely, from our main theorem (Theorem 3.7) below, it will also be clear that hidden weights are insensitive to the sampling distribution.

What Principle 1 Gets Wrong in General Architectures Unfortunately, Principle 1 is not correct if we go beyond biasless MLP. Indeed, if we just add bias to such an MLP, then one can easily generalize the example of Principle 1 to show that biases are also sensitive to their exact sampling distribution. As the architecture becomes complex, it becomes difficult to figure out whether a particular parameter tensor is sensitive or not in an ad hoc fashion (e.g., layernorm weights and biases? Self-attention weights? etc).

Principle for General Case Fortunately, there is a simple rule to tell which parameter tensors are sensitive based on the following.

Definition 2.1 ([33])**.** Let P be a parameter tensor in a neural network of any architecture. As width becomes large, if P's size remains constant, then we say P is *scalar-like*; if exact one dimension of P becomes large, we say P is *vector-like*; if exact two dimensions of P becomes large, we say P is *matrix-like*.[9]

Example 2.2. In Eq. (1), $W^2, \ldots, W^L$ are all matrix-like while W^1 and W^{L+1} are vector-like because d_{in} and d_{out} are fixed even as n varies.[10] If we add biases,

$$f(\xi) = b^{L+1} + W^{L+1}\phi(b^L + W^L\phi(\cdots\phi(b^1 + W^1\xi)\cdots)), \tag{3}$$

then $b^{L+1} \in \mathbb{R}^{d_{\text{out}}}$ is scalar-like while $b^1, \ldots, b^L \in \mathbb{R}^n$ are vector-like.

[5] "same" means between the two initialization methods; on the other hand, different weight entries can have different distributions.

[6] or Adam, or Adagrad, with or without momentum, etc. We keep it simple here and just talk about SGD.

[7] This can be made even more general in the context of Tensor Programs; see [34, 32].

[8] This results in the so-called mean field parametrization [24, 3, 26], which is a special case of the maximal update parametrization [32].

[9] We can further define *tensor-like* and so on, but practically speaking, all relevant large neural networks (e.g., BERT, GPT-3, etc [4, 1]) will have at most matrix-like parameters because of the storage and compute cost of "tensor-like" parameters.

[10] Think of d_{in} and d_{out} as being fixed by the dataset, while you, as a model builder, can freely vary n.

Example 2.3. Some more advanced examples for practitioners, that may be skipped on first reading: If f in Eq. (3) is convolutional (instead of dense), then the same categorization (of W^l, b^l into scalar-, vector-, and matrix-like) holds because the kernel size of convolutions is constant as width increases: 1) $W^2, \ldots, W^L$ are matrix-like, 2) $W^1, W^{L+1}, b^1, \ldots, b^L$ are vector-like, and 3) b^{L+1} is scalar-like. Layernorm weights and biases are vector-like if the input to that layernorm has exactly one hidden dimension (which is almost always the case in practice). Self-attention weights W^k, W^q, W^v are matrix-like if d_{head} is fixed as d_{model} and n_{head} increases or if n_{head} is fixed as d_{model} and d_{head} increases (one of which happens almost always in practice).

With this concept in mind of scalar-, vector-, and matrix-like, we can formulate the general universality principle for neural network random initialization.

Principle 2 (Universality in General Neural Network Initialization)**.** *As width becomes large, two different iid random initializations of a neural network of any architecture* induce identical training behavior[11] *as long as 1) they sample scalar- and vector-like parameters the same way, and 2) they sample matrix-like parameters with the same mean and variance.*

Remark 2.4. Note both Principle 1 and Principle 2 only give *sufficient* conditions for identical training behavior for *all possible datasets and batches*. But, practically, when we only focus on specific datasets and specific training procedures at hand, it is possible that there could be weaker conditions that would ensure identical training behavior in such specific settings. For example, if the input dimension of a dataset is large and each example has entries that look iid, then the input layer is not sensitive to the exact sampling distribution (aside from the first two moments) either. Nevertheless, Principle 1 and Principle 2 yield guidelines that are generally applicable, upon which we may refine our reasoning, if we wish, based on individual specifics.

3 Tensor Programs: Main Result

Colloquially, a Tensor Program is just a computation interleaving matrix multiplication and coordinatewise nonlinearities. In prior works, there have been several different formalizations of this concept. Here, we simplify and take the following definition.

Definition 3.1. Given matrices $A^1, \ldots, A^L \in \mathbb{R}^{n\times n}$, initial vectors $g^1, \ldots, g^{M_0} \in \mathbb{R}^n$, and initial scalars $c^1, \ldots, c^{M_0} \in \mathbb{R}$, consider the following iteration for $i = M_0 + 1, \ldots, M$ that generates new vectors $g^{M_0+1}, \ldots, g^M \in \mathbb{R}^n$ and scalars $c^{M_0+1}, \ldots, c^M \in \mathbb{R}$:

$$g^i_\alpha \leftarrow \sum_{\beta=1}^n W^i_{\alpha\beta} x^i_\beta, \quad c^i \leftarrow \frac{1}{n}\sum_{\beta=1}^n x^i_\beta, \quad \text{where } x^i_\alpha = \phi^i(g^1_\alpha, \ldots, g^{i-1}_\alpha; c^1, \ldots, c^{i-1}). \tag{4}$$

Here each ϕ^i is a chosen scalar function with $(i-1)+(i-1)$ arguments and W^i is an $n \times n$ matrix. Each matrix W^i equals to either some matrix A^j of the program or its transpose $A^{j\top}$. The matrices W^i for different i can possibly be the same. In this work, we shall call any computation of this form a *Tensor Program (TP)*, or just a *program* for short. Thus each program is entirely determined by the data $\{A^j\}_{j=1}^L \cup \{g^i\}_{i=1}^{M_0} \cup \{c^i\}_{i=1}^{M_0} \cup \{\phi^i\}_{i=M_0+1}^M$ along with the correspondence between W^i and A^j or $A^{j\top}$.

This formulation of a Tensor Program is equivalent to NETSOR$\top^+$ in [31], as shown in Appendix E. As such, Eq. (4) can express any computation expressible in a DL framework such as PyTorch [20], including gradient descent iterations of neural networks of any architecture, e.g. [32, 34]. This expressivity allows one to treat a wide range of problems uniformly using just Theorem 3.4 below.

Example program For example, consider the first forward pass of a simple MLP with scalar input ξ and output $f(\xi)$:

$$f(\xi) = V^\top \sigma(W\sigma(\xi U)), \tag{5}$$

where $\xi \in \mathbb{R}; U, V \in \mathbb{R}^n; W \in \mathbb{R}^{n\times n}$, and σ is an activation function. We can express this in a TP as follows: $g^1 \leftarrow U, g^2 \leftarrow nV$ are the initial vectors and $c^1 \leftarrow \xi, c^2 \leftarrow 1$ are initial scalars (where c^2

[11] See the discussion below Principle 1 for the meaning of *identical training behavior*.

will just be ignored). $A^1 \leftarrow W$ is the sole matrix of the program. Then the program computes

$$g^3 \leftarrow W^3 x^3, \quad \text{where } W^3 \leftarrow A^1 = W \text{ and } x^3_\alpha = \phi^3(g^1_\alpha, g^2_\alpha; c^1, c^2) \leftarrow \sigma(\xi U_\alpha)$$

$$f(\xi) = c^4 \leftarrow \frac{1}{n}\sum_{\beta=1}^{n} x^4_\beta, \quad \text{where } x^4_\alpha = \phi^4(g^1_\alpha, g^2_\alpha, g^3_\alpha; c^1, c^2, c^3) \leftarrow (nV)_\alpha \sigma(g^3_\alpha)$$

where c^3 and g^4 are implicitly computed but ignored.[12] Extending this first step, the entire training process can further be written in a TP, where the learned function outputs are expressed as scalars. See [29, 30, 34, 32] for more examples.

Gaussian Tensor Programs The results achieved by the TP framework [28, 31, 34] so far most commonly spring from the following version of the so-called *Master Theorem*:

Theorem 3.2 (Gaussian Master Theorem, original formulation of [31])**.** *Consider Setup 3.3 below. Then, as $n \to \infty$, for any pseudo-Lipschitz ψ,*

$$\frac{1}{n}\sum_{\alpha=1}^{n} \psi\left(g^1_\alpha, \ldots, g^M_\alpha, c^1, \ldots, c^M\right) \xrightarrow{a.s.} \mathring{\Psi}, \tag{6}$$

where $\mathring{\Psi}$ is a deterministic scalar given by a certain recurrent formula.

Setup 3.3 (Gaussian Tensor Programs)**.** *Consider a Tensor Program with M vectors $g^1, \ldots, g^M \in \mathbb{R}^n$ and scalars $c^1, \ldots, c^M$. Suppose 1) all initial vectors $g^1, \ldots, g^{M_0}$ have standard Gaussian entries[13];2) all initial scalars $c^1, \ldots, c^{M_0}$ have almost sure limits as $n \to \infty$; 3) all matrices A^i have iid entries from $\mathcal{N}(0, n^{-1})$; 4) all the nonlinearities ϕ^i are pseudo-Lipschitz.*[14]

For example, Theorem 3.4 implies that Eq. (5)'s function values $f(\xi)$, *after training*, converge to deterministic values in various infinite-width limits, in particular the feature learning limit [32, 33].

We can reformulate the above theorem in a shorter form:

Theorem 3.4 (Gaussian Master Theorem, equivalent formulation)**.** *Consider Setup 3.3. Then, as $n \to \infty$, every scalar c^i converges almost surely to a deterministic limit $\mathring{c}^i$ which can be computed via a recurrent formula.*

It is easy to see that the above statement is equivalent to the original Theorem 3.2. Indeed, any scalar in the program has the form $\frac{1}{n}\sum_\alpha \psi(g^1_\alpha, \ldots, g^M_\alpha; c^1_\alpha, \ldots, c^M_\alpha)$ for some function ψ, while any expression of the above form can be thought as a scalar in a new program. We are going to build upon the formulation of Theorem 3.4 since it introduces fewer entities.

Non-Gaussian Tensor Programs We shall generalize Theorem 3.4 to non-Gaussian distributions. But before we do so rigorously, we first formulate an easily statable principle that summarizes our results in an intuitive way.

Principle 3 (Universality in Tensor Program Sampling)**.** *For simplicity, consider TP without initial scalars. As $n \to \infty$, two different iid random samplings of a TP's matrices and initial vectors result in identical limits of scalars as long as 1) they sample all initial vectors the same way, and 2) they sample all matrix entries with the same variance (and zero mean).*

More generally, for programs with initial scalars, we just require the two samplings have the same almost sure deterministic limit for them.[15]

[12] We intentionally simplified the formulation of TP to the form Eq. (4) in particular to simplify the proofs, but at the cost that expressing common computation encounters some redundancy as shown in this example.

[13] In the original formulation of [31], initial vectors were assumed to be sampled as $(g^1_\alpha, \ldots, g^{M_0}_\alpha) \sim \mathcal{N}(\mu^{in}, \Sigma^{in})$ iid over $\alpha = 1, \ldots, n$, for some $\mu^{in} \in \mathbb{R}^{M_0}, \Sigma^{in} \in \mathbb{R}^{M_0 \times M_0}$. However, one can always construct such vectors from ones with iid standard Gaussian entries using a linear elementwise map. This map can be further absorbed into subsequent nonlinear maps in the program.

[14] A function $f : \mathbb{R}^k \to \mathbb{R}$ is called *pseudo-Lipschitz* if $|f(x) - f(y)| \le C\|x - y\|(1 + \sum_{i=1}^k |x_i|^d + |y_i|^d)$ for some $C, d > 0$.

[15] We can in fact allow the initial scalars to have non-deterministic limits, in which case the non-initial scalars will have limits that are non-deterministic as well, moreover images of the initial scalars under certain functions. In this case, we want the two samplings to have identical limit distributions for the joint distributions of all initial scalars.

Principle 1 and Principle 2 are special cases of Principle 3, since any neural network can be expressed in a TP. Principle 3 itself will follow straightforwardly from Theorem 3.7.

Now let us setup our discussion of our main theorem, Theorem 3.7.

Definition 3.5. We say $f : \mathbb{R}^k \to \mathbb{R}$ is *polynomially smooth* if it is C^∞ and its partial derivatives of any order are polynomially bounded, i.e. for any sequence $(P_1, \ldots, P_r) \in [k]^r$, we have $\left|\frac{\partial^r}{\partial x^{P_1} \cdots \partial x^{P_r}} f\right| \le C(1 + |x_1|^p + \cdots + |x_k|^p)$ for some $C, p > 0$ that may depend on $(P_1, \ldots, P_r)$.

Setup 3.6. *Consider Setup 3.3, but replace 3) and 4) with the following: 3*) there exists a sequence $\nu_3, \nu_4, \ldots > 0$ such that all matrices A^i have independent entries drawn from distributions with zero mean, variance n^{-1}, and all higher kth moment bounded by $\nu_k n^{-k/2}$; 4*) all the nonlinearities ϕ^i are polynomially smooth. We further require 5) all moments of initial scalars $c^1, \ldots, c^{M_0}$ to exist.*

Many widely-used nonlinearities, such as ReLU or MaxPool, are not polynomially smooth, thus contradicting 4*). Nevertheless, if the original nonlinearity is continuous and polynomially bounded (in particular, if it is pseudo-Lipschitz), one may apply Gaussian smoothing and get a polynomially smooth nonlinearity; see Appendix B. The narrower the smoothing kernel, the better the approximation.

The specific scenario satisfying 3*) above that is most relevant to us is if $A^i_{\alpha\beta} \sim \frac{1}{\sqrt{n}}\mathcal{D}$ where $\mathcal{D}$ is a fixed distribution, like uniform or truncated Gaussian with mean 0 and variance 1. In this case, ν_k can just be taken to be the kth moment of $\mathcal{D}$. But in general 3*) allows entries of A^i to come from different distributions.

Our main result is the following:

Theorem 3.7 (Non-Gaussian Master Theorem)**.** *Consider Setup 3.6. Then, as $n \to \infty$, every scalar c^i converges to the same $\mathring{c}^i$ as in Theorem 3.4 almost surely and in L^p for every $p \in [1, \infty)$:*

$$c^i \xrightarrow{\text{a.s. \& } L^p} \mathring{c}^i, \quad \forall p \in [1, \infty).$$

In short, Theorem 3.7 relaxes matrix sampling to be non-Gaussian and non-identically-distributed in general, at the cost of requiring a) more smoothness in nonlinearities and b) all moments of initial scalars to exist. In past applications of TP, b) has always been satisfied but a) has not for relu networks. On the other hand, we also gain L^p convergence compared to Theorem 3.4.[16] Note that Setup 3.6 still requires initial vectors to be Gaussian, but this is not essential, as we discuss in Section 4.

On Tensor Programs with variable dimensions. While Setup 3.3 and 3.6 assume all hidden dimensions to be equal, Theorem 3.4 holds also for Tensor Programs with variable dimensions, see e.g. [31]. Since our proof technique is based on interpolation between Gaussian and non-Gaussian weights, it can be straightforwardly extended to variable dimensions, as long as Theorem 3.4 holds in this setting.

4 Applications

As mentioned in the introduction, the Tensor Programs series of papers so far has proven a wide range of results, which typically have the characteristic of *architectural universality*, i.e., covering any neural architectures. But all of such results have assumed Gaussian weight initialization. Now, armed with Theorem 3.7, we show the same results hold with *non-Gaussian* weight initialization as well, under mild assumptions, thus extending them to other prevalent initializations such as uniform and truncated Gaussian. Theorem 3.7 here acts like a drop-in replacement for Theorem 3.4 in their proofs, except we need to add more smoothness assumptions on nonlinearities.

However, we cannot say that these results are distribution universal, i.e. independent of specifics of weight sampling. The reason is that our Theorem 3.7 requires all "vector-like" variables, such as input and output weights and bias vectors, to be Gaussian. We can still model non-Gaussian vector-like variables with elementwise images of Gaussians. However, this changes the program itself and hence alters the limit $\mathring{c}$ predicted by Theorem 3.7.

[16] We believe L^p convergence can be established without requiring so much smoothness in nonlinearities, but leave this to future work.

Nevertheless, the limit exists no matter if the vector-like variables are Gaussian, or not. All the corollaries of our Theorem 3.7 that we discuss below, rely merely on existence of this limit. However, the corresponding limit quantities, such as NNGP kernels or NTK, could be affected by initial distributions of input and output weights and bias vectors.

Our Theorem 3.7 can be applied to a very general class of neural networks as given by the following:

Definition 4.1. We shall call a scalar random variable a *standard Gaussian image* if it is an image of a standard Gaussian variable under a polynomially smooth function.

Setup 4.2. *Consider a neural network whose forward pass can be expressed as Eq.* (4) *with each matrix W^i corresponding to some A^j but not its transposed. Suppose 1) all the activation functions are polynomially smooth; 2) input and output layer weights and entries of all hidden bias vectors are initialized with iid standard Gaussian images; 3) weights of any other layer are initialized according to 3*) of Setup 3.6; 4) output layer bias vector entries are initialized with zeros.*

NNGP Correspondence. The following result is an immediate consequence of our Theorem 3.7:

Corollary 4.3. *Consider Setup 4.2 and further assume 2^+) all hidden bias vector entries to be initialized with iid standard Gaussians. At initialization, as width tends to infinity, the pre-activation output of any hidden neuron of any layer except for the first one converges weakly to a Gaussian Process (GP). Moreover, the empirical kernel associated to this neuron converges pointwise and almost surely to the kernel of this GP.*

Here, the NNGP kernels may be affected by the distribution of input layer weights. Indeed, assume scalar inputs and consider the kernel associated to neurons of the second hidden layer:

$$K(x_1, x_2) = \left(\sum_{\beta=1}^{n} W_{\alpha\beta}\phi(u_\beta x_1)\right)\left(\sum_{\gamma=1}^{n} W_{\alpha\gamma}\phi(u_\beta x_2)\right), \tag{7}$$

where u is a vector of input layer weights, W is the first hidden layer matrix, and one can take α to be any number from 1 to n. Assume for simplicity ϕ to be an identity function. If $u_\beta = \psi(z_\beta)$ for each β and a polynomially smooth function ψ, the choice of ψ will affect $K(x_1, x_2)$. Indeed, if one takes $\psi(z) = cz$, the kernel gets a factor c^2 compared to $\psi(z) = z$.

NTK Correspondence. We can directly plug our Theorem 3.7 into the proof of [34] and obtain

Corollary 4.4. *Consider Setup 4.2. Then under NTK parameterization*[17] *and SGD weight updates, NTK of the network at any optimization step converges pointwise and almost surely to a finite deterministic limit that does not depend on the timestep. Moreover, the output of the network at any timestep converges pointwise and almost surely to the output of a kernel method with this limit as a kernel.*

Here again, the above kernel (called NTK) could be affected by the distribution of input and output weights. The simplest example is given by a neural net with scalar inputs and outputs, a single hidden layer, and no biases, under NTK parameterization: $f(x) = n^{-1/2}\sum_{\alpha=1}^{n} v_\alpha\phi(u_\alpha x)$. Let the entries of u and v be initialized iid from distribution P_u and P_v correspondingly. The limit NTK at initialization is then given by

$$\Theta(x_1, x_2) = \mathbb{E}_{\xi_u\sim P_u}\mathbb{E}_{\xi_v\sim P_v}\left(\phi(\xi_u x_1)\phi(\xi_u x_2) + \xi_v^2\phi'(\xi_u x_1)\phi'(\xi_u x_2)x_1x_2\right), \tag{8}$$

which clearly depends on the distributions P_u and P_v.

We empirically validate Corollaries 4.3 and 4.4 in Appendix N.

In Appendix D, we present a variant of Theorem 3.7 which works for non-smooth but Lipschitz nonlinearities (like ReLU or MaxPool). It allows to generalize Corollary 4.3 to ReLU nets, but not Corollary 4.4 (that's because a Tensor Program expressing the backward pass of a ReLU net has ReLU derivatives, which are not even continuous, as nonlinearities).

Random Matrix Theory. Our Theorem 3.7 implies the semi-circle law for non-Gaussian Wigner matrices, the Marchenko-Pastur law for $AA^\top$, where A is non-Gaussian, and Free Independence Principle (FIP) for Tensor Programs with non-Gaussian initial weights, thus generalizing TP3 [31]; we discuss these results in Appendix C. Moreover, since our Theorem 3.7 guarantees convergence in mean, we were able to state FIP without assuming linearly bounded nonlinearities as in [31].

[17] For more context on NTK theory, see the seminal work of [13].

Classification of Infinite-Width Limits. Consider now an L-hidden-layer perceptron trained using stochastic gradient descent (SGD). TP4 [32] proposes the notion of *an abc-parameterization*, which is specified by a set of numbers $\{a_l, b_l\}_l \cup \{c\}$: (1) each weight is parameterized as $W^l = n^{-a_l} w^l$ for the actual trainable parameter w^l, (2) $w^l_{\alpha\beta} \sim \mathcal{N}(0, n^{-2b_l})$ at initialization, and (3) the SGD learning rate is taken as ηn^{-c} for some constant η.

When c is too small (i.e. learning rate is too large), SGD can lead to blowup of logits; such a parameterization is called unstable. When c is too large, the function computed by the network does not change with time; such a parameterization is called trivial. The result below follows from Theorem 3.4:

Corollary 4.5 (Dynamical dichotomy; [32])**.** *Any nontrivial stable abc-parameterization yields a (discrete-time) infinite-width limit. This limit either (1) allows the last layer embedding to evolve nontrivially, or (2) is described by kernel gradient descent in function space, but not both.*

The first situation is called feature learning, as opposed to the second one, called kernel learning. Feature learning is argued to be a more sensible learning regime for a neural network.

In TP4, [32] proposed the so-called μ-parameterization, which uniquely exhibits maximal feature learning in the infinite-width limit among all abc-parameterizations. In TP5, [33] generalized this μ-parameterization to the Adam optimizer [14] and a wider class of architectures and used it to facilitate hyperparameter tuning of large models.

Since Dynamical dichotomy (Corollary 4.5) is a direct consequence of convergence of averages given by Theorem 3.4, we can readily apply our Theorem 3.7 for the same purpose:

Corollary 4.6 (Dynamical dichotomy; ours)**.** *Consider an L-hidden-layer perceptron trained using stochastic gradient descent (SGD) under abc-parameterization, but with w_l sampled independently from a distribution with zero mean, variance n^{-2b_l}, and all higher moments existing, for $l = 2, \ldots, L$. Then the statement of Corollary 4.5 holds.*

The μ-parameterization in this case is the same, and can be recovered by the same argument as in TP4. We therefore theoretically justify the hyperparameter tuning technique used in TP5 for neural networks with non-Gaussian initializations commonly used in practice.

On potential societal impacts. Our work concerns generic behavior of neural nets in the limit of infinite width and therefore does not provide any foreseen societal impacts. The only direct practical application of our work we are aware of is theoretical justification for the hyperparameter tuning method of TP5 [33] for non-Gaussian weight initializations. However, what our work provides is merely justification for this method, while the method itself existed before our work (and was well-justified for Gaussian weight initializations).

5 Proof of Theorem 3.7

Limitations of the Proof Technique of Theorem 3.4. The proof of Theorem 3.4 as given in [31] uses the Gaussianity of the matrices A^j in a very essential way. It leverages the property of multivariate Gaussians to remain Gaussian after conditioning on *linear* constraints.

In particular, it lets one understand the distribution of g^i conditioned on $g^{i-1}, \ldots, g^1$. Indeed, under such conditioning, W^i is in general no longer iid but nevertheless constrained only by the following equalities (where now g^j and x^j should be considered deterministic under such conditioning):

$$g^j = W^j x^j, \quad \text{for all } j < i \text{ and where } W^j = W^i \text{ or } W^{i\top}.$$

This constraint is *linear* in W^i (even though it's generally nonlinear in $g^1, \ldots, g^{i-1}$ since x^j is a nonlinear function of them). Therefore, when conditioned on $g^{i-1}, \ldots, g^1$, the matrix W^i is *still Gaussian*, albeit with some conditional mean and covariance. Quickly glossing over the rest of the proof, this allows one to reason about the conditional distribution of g^i and eventually to reduce the almost sure convergence of the $(i+1)$th scalar c^{i+1} to the almost sure convergence of the ith scalar in a different program. Then an inductive argument over M for all programs of size M can prove Theorem 3.4. A more detailed proof sketch can be found in [31, Section 6].

Now, when the matrices W^i are no longer Gaussian, this argument completely breaks down as we no longer have good control of the conditional distribution of W^i for general entry distributions. We

therefore apply a different argument. The idea is to interpolate the weights from the Gaussian ones, for which convergence of c^i is given by Theorem 3.4, to the non-Gaussian ones, and show that c^i does not change along this interpolation in the limit of large n.

Our Proof of Theorem 3.7

Definition 5.1 (Interpolated Program)**.** Given a program T in Setup 3.6, let $\tilde{A}^i$ denote an iid Gaussian matrix with the same mean and variance as the non-Gaussian matrix A^i of the program. Then we define *interpolated program* $T(t)$ for $t \in [0,1]$ as follows: $T(t)$ is identical to T except that its matrices take the following values:

$$A^i(t) \stackrel{\text{def}}{=} \tilde{A}^i \cos\left(\frac{\pi}{2}t\right) + A^i \sin\left(\frac{\pi}{2}t\right), \quad \text{for all } t \in [0,1] \text{ and } i = M_0+1, \ldots, M, \tag{9}$$

Naturally $W^i(t)$ inherits the same definition. The vectors and scalars in the program will change continuously as t varies, and consequently we write them as $g^i(t)$ and $c^i(t)$ for $i = 1, \ldots, M$.[18]

In Section 5.1, we will discuss why this specific form of interpolation[19] is important in our case.

Here $t = 0$ corresponds to the Gaussian program, while $t = 1$ corresponds to the "target" non-Gaussian one. Eventually, we aim to show that all scalars $c^i(t)$, almost surely in the limit of $n \to \infty$, will remain constant as t varies from 0 to 1 (as stated by Theorem 5.2 shortly after setting some notations), thus proving that the Gaussian and non-Gaussian programs have the same limits.

Notations For any object (matrix, vector, or scalar) $\omega(t)$ of this interpolated program, we shorthand

$$\dot{\omega}(t) \stackrel{\text{def}}{=} \frac{d}{dt}\omega(t).$$

Big-O notation, e.g., $O(n^{-1/2})$, always suppress multiplicative constants independent of n, but which may depend on everything else. The supremum $\sup_t$ is always taken over $t \in [0,1]$.

Theorem 5.2. *Consider a program in Setup 3.6 and let c be a scalar in it. For any $p \geq 1$, we have*[20]

$$\sup_t \mathbb{E}\, |\dot{c}(t)|^p = O(n^{-p/2}), \quad \textit{as } n \to \infty.$$

In other words, this result says $\dot{c}(t)$ is small in L^p norm uniformly over all t. With Theorem 5.2, some routine calculations (detailed below) then show that $c(1)$ converges almost surely and in L^p to the same limit as $c(0)$ yielding a proof of Theorem 3.7 as desired. Theorem 5.2 is proven in Appendix L, but we shall demonstrate it on a simple example in Appendix A.

Routine Calculations Finishing the Proof of Theorem 3.7 from Theorem 5.2 Once Theorem 5.2 is proven, we have, for any $p \geq 1$,

$$\mathbb{E}\,|c(1) - c(0)|^p = \mathbb{E}\left|\int_0^1 \dot{c}(t)dt\right|^p \leq \int_0^1 \mathbb{E}\,|\dot{c}(t)|^p\, dt = O(n^{-p/2})$$

where the inequality follows from power mean or Hölder's inequality and the last equality follows from Theorem 5.2. Since the RHS goes to 0 with n, this implies that $c(1) - c(0)$ converges to 0 in L^p as $n \to \infty$. With a standard application of Borel-Cantelli lemma, this also implies almost sure convergence of $c(1) - c(0)$. Since $c(0)$ converges to $\mathring{c}$ almost surely by the Gaussian theorem, Theorem 3.4, the same holds for $c(1)$.

In the process of proving Theorem 5.2, we will have proved pth moment bound on $c(1)$ for every finite $p \geq 1$ (Lemma J.6). Then a standard truncation technique uses this to convert the almost sure convergence of $c(1)$ to L^p convergence for all $p \in [1, \infty)$; see Theorem M.3.[21]

It remains to prove Theorem 5.2, which we demonstrate on a simple example in Appendix A after discussing the key properties of our interpolation (Eq. (9)).

[18] but note that they are invariant to t for any n when $i \leq M_0$ because by construction, the initial vectors and scalars are the same between the Gaussian and non-Gaussian programs.

[19] This interpolation is similar to that used in [2] in the context of approximate message passing, which can be considered as an application of the TP framework to a specific kind of programs; see Appendix F for comparison.

[20] The hidden multiplicative constant here is only independent of n but can depend on p and other details of the program.

[21] Note since $c(0)$ (the scalar in the Gaussian program) is not guaranteed by Theorem 3.4 to converge in L^p, the above L^p convergence of $c(1) - c(0)$ does not immediately imply the L^p convergence of $c(1)$.

5.1 Key Properties of Our Matrix Interpolation

There are may ways to interpolate between (or, more generally, to *couple*) $\tilde{A}$ and A other than Eq. (9). Why did we pick Eq. (9)? The following lemma proven in Appendix G summarizes the key properties of it, where especial attention should be paid to the fact that $\mathbb{E}\, a\dot{a} = 0$:

Lemma 5.3 (Interpolation Properties)**.** *For any matrix entry $a(t) = A^i_{\alpha\beta}(t)$ of a program in Setup 3.6: (1) $\mathbb{E}\, \dot{a}(t) = \mathbb{E}\, a(t)\dot{a}(t) = 0$ for all t; (2) For any integers $j, k \geq 0$ with sum $\ell = j + k$, $\sup_t \mathbb{E}\, |a(t)^j \dot{a}(t)^k| \leq \pi^\ell \nu_\ell n^{-\ell/2}$, where ν_ℓ is the scaled moment bound in Setup 3.6.*

The fact $\mathbb{E}\, a\dot{a} = 0$ is crucial and the main reason we picked the specific form Eq. (9) of the interpolation. It will allow us to kill leading terms in the Taylor expansions important to proving Theorem 5.2. We demonstrate this in an example in Eq. (13). On the other hand, (2) is just a very strong version of the obvious statement that "a and $\dot{a}$ both have typical size $1/\sqrt{n}$," but importantly, this is uniform over $t \in [0, 1]$.

6 Related works

The Tensor Programs series discusses different applications of the (Gaussian) Master theorem proven by [28]. They include: Gaussian process behavior at initialization (TP1, [29]), convergence to a kernel method (TP2, [30]), Free Independence Principle (TP3, [31]), dynamical dichotomy and μ-parameterization (TP4, [32]), and finally application of μP to hyperparameter tuning (TP5, [33]).

Neural networks converge to Gaussian processes as their width goes to infinity, as was proven by [15, 16] for fully-connected nets, and by [18, 5] for convolutional nets; see also [8]. Using the Master theorem, [29] showed that this behavior holds for a very wide class of architectures, including not only convolutional, but also graph convolutional and recurrent neural nets, ResNets, networks with batch normalization, and networks with attention. The seminal work of [13] demonstrated that under certain parameterization, the learning dynamics a neural net converges to that of a kernel method. The corresponding kernel was called Neural Tangent Kernel, or NTK, and drawn a lot of attention in recent years. While the result of [13] was proven only for fully-connected nets with smooth activations, the Master theorem allows to generalize this result for a wider class of architectures (the same as mentioned above), see [30]. [23, 21, 27] and others studied trainability of very deep and wide neural networks using random matrix theory. Their analysis crucially relied on the assumption that hidden representations of a neural network at initialization were *freely independent* from the weights. TP3 [31] was among the first works to validate this assumption rigorously; see also [19]. Infinite-width behavior of a neural net depends on scaling of its hyperparameters (like initial weights variance and learning rate) with width. Dynamical dichotomy proposed in TP4 [32] is a classification of scalings that are meaningful in a certain sense. Another classification of scalings with a different notion of meaningfulness was proposed earlier by [6, 7], but only for two-layered networks.

A distribution universality property similar to our Theorem 3.7 was shown by [2] for approximate messaging passing. However, their model does not cover most of possible neural network computations; see Appendix F for discussion.

7 Limitations of our results

First, our Theorem 3.7 is applicable only to Tensor Programs with smooth nonlinearities, which rules out several popular activation functions like ReLU or MaxPool. Our Theorem D.2 (see Appendix D) does not really solve the issue since a Tensor Program expressing the backward pass involves derivatives of the activation functions, which are not even continuous for ReLU. As a workaround, we could consider their smoothed versions (e.g. Softplus instead of ReLU) with a controllable smoothness parameter, and put this parameter very close to zero, thus getting "almost ReLU".

8 Conclusions

We present a generalization of the Master theorem of [28] to non-Gaussian weight initializations. Our generalization allows for the same applications as the original Master theorem, thus broadening the scope of applicability of the Tensor Programs machinery.

References

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[2] Wei-Kuo Chen and Wai-Kit Lam. Universality of approximate message passing algorithms. *Electronic Journal of Probability*, 26:1–44, 2021.

[3] Lenaic Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. *arXiv preprint arXiv:1805.09545*, 2018.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[5] Adrià Garriga-Alonso, Carl Edward Rasmussen, and Laurence Aitchison. Deep convolutional networks as shallow gaussian processes. *arXiv preprint arXiv:1808.05587*, 2018.

[6] Eugene Golikov. Towards a general theory of infinite-width limits of neural classifiers. In *International Conference on Machine Learning*, pages 3617–3626. PMLR, 2020.

[7] Eugene A Golikov. Dynamically stable infinite-width limits of neural classifiers. *arXiv preprint arXiv:2006.06574*, 2020.

[8] Boris Hanin. Random neural networks in the infinite width limit as gaussian processes. *arXiv preprint arXiv:2107.01562*, 2021.

[9] Michael Hardy. Combinatorics of partial derivatives. *arXiv preprint math/0601149*, 2006.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

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

[12] Geoffrey E Hinton, Alex Krizhevsky, and Ilya Sutskever. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25(1106-1114):1, 2012.

[13] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

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

[15] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

[16] Alexander G de G Matthews, Jiri Hron, Mark Rowland, Richard E Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 2018.

[17] Radford M Neal. *BAYESIAN LEARNING FOR NEURAL NETWORKS*. PhD thesis, University of Toronto, 1995.

[18] Roman Novak, Lechao Xiao, Jaehoon Lee, Yasaman Bahri, Greg Yang, Jiri Hron, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. *arXiv preprint arXiv:1810.05148*, 2018.

[19] Leonid Pastur. On random matrices arising in deep neural networks. gaussian case. *arXiv preprint arXiv:2001.06188*, 2020.

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[21] Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In *Advances in neural information processing systems*, pages 4785–4795, 2017.

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

[23] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances in neural information processing systems*, pages 3360–3368, 2016.

[24] Grant M Rotskoff and Eric Vanden-Eijnden. Trainability and accuracy of neural networks: An interacting particle system approach. *arXiv preprint arXiv:1805.00915*, 2018.

[25] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[26] Justin Sirignano and Konstantinos Spiliopoulos. Mean field analysis of neural networks: A central limit theorem. *Stochastic Processes and their Applications*, 130(3):1820–1852, 2020.

[27] Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In *International Conference on Machine Learning*, pages 5393–5402. PMLR, 2018.

[28] Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019.

[29] Greg Yang. Tensor programs i: Wide feedforward or recurrent neural networks of any architecture are gaussian processes. *arXiv preprint arXiv:1910.12478*, 2019.

[30] Greg Yang. Tensor programs ii: Neural tangent kernel for any architecture. *arXiv preprint arXiv:2006.14548*, 2020.

[31] Greg Yang. Tensor programs iii: Neural matrix laws. *arXiv preprint arXiv:2009.10685*, 2020.

[32] Greg Yang and Edward J Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning*, pages 11727–11737. PMLR, 2021.

[33] Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*, 2022.

[34] Greg Yang and Etai Littwin. Tensor programs iib: Architectural universality of neural tangent kernel training dynamics. In *International Conference on Machine Learning*, pages 11762–11772. PMLR, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 3 for scope and Section 3, 4, and 5 for contributions.
   (b) Did you describe the limitations of your work? [Yes] See Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See the last paragraph of Section 4.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3 and Setup 3.6 in particular.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Section 5 and Appendices D, G, I, J and L.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix N.2.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See plots in Appendix N.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix N.2.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Proof of Theorem 5.2 for $p = 2$ in a simple scenario

Setup To make our arguments clear, we illustrate the proof of Theorem 5.2 on the simplest non-trivial case, with only a single matrix $A = A^1$, a single initial vector $g^1 \sim \mathcal{N}(0, I)$, and with c^1, c^2, and g^3 ignored. The objects of the interpolated program (Definition 5.1) are as follows:

$$g^2(t) = A(t)\phi(g^1(t)), \quad c(t) = \frac{1}{n}\sum_{\alpha=1}^{n} x_\alpha(t), \quad \text{where } x_\alpha(t) = \psi(g^1_\alpha(t), g^2_\alpha(t))$$

where, for brevity, we have shorthanded $c = c^3$, $x = x^3$, $\phi = \phi^1$, and $\psi = \phi^3$.[22]

[22] So we would have $M_0 = 1$, $M = 3$, but these values are not important for our purposes.

For simplicity, we will assume ϕ and ψ have derivatives of all orders bounded by 1 in absolute value. Our goal in this section is to demonstrate Theorem 5.2 with $p = 2$:

$$\sup_t \mathbb{E}\, \dot{c}(t)^2 = O(n^{-1}). \tag{10}$$

In what follows, we will only talk about the interpolated program, so we will suppress the argument (t) to lighten notation.

A.1 Bounding Derivatives of c against A

We will denote $x'_\alpha \stackrel{\text{def}}{=} \partial_{g^2_\alpha} \psi(g^1_\alpha, g^2_\alpha)$, $x''_\alpha \stackrel{\text{def}}{=} \partial^2_{g^2_\alpha} \psi(g^1_\alpha, g^2_\alpha)$, and so on, which shall cause no confusion because we will never take other derivatives of x or ψ. Then we can calculate

$$\frac{\partial c}{\partial A_{\alpha\beta}} = \frac{1}{n} x'_\alpha \phi(g^1_\beta), \quad \frac{\partial^2 c}{\partial^2 A_{\alpha\beta}} = \frac{1}{n} x''_\alpha \phi(g^1_\beta)^2, \quad \text{etc.} \tag{11}$$

By the assumption that ψ and ϕ have derivatives of any order bounded by 1, this shows c's derivative against $A_{\alpha\beta}$ of any order ≥ 1 is bounded in absolute value by $1/n$. Generalizing the above calculation to mixed derivatives, one can easily see more generally,

$$|\partial^r c| \leq 1/n \quad \text{for any order } r \geq 1 \text{ mixed derivative } \partial^r \text{ in entries of } A. \tag{12}$$

If we allow ψ and ϕ to be polynomially smooth in general, then this is still true in expectation, up to multiplicative constants; see Lemma K.6.

A.2 Expanding $\dot{c}^2$

At this point it helps to think of the entries of A as a big vector a of size $N = n^2$. We will index entries of a using letters like κ, which stands for a pair $(\alpha, \beta) \in [n]^2$. We also let $D \in \mathbb{R}^N$ be the vector of derivatives $D_\kappa \stackrel{\text{def}}{=} \partial_{a_\kappa} c$ (with exact values given by Eq. (11)), so that, by chain rule,

$$\dot{c} = \sum_\kappa D_\kappa \dot{a}_\kappa.$$

Squaring this sum, we get

$$\dot{c}^2 = \sum_{\kappa,\lambda} D_\kappa D_\lambda \dot{a}_\kappa \dot{a}_\lambda = \sum_\kappa D_\kappa^2 \dot{a}_\kappa^2 + \sum_{\kappa \neq \lambda} D_\kappa D_\lambda \dot{a}_\kappa \dot{a}_\lambda$$

To show Eq. (10), it suffices to show that both sums $\sum_\kappa$ and $\sum_{\kappa\neq\lambda}$ in expectation can be bounded by $O(n^{-1})$ with constant independent of $t \in [0, 1]$.

A.3 Bounding $\sum_\kappa D_\kappa^2 \dot{a}_\kappa^2$.

Because κ ranges over a set of size $N = n^2$, we have

$$\mathbb{E} \sum_\kappa D_\kappa^2 \dot{a}_\kappa^2 \leq n^2 \max_\kappa \mathbb{E}\, D_\kappa^2 \dot{a}_\kappa^2.$$

So it suffices to show $\mathbb{E}\, D_\kappa^2 \dot{a}_\kappa^2$ is bounded by Cn^{-3} where C is a constant independent of κ and $t \in [0, 1]$. But by Cauchy-Schwarz,

$$\mathbb{E}\, D_\kappa^2 \dot{a}_\kappa^2 \leq \sqrt{\mathbb{E}\, D_\kappa^4} \cdot \sqrt{\mathbb{E}\, \dot{a}_\kappa^4} = n^{-2} \cdot O(n^{-1}) = O(n^{-3})$$

where we used Eq. (12) and Lemma 5.3. The hidden constant is independent of κ and t, as desired.

A.4 Bounding $\sum_{\kappa\neq\lambda} D_\kappa D_\lambda \dot{a}_\kappa \dot{a}_\lambda$

Because (κ, λ) ranges over a set of size $\leq N^2 = n^4$, we have

$$\mathbb{E} \sum_{\kappa\neq\lambda} D_\kappa D_\lambda \dot{a}_\kappa \dot{a}_\lambda \leq n^4 \max_{\kappa\neq\lambda} \mathbb{E}\, D_\kappa D_\lambda \dot{a}_\kappa \dot{a}_\lambda.$$

So it suffices to show $\mathbb{E}\, D_\kappa D_\lambda \dot{a}_\kappa \dot{a}_\lambda$ is bounded by Cn^{-5} where C is a constant independent of κ, λ (as long as $\kappa \neq \lambda$) and $t \in [0, 1]$.

Taylor Expansion Now $D_\kappa D_\lambda$ is a function of a. In particular, we will think of it as a function of a_κ and a_λ, keeping other entries of a fixed. To this end, we will write $D_\kappa D_\lambda = f(a_\kappa, a_\lambda)$. To reduce the amount of subscripts, denote $y = a_\kappa, z = a_\lambda$. Then Taylor expanding f to some order K to be specified below, we have

$$f(y,z) = \sum_{i+j\le K} y^i z^j \Delta_{ij} + \sum_{i+j=K+1} y^i z^j R_{ij}(y,z)$$

where $\Delta_{ij} = \frac{1}{(i+j)!}\binom{i+j}{i}\partial_y^i \partial_z^j f(0,0)$ and $R_{ij}(y,z) = \frac{1}{K!}\binom{K+1}{i}\int_0^1 (1-\xi)^K \partial_y^i \partial_z^j f(\xi y, \xi z) d\xi$.

This may look like a big scary expression, but as we will see soon enough, the exact form of Δ_{ij} doesn't matter beyond the fact that it's independent from y and z, and we can easily bound R_{ij} using Eq. (12).

Cancellation Using Independence and Zero-Mean Since Δ_{ij} is independent from y and z (as random variables), we have

$$\mathbb{E}\, \dot{y}\dot{z} y^i z^j \Delta_{ij} = (\mathbb{E}\, \dot{y} y^i)(\mathbb{E}\, \dot{z} z^j)\mathbb{E}\, \Delta_{ij}. \tag{13}$$

We see immediately that if $i < 2$ or $j < 2$, then this expectation will vanish due to Lemma 5.3. This motivates us to take K (the order of Taylor expansion) to be 3, so that

$$\mathbb{E}\, \dot{y}\dot{z} f(y,z) = \mathbb{E}\, \dot{y} y^2 \dot{z} z^2 R_{ij}(y,z)$$

Now $R_{ij}(y,z)$ is not independent from y and z unlike Δ_{ij}, but by Cauchy-Schwarz, we have

$$\mathbb{E}\, \dot{y} y^2 \dot{z} z^2 R_{ij}(y,z) \le \sqrt{\mathbb{E}\,(\dot{y} y^2 \dot{z} z^2)^2}\sqrt{\mathbb{E}\, R_{ij}(y,z)^2}$$

By Eq. (12) and the product rule, we see all order-$(K+1)$ (mixed) derivatives of f are bounded by $(K+2)/n^2 = 5/n^2$, so that $R_{ij}(y,z) \le C/n^2$, for some absolute constant C. This implies $\sqrt{\mathbb{E}\, R_{ij}(y,z)^2} \le C/n^2$ as well.

Finally, by Lemma 5.3, $\mathbb{E}\,(\dot{y} y^2 \dot{z} z^2)^2 = \mathbb{E}\,(\dot{y} y^2)^2 \mathbb{E}\,(\dot{z} z^2)^2 = O(n^{-6})$ independent of κ and λ. So altogether,

$$\mathbb{E}\, \dot{y} y^2 \dot{z} z^2 R_{ij}(y,z) \le O(n^{-3})O(n^{-2}) = O(n^{-5})$$

with constant independent of t, κ, and λ. This finishes the proof of Theorem 5.2 with $p = 2$ in our simple scenario.

From this example, one can see the importance of $\mathbb{E}\, a\dot{a} = 0$ in Lemma 5.3: had it not been the case, then Eq. (13) can be nonvanishing for $i = 2, j = 0$, so we can only Taylor expand f to order 1 instead of 3, as a result losing a factor of n in the final bound.

A.5 Going Beyond This Simple Example

To get the full result of Theorem 5.2, we need to extend this argument to 1) all powers p instead of just 2 (i.e., bounding $\mathbb{E}\,|\dot{c}|^p$), and to 2) all polynomially smooth nonlinearities.

For 1), the extension follows more or less the line of reasoning demonstrated here, which can be extracted as a general *moment argument* which we explain in Appendix I.

For 2), significant effort is needed to establish the property corresponding to Eq. (12). In fact, this is the bulk of the technical content of our work, done in Appendix J and Appendix K.1, culminating in Lemma K.6 that generalizes Eq. (12).

B Gaussian smoothing makes polynomially bounded nonlinearities polynomially smooth

Lemma B.1. *Let $\phi : \mathbb{R}^k \to \mathbb{R}$. Then for any $\delta > 0$ there exists $\phi^\delta \in C^\infty(\mathbb{R}^k)$ such that*

1. *For any $x \in \mathbb{R}^k$,*

$$|\phi(x) - \phi^\delta(x)| \le \sup_{x_1, x_2 \in \bar{B}_\delta(x)} |\phi(x_1) - \phi(x_2)|,$$

where $\bar{B}_\delta(x)$ is a closed δ-vicinity of x;

2. *If ϕ is polynomially bounded then ϕ^δ is polynomially smooth;*
3. *If ϕ is Lipschitz with a Lipschitz constant λ, then ϕ^δ is also Lipschitz with a Lipschitz constant λ and*
$$\sup_{x\in\mathbb{R}^k} |\phi(x) - \phi^\delta(x)| \le 2\lambda\delta.$$

Proof. Let ζ be a standard bump function with k arguments:
$$\zeta(z) = Ce^{-\frac{1}{1-\|z\|_2^2}}\mathbb{I}(\|z\|_2 \le 1). \tag{14}$$
It is a non-negative $C^\infty(\mathbb{R}^k)$ function that is supported supported on a unit ball. We choose the constant C so that ζ sums to one.

Claim 1 Let $\zeta^\delta(x) \stackrel{\text{def}}{=} (1/\delta)^k\zeta(x/\delta)$. Set $\phi^\delta(x) \stackrel{\text{def}}{=} [\phi * \zeta^\delta](x)$. Then
$$\begin{aligned}\phi^\delta(x) &= \int_{\mathbb{R}^k} \zeta^\delta(y)\phi(x-y)\,dy \\ &= (1/\delta)^k \int_{\mathbb{R}^k} \zeta(y/\delta)\phi(x-y)\,dy \\ &= \int_{\mathbb{R}^k} \zeta(z)\phi(x-z\delta)\,dz.\end{aligned}$$
We have for any $x \in \mathbb{R}^k$,
$$\begin{aligned}|\phi^\delta(x) - \phi(x)| &= \left|\int_{\mathbb{R}^k} \zeta(z)\left(\phi(x-z\delta) - \phi(x)\right)\,dz\right| \\ &\le \left|\int_{\|z\|_2\le 1} \zeta(z)\,dz\right| \sup_{\|z\|_2\le 1} |\phi(x-z\delta) - \phi(x)| \\ &\le \sup_{x_1,x_2\in\bar{B}_\delta(x)} |\phi(x_1) - \phi(x_2)|.\end{aligned}$$

Claim 2 Suppose ϕ is polynomially bounded, i.e. $|\phi(x)| \le C(1+\|x\|_p^p)$ for some $p \ge 1$. For any $r \ge 0$ and any $j_1, \ldots, j_r \in [k]$,
$$\begin{aligned}\partial_{j_1,\ldots,j_r}\phi^\delta(x) &= (1/\delta)^{k+\sum_{i=1}^r j_i} \int_{\mathbb{R}^k} \partial_{j_1,\ldots,j_r}\zeta((x-y)/\delta)\phi(y)\,dy \\ &= (1/\delta)^{k+\sum_{i=1}^r j_i} \int_{\|y\|_2\le\delta} \partial_{j_1,\ldots,j_r}\zeta(y/\delta)\phi(x-y)\,dy \\ &= (1/\delta)^k \int_{\|z\|_2\le 1} \partial_{j_1,\ldots,j_r}\zeta(z)\phi(x-z\delta)\,dz.\end{aligned}$$
Therefore,
$$\begin{aligned}\left|\partial_{j_1,\ldots,j_r}\phi^\delta(x)\right| &\le (1/\delta)^k \sup_{\|z\|_2\le 1} |\phi(x-z\delta)| \left|\int_{\|z\|_2\le 1} \partial_{j_1,\ldots,j_r}\zeta(z)\,dz\right| \\ &\le C(1/\delta)^k \sup_{\|z\|_2\le 1} (1+\|x-z\delta\|_p^p) \le C(1/\delta)^k \sup_{\|z\|_2\le 1} (1+2^p\|x\|_p^p + 2^p\delta^p\|z\|_p^p) \\ &\le C(1/\delta)^k \sup_{\|z\|_2\le 1} (1+2^p\|x\|_p^p + 2^p\delta^p p^p\|z\|_\infty^p) \le C(1/\delta)^k(1+2^p\|x\|_p^p + (2p\delta)^p),\end{aligned}$$
which implies that $\partial_{j_1,\ldots,j_r}\phi^\delta$ is polynomially bounded.

Claim 3 Suppose ϕ is Lipschitz with a Lipschitz constant λ. Let us check that ϕ^δ is also Lipschitz with a Lipschitz constant λ:
$$\begin{aligned}|\phi^\delta(x_1) - \phi^\delta(x_2)| &= \left|\int_{\mathbb{R}^k} \zeta(z)\left(\phi(x_1-z\delta) - \phi(x_2-z\delta)\right)\,dz\right| \\ &\le \lambda|x_1-x_2| \int_{\mathbb{R}^k} \zeta(z)\,dz = \lambda|x_1-x_2|.\end{aligned}$$

Then for any $x \in \mathbb{R}^k$,

$$|\phi(x) - \phi^\delta(x)| \leq \sup_{x_1, x_2 \in \bar{B}_\delta(x)} |\phi(x_1) - \phi(x_2)| \leq \lambda \sup_{x_1, x_2 \in \bar{B}_\delta(x)} \|x_1 - x_2\|_2 \leq 2\lambda\delta. \tag{15}$$

□

C Additional applications of Theorem 3.7

TP3 demonstrated the following applications of Theorem 3.4: the semi-circle law for Gaussian orthogonal ensembles (GOE), the Marchenko-Pastur law for Gaussian Wishart ensembles, and the free independence principle for neural nets with Gaussian initialization. Here we generalize all the above results to non-Gaussian weight initializations.

C.1 Semicircle law for non-Gaussian Wigner ensembles

The semicircle law for GOE proven in [31] is the following result:

Theorem C.1. *For each $n \geq 1$, define the random matrix $A = W + W^\top$ for iid Gaussian matrix $W \in \mathbb{R}^{n \times n}$, $W_{\alpha\beta} \sim \mathcal{N}(0, 1/2n)$. Let $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$ be the eigenvalues of A and $\mu_n = \frac{1}{n}\sum_{\alpha=1}^n \delta_{\lambda_\alpha}$ be their empirical distribution. Then $\mu_n \xrightarrow{a.s.} \mu_{sc}$, where μ_{sc} is the distribution with density $\propto \sqrt{4 - x^2}$.*

Here by almost sure (weak) convergence of measures we mean that for every compactly supported, continuous $\phi : \mathbb{R} \to \mathbb{R}$, as $n \to \infty$, we have

$$\frac{1}{n}\sum_{\alpha=1}^n \phi(\lambda_\alpha) \xrightarrow{\text{a.s.}} \mathbb{E}_{\lambda \sim \mu_{sc}} \phi(\lambda). \tag{16}$$

In order to be able to apply Tensor Programs machinery to prove this result, [31] used the moment method which states that it suffices to prove almost sure convergene of moments of μ_n:

$$\mathbb{E}_{\lambda \sim \mu_n} \lambda^r = \frac{1}{n}\sum_{\alpha=1}^n \lambda_\alpha^r = n^{-1}\operatorname{tr}(A^r) \xrightarrow{\text{a.s.}} \mathbb{E}_{\lambda \sim \mu_{sc}} \lambda^r \quad \forall r \in \mathbb{N}. \tag{17}$$

The trace can be expressed as $\operatorname{tr}(A^r) = \mathbb{E}_z[z^\top A^r z]$ for $z_\alpha \sim \mathcal{N}(0,1)$ iid for each $\alpha \in [n]$. For Gaussian z, $A^r z$ can be expressed as a vector in the following Tensor Program:

$$g^1 = z, \quad g^2 = Wz, \quad g^3 = W^\top z, \tag{18}$$

$$g^{2k} = W(g^{2k-1} + g^{2k-2}), \quad g^{2k+1} = W^\top(g^{2k-1} + g^{2k-2}) \quad \forall k \in [2 : r]. \tag{19}$$

Here each nonlinearity simply adds two vectors. Note that these nonlinearities are linear polynomials. Then we get

$$\begin{aligned} n^{-1}\operatorname{tr}(A^r) &= \frac{1}{n}\mathbb{E}_{z \sim \mathcal{N}(0,1)}[z^\top A^r z] = \\ &= \frac{1}{n}\sum_{\alpha=1}^n \mathbb{E}_{g_\alpha^1 \sim \mathcal{N}(0,1)}\left[g_\alpha^1\left(g_\alpha^{2r} + g_\alpha^{2r+1}\right)\right] = \frac{1}{n}\sum_{\alpha=1}^n \mathbb{E}_{g_\alpha^1 \sim \mathcal{N}(0,1)} \psi(g_\alpha^1, \ldots, g_\alpha^{2r+1}), \end{aligned} \tag{20}$$

where $\psi(x_1, \ldots, x_{2r+1}) = x_1(x_{2r} + x_{2r+1})$ — a quadratic polynomial.

The only thing that prevents us from directly applying Theorem 3.4 to the expression above is conditional expectation. [31] proves the following conditional theorem:

Theorem C.2 (Gaussian Conditional Master Theorem, [31])**.** *Consider Setup 3.3 and assume ψ to be quadratically bounded and all the nonlinearities to be linearly bounded. Let S be a subset of initial vectors. Then, as $n \to \infty$,*

$$\frac{1}{n}\sum_{\alpha=1}^n \mathbb{E}_S \psi\left(g_\alpha^1, \ldots, g_\alpha^M, c^1, \ldots, c^M\right) \xrightarrow{a.s.} \mathring{\Psi}, \tag{21}$$

where $\mathring{\Psi}$ is the same as in Theorem 3.2.

We had to come back to the original form of Master theorems (see Theorem 3.2) since there is a distinction between nonlinearities (which have to be linearly bounded) and test functions ψ, which could be quadratically bounded.

The above conditional theorem implies $n^{-1}\operatorname{tr}(A^r) \xrightarrow{\text{a.s.}} \mathring{\Psi}$ for certain $\mathring{\Psi}$. Comparing with Eq. (17), what remains to establish Theorem C.1, is to prove that $\mathring{\Psi}$ equals to the r-th moment of the semicircle law. One can find the proof in [31]; we do not reproduce it here.

Non-Gaussian conditional Master theorem. Consider Setup 3.6 and let S be a subset of initial vectors. Then for any scalar c^i in the program, $\mathbb{E}\,|\mathbb{E}_S c^i - \mathring{c}^i|^p \le \mathbb{E}\,|c^i - \mathring{c}^i|^p$, which converges to zero by Theorem 3.7. Therefore $\mathbb{E}_S c^i$ converges to $\mathring{c}^i$ in L^p.

Consider now the same interpolation between Gaussian and non-Gaussian weights $A^l(t)$ as in the proof of Theorem 3.7, see Section 5. Theorem 5.2 then gives $\sup_t \mathbb{E}\,|\mathbb{E}_S c^i(t)|^p \le \sup_t \mathbb{E}\,|c^i(t)|^p = O(n^{-p/2})$ for any $p \in [1,\infty)$. Following the argument in Section 5, we get $\mathbb{E}_S c^i(1) - \mathbb{E}_S c^i(0) \xrightarrow{\text{a.s.}} 0$. Assuming the conditions of the above Gaussian theorem, Theorem C.2, namely, that ϕ^i is quadratically bounded, while ϕ^j for all $j < i$ are linearly bounded, we get $\mathbb{E}_S c^i(0) \xrightarrow{\text{a.s.}} \mathring{c}^i$, and therefore $\mathbb{E}_S c^i(1) \xrightarrow{\text{a.s.}} \mathring{c}^i$:

Theorem C.3 (Non-Gaussian Conditional Master Theorem, ours)**.** *Consider Setup 3.6 and let S be a subset of initial vectors. Then every scalar c^i conditioned on S converges to the same $\mathring{c}^i$ as in Theorem 3.4 in L^p for any $p \in [1,\infty)$:*

$$\mathbb{E}_S c^i \xrightarrow{L^p} \mathring{c}^i \quad \forall p \in [1,\infty). \tag{22}$$

If moreover all the nonlinearities are linearly bounded then for any quadratically bounded polynomially smooth ψ,

$$\frac{1}{n}\sum_{\alpha=1}^{n} \mathbb{E}_S \psi\left(g_\alpha^1, \ldots, g_\alpha^M, c^1, \ldots, c^M\right) \xrightarrow{\text{a.s.}} \mathring{\Psi} \tag{23}$$

as $n \to \infty$, where $\mathring{\Psi}$ is the same as in Theorem 3.2.

Since the program used to compute the moments conforms not only Setup 3.3 but also a stronger Setup 3.6, we get a full analogue of Theorem C.1:

Theorem C.4. *For each $n \ge 1$, define the random matrix $A = W + W^\top$ for W being an $n \times n$ matrix with iid entries with zero mean, variance $1/(2n)$, and with all higher moments existing. Let $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$ be the eigenvalues of A and $\mu_n = \frac{1}{n}\sum_{\alpha=1}^{n} \delta_{\lambda_\alpha}$ be their empirical distribution. Then $\mu_n \xrightarrow{a.s.} \mu_{sc}$, where μ_{sc} is the distribution with density $\propto \sqrt{4-x^2}$.*

Relaxing boundedness. Could we relax the assumption on linear and quadratic boundedness in our Theorem C.3? Inspired by Central Limit Theorem and our matrix derivative bound below, Lemma K.6, we conjecture the following moment bound that quantifies the rate of L^p convergence:

Conjecture C.5. *Under Setup C.6 below, $\mathbb{E}\,|c^i - \mathring{c}^i|^p = O(n^{-p/2})$ for any scalar c^i in the program and the corresponding almost sure limit $\mathring{c}^i$, and any $p \in [1,\infty)$.*

Setup C.6. *Consider Setup 3.6 but replace 5) with 5*) for any initial scalar c^i and any $p \in [1,\infty)$, $\mathbb{E}\,|c^i - \mathring{c}^i|^p = O(n^{-p/2})$, where $\mathring{c}^i$ is the almost sure limit of c^i which exists due to 2).*

If the above conjecture holds, we will get a similar bound for scalars conditioned on S, i.e. $\mathbb{E}\,|\mathbb{E}_S c^i - \mathring{c}^i|^p = O(n^{-p/2})$, which will imply $\mathbb{E}_S c^i \xrightarrow{\text{a.s.}} \mathring{c}^i$ by Borel-Cantelli lemma:

Conjecture C.7. *Consider Setup C.6 and let S be a subset of initial vectors. Then every scalar c^i conditioned on S converges to the same $\mathring{c}^i$ as in Theorem 3.4 almost surely and in L^p for any $p \in [1,\infty)$:*

$$\mathbb{E}_S c^i \xrightarrow{\text{a.s. \& } L^p} \mathring{c}^i \quad \forall p \in [1,\infty). \tag{24}$$

We leave proving Conjecture C.5 for future work.

C.2 Marchenko-Pastur law for non-Gaussian Wishart ensembles

Using exactly the same machinery as for the semi-circle law, [31] proves the Marchenko-Pastur law for Wishart ensembles, i.e. for matrices of the form $AA^\top$, where A is an $m \times n$ Gaussian matrix with zero mean and variance n^{-1}, and $m/n \to \rho \in (0, \infty)$ as $n \to \infty$. Our Theorem 3.7 (with a remark on programs with variable dimensions, see Section 3) gives a similar result with no assumptions on Gaussianity of A:

Theorem C.8 (Ours)**.** *For each $n \geq 1$, let A be an $m \times n$ matrix with iid entries with zero mean, variance $1/n$, and with all higher moments existing. Let $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$ be the eigenvalues of $AA^\top$ and $\mu_n = \frac{1}{n}\sum_{\alpha=1}^{n} \delta_{\lambda_\alpha}$ be their empirical distribution. Let m go to infinity as $n \to \infty$ such that $m/n \to \rho \in (0, \infty)$. Then $\mu_n \xrightarrow{a.s.} \mu_{mp}$, where μ_{mp} has "density" $p_{mp}(x)$ given below:*

$$p_{mp}(x) = \max(0, 1 - \rho^{-1})\delta(x) + \frac{1}{\rho 2\pi x}\sqrt{(b-x)(x-a)}1_{x\in[a,b]}, \tag{25}$$

where $\delta(x)$ is the Dirac Delta, $a = (1 - \sqrt{\rho})^2$, and $b = (1 + \sqrt{\rho})^2$.

C.3 Free Indepedence Principle for Tensor Programs with non-Gaussian weights

Using the same machinery again, [31] proves Free Independence Principle for Tensor Programs:

Theorem C.9 ([31])**.** *Consider Setup 3.3 and assume all nonlinearities to be linearly bounded. Let $\mathcal{D}$ denote the collection of diagonal matrices formed from bounded coordinatewise images of vectors in the program:*

$$\mathcal{D} = \left\{\operatorname{diag}\left(\psi\left(g^1, \ldots, g^M\right)\right) : \ \psi : \mathbb{R}^M \to \mathbb{R} \text{ is bounded}\right\}. \tag{26}$$

Then $\mathcal{D}$, along with the random matrix collections $\{A, A^\top\}$ for all matrices in the program are almost surely asymptotically free as $n \to \infty$.

Here by almost sure asymptotical freeness we mean the following:

Definition C.10. Fix k. Consider collections of random matrices $\mathcal{W}_n^1, \ldots, \mathcal{W}_n^k \subseteq \mathbb{R}^{n\times n}$ for each $n \geq 1$, of constant cardinalities (with n). We say $\mathcal{W}_n^1, \ldots, \mathcal{W}_n^k$ are almost surely asymptotically freely independent (or just almost surely asymptotically free), if

$$n^{-1}\operatorname{tr}\left(\prod_{i=1}^{k}\left(P_i(\mathcal{W}_n^{j_i}) - \tau_i I\right)\right) \xrightarrow{\text{a.s.}} 0, \tag{27}$$

where $\tau_i = n^{-1}\operatorname{tr}(P_i(\mathcal{W}_n^{j_i}))$, P_i is a non-commutative polynomial in $|\mathcal{W}_n^{j_i}|$ variables, $j_1, \ldots, j_k \in [k]$ are indices with no two adjacent j_i equal, and $\{P_i\}_i$, $\{j_i\}_i$ independent on n.

The proof strategy for Theorem C.9 is very similar to the one of Theorem C.1. The trace in each τ_i can be expressed as an expectation $\mathbb{E}_{z\sim\mathcal{N}(0,1)}[z^\top P_i(\mathcal{W}_n^{j_i})z]$ and $P_i(\mathcal{W}_n^{j_i})z$ can be expressed as a vector in a program with linearly bounded nonlinearities (that's the reason why we require ψ in the definition of $\mathcal{D}$ to be bounded). Therefore each τ_i converges by Theorem C.2, and therefore can be thought as a scalar in a new Tensor Program. The "outer" trace in Eq. (27) can be expressed as conditional expectation using the same trick, and the corresponding Eq. (27) again converges by Theorem C.2. We refer the reader to [31] for details.

Simply replacing Theorem C.2 in the above reasoning with its non-Gaussian analogue, Theorem C.3, we get a non-Gaussian analogue of Theorem C.9:

Theorem C.11 (Ours)**.** *Consider Setup 3.6 and assume all nonlinearities to be linearly bounded. Let $\mathcal{D}$ denote the collection of diagonal matrices formed from bounded polynomially smooth coordinatewise images of vectors in the program:*

$$\mathcal{D} = \left\{\operatorname{diag}\left(\psi\left(g^1, \ldots, g^M\right)\right) : \ \psi : \mathbb{R}^M \to \mathbb{R} \text{ is bounded and polynomially smooth}\right\}. \tag{28}$$

Then $\mathcal{D}$, along with the random matrix collections $\{A, A^\top\}$ for all matrices in the program are almost surely asymptotically free as $n \to \infty$.

We had to require all nonlinearities to be linearly bounded and the functions ψ in the definition of diagonal matrices $\mathcal{D}$ to be bounded in order to apply a conditional theorem, Theorem C.2 or Theorem C.3. Note that our original non-Gaussian Master theorem, Theorem 3.7, gives also convergence in L^1, which implies convergence of (full, non-conditional) expectations without requiring linearly bounded nonlinearities[23]. Convergence of full expectations allows us to prove more traditional asymptotic freeness in expectation in our non-Gaussian case:

Definition C.12. Fix k. Consider collections of random matrices $\mathcal{W}_n^1, \ldots, \mathcal{W}_n^k \subseteq \mathbb{R}^{n\times n}$ for each $n \geq 1$, of constant cardinalities (with n). We say $\mathcal{W}_n^1, \ldots, \mathcal{W}_n^k$ are asymptotically freely independent in expectation (or just asymptotically free), if

$$n^{-1}\mathbb{E}\left[\operatorname{tr}\left(\prod_{i=1}^{k}\left(P_i(\mathcal{W}_n^{j_i}) - \tau_i I\right)\right)\right] \to 0, \tag{29}$$

where $\tau_i = n^{-1}\mathbb{E}\ \operatorname{tr}(P_i(\mathcal{W}_n^{j_i}))$, P_i is a non-commutative polynomial in $|\mathcal{W}_n^{j_i}|$ variables, $j_1, \ldots, j_k \in [k]$ are indices with no two adjacent j_i equal, and $\{P_i\}_i$, $\{j_i\}_i$ independent on n.

Theorem C.13 (Ours)**.** *Consider Setup 3.6. Let $\mathcal{D}$ denote the collection of diagonal matrices formed from polynomially smooth (not necessarily bounded) coordinatewise images of vectors in the program:*

$$\mathcal{D} = \left\{\operatorname{diag}\left(\psi\left(g^1, \ldots, g^M\right)\right) :\ \psi : \mathbb{R}^M \to \mathbb{R} \text{ is polynomially smooth}\right\}. \tag{30}$$

Then $\mathcal{D}$, along with the random matrix collections $\{A, A^\top\}$ for all matrices in the program are asymptotically free in expectation as $n \to \infty$.

We use the same tensor program to compute $c^i = \frac{1}{n}\sum_{\alpha\in[n]} z^i_\alpha (P_i(\mathcal{W}_n^{j_i})z^i)_\alpha$ required to compute the traces $\tau_i = \mathbb{E}_{z^i\sim\mathcal{N}(0,I)} c^i$. Since we now rely on a theorem that does not require linearly bounded nonlinearities, the nonlinearities of the program at hand and ψ in the definition of $\mathcal{D}$ can be polynomially smooth. We embed the programs for $P_i(\mathcal{W}_n^{j_i})z$ into the "outer" program that computes the scalar $\frac{1}{n}\sum_{\alpha\in[n]} \bar{z}_\alpha \left(\prod_{i=1}^k \left(P_i(\mathcal{W}_n^{j_i}) - c_i I\right)\bar{z}\right)_\alpha$. Taking the expectation over all randomness and applying Theorem C.3 gives Eq. (29).

D Non-Gaussian Master theorem for Lipschitz nonlinearities

One of the limitations of our Theorem 3.7 is smoothness requirement. In the present section, we prove a similar result that assumes nonlinearities to be Lipschitz but not necessarily smooth:

Setup D.1. *Consider Setup 3.6, but replace 3*) and 4*) with the following: 3**) all matrices A^i have independent entries drawn from sub-Gaussian distributions with zero mean and variance n^{-1}; 4**) all the nonlinearities ϕ are Lipschitz (but not necessarily smooth).*

Theorem D.2 (Non-Gaussian Master theorem for Lipschitz nonlinearities)**.** *Consider Setup D.1. Then, as $n \to \infty$, every scalar c^i converges to the same $\mathring{c}^i$ as in Theorem 3.4 almost surely and in mean:*

$$c^i \xrightarrow{\text{a.s. \& } L^1} \mathring{c}^i.$$

The following result is an immediate consequence of the above theorem:

Corollary D.3 (Convergence to GP at initialization)**.** *Consider a neural network whose forward pass can be expressed as Eq.* (4) *with each matrix W^i corresponding to some A^j but not its transposed. Suppose 1) all the activation functions are Lipschitz; 2) input and output layer weights are initialized with iid standard Gaussian images; 3) entries of all hidden bias vectors are initialized with iid standard Gaussians; 4) weights of any other layer are initialized according to 3**) of Setup D.1; 5) output layer bias vector entries are initialized with zeros. Then at initialization, as width tends to infinity, the pre-activation output of any hidden neuron of any layer except for the first one converges weakly to a Gaussian Process (GP).*

[23] This proves Conjecture A.4 of [31] for polynomially smooth nonlinearities

Proof. We use the fact that when all matrices in Eq. (4) are not transposed, the limit in Theorem 3.4 (and hence in our Theorem D.2) takes the form $\mathring{\Psi} = \mathbb{E}_{z\sim\mathcal{N}(\mu,\Sigma)}\psi(z)$, where μ and Σ are computed using a certain recurrent formula; see [29].

Let $g^1, \dots, g^B$ be the pre-activation outputs of a given layer on a batch of inputs of size B. Let ψ be Lipschitz and bounded, and depend only on these B vectors. Then for any $\alpha \in [n]$, by symmetry of neurons, $\mathbb{E}\left[\psi(g^1_\alpha, \dots, g^B_\alpha)\right] \to \mathbb{E}_{z\sim\mathcal{N}(\mu,\Sigma)}\psi(z)$ from Theorem D.2. This means that on any batch of inputs of size B, outputs of a given neuron $g^1_\alpha, \dots, g^B_\alpha$ converge weakly to a Gaussian vector. This means that the output of this neuron converges weakly to a Gaussian process as a function of network's input. □

Proof of Theorem D.2. Let λ be the maximal Lipschitz constant among all ϕ^i in the program. Fix some $\delta > 0$. Let $\phi^{i,\delta}$ be the corresponding polynomially smooth Lipschitz functions given by Lemma B.1. W.l.o.g., we substitute δ with $\delta/2L$ so that $\sup_{x\in\mathbb{R}^{2(i-1)}} |\phi^{i,\delta}(x) - \phi^i(x)| \le \delta$.

Define the "smoothed" version of the given Tensor Program:

$$g^{i,\delta}_\alpha \leftarrow \sum_{\beta=1}^n W^i_{\alpha\beta} x^{i,\delta}_\beta, \quad c^{i,\delta} \leftarrow \frac{1}{n}\sum_{\beta=1}^n x^{i,\delta}_\beta, \quad \text{where } x^{i,\delta}_\alpha = \phi^i(g^{1,\delta}_\alpha, \dots, g^{i-1,\delta}_\alpha; c^{1,\delta}, \dots, c^{i-1,\delta}). \tag{31}$$

for $i \in [M_0 + 1 : M]$, where all W^i are the same as in the original program. All input vectors and scalars coincide with the original program: $g^{i,\delta} = g^i$, $c^{i,\delta} = c^i$ for $i \in [M_0]$.

We consider the same weight interpolation $A^l(t)$ as in the main: $A^l(0)$ corresponds to Gaussian weights $\tilde{A}^l$ with the same mean and variance as A^l, while $A^l(1)$ corresponds to the original weights A^l. Consequently, $g^{i,\delta}(t)$ denotes a vector obtained using $A^1(t), \dots, A^L(t)$, and similarly for $c^{i,\delta}(t)$.

Lemma D.4. *Under premise of Theorem D.2, for any $i \in [M_0 + 1, M]$, there exists a polynomial P^i such that for any $n \ge 1$, any $\delta > 0$, and any $t \in [0, 1]$,*

$$\left\|x^i(t) - x^{i,\delta}(t)\right\|_2 \le \sqrt{n}\delta P^i\left(\lambda\|A^1(t)\|_2, \dots, \lambda\|A^L(t)\|_2\right). \tag{32}$$

Proof. We will drop the t-argument in the proof to lighten the exposition. We prove the statement by induction on i.

Induction base: since $g^1 = g^{1,\delta}, \dots, g^{M_0} = g^{M_0,\delta}$,

$$\begin{aligned}&\left\|x^{M_0+1}(t) - x^{M_0+1,\delta}(t)\right\|_2 \le \\ &\le \left\|\phi^{M_0+1}(g^1, \dots, g^{M_0}; c^1, \dots, c^{M_0}) - \phi^{M_0+1,\delta}(g^{1,\delta}, \dots, g^{M_0,\delta}, c^{1,\delta}, \dots, c^{M_0,\delta})\right\|_2 \le \sqrt{n}\delta.\end{aligned} \tag{33}$$

Here $P^{M_0+1} \equiv 1$; in particular, it does not depend on t.

Suppose the induction hypothesis holds for i. Then we get the following for $i + 1$:

$$\begin{aligned}\left\|x^{i+1} - x^{i+1,\delta}\right\|_2 &= \left\|\phi^i(g^1, \dots, g^i; c^1, \dots, c^i) - \phi^{i,\delta}(g^{1,\delta}, \dots, g^{i,\delta}; c^{1,\delta}, \dots, c^{i,\delta})\right\|_2 \le \\ &\le \left\|\phi^i(g^1, \dots, g^i; c^1, \dots, c^i) - \phi^{i,\delta}(g^1, \dots, g^i; c^1, \dots, c^i)\right\|_2 + \\ &+ \left\|\phi^{i,\delta}(g^1, \dots, g^i; c^1, \dots, c^i) - \phi^{i,\delta}(g^{1,\delta}, \dots, g^{i,\delta}; c^{1,\delta}, \dots, c^{i,\delta})\right\|_2 \le \\ &\le \sqrt{n}\delta + \sqrt{\sum_{\alpha=1}^n \left|\phi^{i,\delta}(g^1_\alpha, \dots, g^i_\alpha; c^1, \dots, c^i) - \phi^{i,\delta}(g^{1,\delta}_\alpha, \dots, g^{i,\delta}_\alpha; c^{1,\delta}, \dots, c^{i,\delta})\right|^2},\end{aligned} \tag{34}$$

where the last inequality holds due to the approximation property of ϕ^δ.

$$\sum_{\alpha=1}^{n} \left|\phi^{i,\delta}(g_\alpha^1,\dots,g_\alpha^i;c^1,\dots,c^i) - \phi^{i,\delta}(g_\alpha^{1,\delta},\dots,g_\alpha^{i,\delta};c^{1,\delta},\dots,c^{i,\delta})\right|^2 \le$$

$$\le \lambda^2 \sum_{\alpha=1}^{n} \sum_{i'=M_0+1}^{i} \left(\left|g_\alpha^{i'} - g_\alpha^{i',\delta}\right|^2 + |c^{i'} - c^{i',\delta}|^2\right) =$$

$$= \lambda^2 \sum_{i'=M_0+1}^{i} \left(\|g^{i'} - g^{i',\delta}\|_2^2 + n|c^{i'} - c^{i',\delta}|^2\right) \le \lambda^2 \sum_{i'=M_0+1}^{i} \left(\|W^{i'}\|_2^2 + 1\right) \left\|x^{i'} - x^{i',\delta}\right\|_2^2 \le$$

$$\le n\delta^2\lambda^2 \sum_{i'=M_0+1}^{i} \left(\|W^{i'}\|_2^2 + 1\right) \left(P^{i'}(\lambda\|A^1\|_2,\dots,\lambda\|A^L\|_2)\right)^2 \quad (35)$$

by induction hypothesis. Plugging this expression back, we get

$$\left\|x^{i+1} - x^{i+1,\delta}\right\|_2 \le \sqrt{n}\delta \left(1 + \lambda\sqrt{\sum_{i'=M_0+1}^{i} (\|W^{i'}\|_2^2 + 1)\left(P^{i'}(\lambda\|A^1\|_2,\dots,\lambda\|A^L\|_2)\right)^2}\right) \le$$

$$\le \sqrt{n}\delta \left(1 + \lambda \sum_{i'=M_0+1}^{i} \left(\|W^{i'}\|_2 + 1\right) P^{i'}(\lambda\|A^1\|_2,\dots,\lambda\|A^L\|_2)\right). \quad (36)$$

Define $j_{i'}$ such that $W^{i'}$ equals $A^{j_{i'}}$ or its transposition for every $i' \in [M_0+1, M]$. Define the new polynomial P^{i+1} as

$$P^{i+1}(x_1,\dots,x_L) = 1 + \lambda \sum_{i'=M_0+1}^{i} \left(x_{j_{i'}} + 1\right) P^{i'}(x_1,\dots,x_L), \quad (37)$$

which does not depend on t since neither of $P^{i'}$ for $i' \le i$ do. This proves the induction. □

The above lemma implies for any $t \in [0,1]$,

$$|c^i(t) - c^{i,\delta}(t)| = \frac{1}{n}\|x^i(t) - x^{i,\delta}(t)\|_1 \le \frac{1}{\sqrt{n}}\|x^i(t) - x^{i,\delta}(t)\|_2 \le$$

$$\le \delta P^i(\lambda\|A^1(t)\|_2,\dots,\lambda\|A^L(t)\|_2). \quad (38)$$

For any $p \in [1,\infty)$,

$$\mathbb{E}\,|c^i(1) - c^i(0)|^p \le 3^p\mathbb{E}\,|c^i(1) - c^{i,\delta}(1)|^p + 3^p\mathbb{E}\,|c^{i,\delta}(1) - c^{i,\delta}(0)|^p + 3^p\mathbb{E}\,|c^i(0) - c^{i,\delta}(0)|^p. \quad (39)$$

The second term goes to zero as $n \to \infty$ for any $\delta > 0$ by our "smooth" Master theorem, Theorem 3.7 (see also the reasoning in Section 5). The first and the last terms are bounded as follows:

$$\mathbb{E}\,|c^i(1) - c^{i,\delta}(1)|^p \le \delta^p\mathbb{E}\left[\left(P^i(\lambda\|A^1\|_2,\dots,\lambda\|A^L\|_2)\right)^p\right], \quad (40)$$

$$\mathbb{E}\,|c^i(0) - c^{i,\delta}(0)|^p \le \delta^p\mathbb{E}\left[\left(P^i(\lambda\|\tilde{A}^1\|_2,\dots,\lambda\|\tilde{A}^L\|_2)\right)^p\right], \quad (41)$$

which are both bounded uniformly wrt n by $C\delta^p$ for some constant C independent on n since all $A^1,\dots,A^L$ and $\tilde{A}^1,\dots,\tilde{A}^L$ are sub-Gaussian and therefore have moments, which are uniformly bounded wrt n. Therefore

$$\lim_{n\to\infty} \mathbb{E}\,|c^i(1) - c^i(0)|^p \le 2C\delta^p. \quad (42)$$

Taking infium over $\delta > 0$ gives simply $\lim_{n\to\infty} \mathbb{E}\,|c^i(1) - c^i(0)|^p = 0$, which means that $c^i(1) - c^i(0)$ converges to zero in L^p. Since this is true for any $p \ge 1$, similarly to the proof of Theorem 3.7, Borel-Cantelli lemma will imply that $c^i(1) - c^i(0)$ converges to zero almost surely. Since Gaussian Master theorem, Theorem 3.4, works for non-smooth nonlinearities, $c^i(1)$ converges almost surely to the same limit as $c^i(0)$.

Since all of our nonlinearities are Lipschitz, they are linearly bounded. For such nonlinearities, Gaussian Master theorem guarantees also convergence in mean, see Theorem A.1 of [31]. Therefore

$$\lim_{n\to\infty} \mathbb{E}\,|c^i(1) - \mathring{c}^i| \leq \lim_{n\to\infty} \mathbb{E}\,|c^i(1) - c^i(0)| + \lim_{n\to\infty} \mathbb{E}\,|c^i(0) - \mathring{c}^i| = 0, \tag{43}$$

which means that $c^i(1)$ converges to $\mathring{c}^i$ in mean.

□

E Tensor Program formulation equivalence

The Tensor Program series [29, 30, 31, 34, 32, 33] defines a Tensor Program as a pair of initial state and a sequence of commands. The initial state consists of variables of three different types: A, G, and C, which correspond to matrices, vectors, and scalars in Eq. (4), respectively. In its body, the program can also generate X-vars (that correspond to a vector x in Eq. (4)), which are size-n vectors but with a different meaning compared to G-vars. The G- and C-vars in the initial state are called input variables.

Each command takes some variables from the state, generates a new variable, and appends it to the state. In the most general version of a Tensor Program, NETSOR$\top^+$, the following commands are available:

- Trsp. Input: A : A. Output $A^\top$: A.
- MatMul. Input: A : A, x : X. Output: Ax : G.
- Nonlin$^+$. Input: g^1 : G, $\ldots, g^k$: G, c^1 : C, $\ldots, c^l$: C. Output: x : X, where $x_\alpha = \phi(g^1_\alpha, \ldots, g^k_\alpha, c^1, \ldots, c^l)\ \forall \alpha \in [n]$.
- Moment. Input: g^1 : G, $\ldots, g^k$: G, c^1 : C, $\ldots, c^l$: C. Output: c : C, where $c = \frac{1}{n}\sum_{\alpha=1}^n \phi(g^1_\alpha, \ldots, g^k_\alpha, c^1, \ldots, c^l)$.

It is easy to see that the iteration of Eq. (4) can be expressed as a NETSOR$\top^+$ program above. Note that in all Master theorems, we care only about vectors of type G and scalars. Let us now show that for any NETSOR$\top^+$ program, we can construct an iteration in the form of Eq. (4) that generates the same set of G- and C-vars.

Note that G-vars can only be generated by MatMul. Each such MatMul uses a single X-var which can be generated only by Nonlin$^+$. For the m-th G-var, we therefore get the following iteration:

$$g^m_\alpha = \sum_{\beta=1}^n A^{m,T_m}_{\alpha\beta}\, \phi^m(g^1_\beta, \ldots, g^{k_m}_\beta, c^1, \ldots, c^{l_m}),$$

where we put $T_m = 1$ if the corresponding A-var has undergone Trsp even number of times by the moment of generating the above G-var, or $T_m = \top$ otherwise. By generating "placeholder" G- or C-vars (say, zeros), we can assume w.l.o.g. that $k_m = l_m = m - 1$. Moreover, we can generate a G-var and a C-var at the same time using the same function ϕ^m as a subsequent nonlinearity may not depend on one of them if not necessary. This gives us an iteration of the form of Eq. (4).

F Comparison with Chen and Lam [2]

The work of [2] considers the following iteration:

$$\tilde{x}^2_\alpha = \phi^2\left(\sum_\delta \tilde{A}_{\alpha\beta} x^1_\beta\right), \quad \tilde{x}^m_\alpha = \phi^m\left(\sum_\delta \tilde{A}_{\alpha\beta}\tilde{x}^{m-1}_\beta, \tilde{x}^{m-2}_\alpha, \ldots, \tilde{x}^2_\alpha, x^1_\alpha\right) \quad \forall m > 2\ \forall \alpha \in [n], \tag{44}$$

where $\tilde{A}$ is a sum of two **symmetric** $n \times n$ matrices: one has iid sub-Gaussian entries with zero mean and variance n^{-1}, while the other is a deterministic matrix divided by n. Denote the first matrix as $\tilde{Z}$, and the second as X/n, so $\tilde{A} = \tilde{Z} + X/n$. Let $A = Z + X/n$, where Z is a symmetric iid Gaussian

matrix with zero mean and variance n^{-1}. Consider the corresponding iteration:

$$x^2_\alpha = \phi^2\left(\sum_\delta A_{\alpha\beta} x^1_\beta\right), \quad x^m_\alpha = \phi^m\left(\sum_\delta A_{\alpha\beta} x^{m-1}_\beta, x^{m-2}_\alpha, \ldots, x^2_\alpha, x^1_\alpha\right) \quad \forall m > 2\ \forall \alpha \in [n], \tag{45}$$

The main result of [2] follows:

Theorem F.1 ([2])**.** *Take $M \geq 2$ and a function ψ with M arguments. Suppose ψ and all ϕ^m for $m \in [2 : M]$ are Lipschitz. Then*

$$\lim_{n\to\infty}\left|\frac{1}{n}\sum_{\alpha=1}^{n}\left(\psi(x^1_\alpha, \ldots, x^M_\alpha) - \psi(\tilde{x}^1_\alpha, \ldots, \tilde{x}^M_\alpha)\right)\right| = 0 \tag{46}$$

in probability.

In this formulation, $\frac{1}{n}\sum_{\alpha=1}^{n}\psi(x^1_\alpha, \ldots, x^M_\alpha)$ does not converge to $\mathbb{E}_{z\sim\mathcal{N}(\mu,\Sigma)}\psi(z)$ for some μ and Σ since $x^1_\alpha, \ldots, x^M_\alpha$ are images of nonlinear functions. Therefore the above formulation does not allow for a clear Gaussian process interpretation as in our Corollary 4.3 or Corollary D.3.

One can show equivalence of our iteration (4) and the above iteration in a certain scenario. Namely for the above, let X be and identity matrix and assume x^1 is an elementwise image of a Gaussian vector. For our iteration (4), let $M_0 = 1$, let us generate a "twin" variable for each vector, where the only difference is that one uses A^m, while the other uses $A^{m,\top}$, and let each nonlinearity use a sum of x^j_α, $\bar{x}^j_\alpha$, and c^j for each $j \in [m-1]$, where $\bar{x}^j$ is a twin variable for x^j. Equivalence is then showed by reorganizing nonlinearities of the two variants.

Apart from equivalence in the above scenario, the use of iteration (44) for expressing deep learning computations is very limited. First, iteration (44) assumes weight matrices to be symmetric, while such weight initializations are rarely used in practice. Second, iteration (44) applies the same weight matrix each time. While some layers in neural nets can indeed share weights, assuming that all layers share weights strictly limits us to vanilla recurrent neural nets, drawing out even simplest feedforward nets. Third, the form $A = Z + X/n$ has no clear interpretation in typical neural network computations. And last, iteration (44) assumes only one input variable. In terms of neural network computations, this limits us to a single input, no bias vectors, and no output layer. It automatically draws out the ability to express a backward pass (and therefore learning process). Moreover, we cannot hope even to show Gaussian process behavior (as in Corollary 4.3) as it requires evaluating the network on a batch of inputs of any finite size.

While our proof is based on the same weight interpolation idea as the proof of Theorem F.1, certain crucial details are different. While Theorem F.1 requires everything to be Lipschitz, our Theorem 3.7 allows for certain smooth polynomially bounded nonlinearities and ψ-functions. While restricting oneself to Lipschitz nonlinearities still allows to express a forward pass of a, say, ReLU net, it does not allow for expressing its backward pass since the derivative of a ReLU is not Lipschitz (and not even continuous). Moreover, computing a kernel (NNGP or NTK, see Corollary 4.3 and Corollary 4.4) requires a quadratic ψ-function. Finally, they prove only convergence in probability, while our Theorem 3.7 states almost sure convergence and convergence in L^p for any p, which is much stronger.

G Proof of Lemma 5.3

We prove Lemma 5.3, recalled here for convenience.

Lemma 5.3 (Interpolation Properties)**.** *For any matrix entry $a(t) = A^i_{\alpha\beta}(t)$ of a program in Setup 3.6: (1) $\mathbb{E}\,\dot{a}(t) = \mathbb{E}\,a(t)\dot{a}(t) = 0$ for all t; (2) For any integers $j, k \geq 0$ with sum $\ell = j + k$, $\sup_t \mathbb{E}\,|a(t)^j \dot{a}(t)^k| \leq \pi^\ell \nu_\ell n^{-\ell/2}$, where ν_ℓ is the scaled moment bound in Setup 3.6.*

Beyond our TP setting, our proof shows that these two properties hold for any interpolation $a(t) = a_1 \cos\frac{\pi}{2}t + a_2 \sin\frac{\pi}{2}t$ between centered random variables a_1 and a_2 with identical variance $1/n$ and satisfying the scaled moment bound $\mathbb{E}\,|a_1|^k, \mathbb{E}\,|a_2|^k \leq \nu_k/n^{-k/2}$ for all $k \geq 3$.

Proof. Write

$$a(t) = \tilde{a}\cos\frac{\pi}{2}t + a\sin\frac{\pi}{2}t$$

where $\tilde{a}$ is the Gaussian random variable and a is the non-Gaussian random variable. Then

$$\dot{a}(t) = \frac{\pi}{2}(-\tilde{a}\sin\frac{\pi}{2}t + a\cos\frac{\pi}{2}t).$$

Then $\mathbb{E}\,\dot{a}(t) = 0$ follows from the zero-mean property of $\tilde{a}$ and a. Likewise, straightforward calculation using the independence between $\tilde{a}$ and a shows $\mathbb{E}\,a(t)\dot{a}(t) = 0$.

To show (2), first note that, for any t,

$$|a(t)|, \frac{2}{\pi}|\dot{a}(t)| \leq |\tilde{a}| + |a|.$$

Then

$$\mathbb{E}\,|a(t)^j\dot{a}(t)^k| \leq \left(\frac{\pi}{2}\right)^k \sum_{i=0}^{k}\binom{\ell}{i}\mathbb{E}\,|\tilde{a}|^i|a|^{\ell-i}.$$

By Hölder's inequality,

$$\mathbb{E}\,|\tilde{a}|^i|a|^{\ell-i} \leq (\mathbb{E}\,|\tilde{a}|^\ell)^{\frac{i}{\ell}}(\mathbb{E}\,|a|^\ell)^{\frac{\ell-i}{\ell}} \leq \nu_\ell n^{-\ell/2}.$$

Therefore,

$$\mathbb{E}\,|a(t)^j\dot{a}(t)^k| \leq \left(\frac{\pi}{2}\right)^k \sum_{i=0}^{k}\binom{\ell}{i}\nu_\ell n^{-\ell/2} = \left(\frac{\pi}{2}\right)^k 2^\ell\nu_\ell n^{-\ell/2} = \pi^\ell\nu_\ell n^{-\ell/2}.$$

□

H Technical Preliminaries

H.1 L^p Norm

For any vector $v \in \mathbb{R}^k$ and $p \geq 1$, we write $\|v\|_p$ to denote its L^p norm $\|v\|_p \stackrel{\text{def}}{=} \sqrt[p]{|v_1|^p + \cdots + |v_k|^p}$. When $p = 2$, we will just write $\|v\| = \|v\|_2$ when there's no cause for confusion. The following L^p norm bound is standard.

Lemma H.1. *For any vector $b \in \mathbb{R}^k$, if $p \leq q$, then*

$$\|b\|_q \leq \|b\|_p \leq k^{\frac{1}{p}-\frac{1}{q}}\|b\|_q$$

We will use the following trivial but useful fact repeatedly. It follows trivially from Lemma H.1.

Lemma H.2. *For any integer $m \geq 0$ and reals $a_i \in \mathbb{R}$, $i \in [k]$,*

$$\left|\sum_{i=1}^{k} a_i\right|^m \leq k^{m-1}\sum_{i=1}^{k}|a_i|^m.$$

H.2 Multisets

Definition H.3 (Multiset)**.** A *multiset* is a set allowing multiple occurences of the same element, e.g., $\{1, 1, 1, 2, 2, 3\}$ (which is not equal to $\{1, 2, 3\}$ as a multiset). We will use capital italic font to denote multisets, such as $\mathcal{P}$. Thus, e.g., when we write $\sum_{p\in\mathcal{P}} f(p)$, p could take the same value multiple times. The number of elements in $\mathcal{P}$, counting multiplicity, is denoted $|\mathcal{P}|$. The set of *unique* elements is denoted $\mathrm{uniq}(\mathcal{P})$.

Definition H.4 (Partition of multiset)**.** A *partition* τ of a multiset $\mathcal{P}$ expresses $\mathcal{P}$ as a disjoint union of multisets. Concretely, τ is a multiset $\{\mathcal{P}_1, \ldots, \mathcal{P}_k\}$ such that $\mathcal{P} = \bigsqcup_i \mathcal{P}_i$, and $|\tau| = k$ is the number of sets in the partition. For example, $\mathcal{P} = \{1, 1, 1, 2, 2, 3\} = \{1, 1\} \sqcup \{2, 2\} \sqcup \{1, 3\}$, so that $\tau = \{\{1, 1\}, \{2, 2\}, \{1, 3\}\}$ is a partition of $\mathcal{P}$. As another example, $\mathcal{P} = \{1\}\sqcup\{1\}\sqcup\{1\}\sqcup\{2\}\sqcup\{2\}\sqcup\{3\}$ is also a partition, with repeated sets, which emphasizes the fact that τ is allowed to be a multiset.

Obviously, the typical notion of a partition of a *set* is just a special case of the above concept applied to sets.

Definition H.5 (Partition induced by multiset)**.** Given a multiset $\mathcal{P}$, we can count the multiplicity of each element in $\mathcal{P}$ and list them in nonincreasing order, say $\pi_1 \geq \pi_2 \geq \cdots \geq \pi_k \geq 1$ where $\sum_{i=1}^k \pi_i = |\mathcal{P}|$. Then the multiset $\pi = \{\pi_i\}_{i=1}^k$ form a *partition* $\pi(\mathcal{P})$ of the integer $|\mathcal{P}|$, which we call the *partition induced by* $\mathcal{P}$, and we denote $|\pi| \stackrel{\text{def}}{=} k$, the *size of the partition*. Again, note that $\pi(\mathcal{P})$ does not contain any information about the identity of elements in $\mathcal{P}$, only their counts.

For example, $\mathcal{P} = \{\bullet, \bullet, \bullet, \star, \star, \times\}$ would induce the partition $|\mathcal{P}| = 6 = 3 + 2 + 1$ because 3 is the multiplicity of $\bullet$, 2 is the multiplicity of $\star$, and 1 is the multiplicity of $\times$.

Note that the partitions of integer n are equivalent to the partitions of the multiset $\{\underbrace{1, \ldots, 1}_{n}\}$.

Definition H.6 (Multiset from vector)**.** A multiset can be obtained from a vector by forgetting the order of the vector's entries, e.g., $\{1, 1, 1, 2, 2, 3\}$ can obtained this way from the vector $(1, 2, 1, 3, 2, 1)$ or $(2, 3, 1, 1, 2, 1)$. We will use capital bold font to denote such vectors in the context of multisets, such as $\mathbf{P} = (P_1, P_2, \ldots)$, and such obtained multiset would be written as $\tilde{\mathbf{P}} = \{P_1, P_2, \ldots\}$.

Definition H.7 ($N_{\mathcal{P}}$)**.** For any fixed multiset $\mathcal{P}$, we can ask how many distinct vectors $\mathbf{P}$ are there such that $\mathcal{P} = \tilde{\mathbf{P}}$. The answer is the multinomial coefficient $N_{\mathcal{P}} \stackrel{\text{def}}{=} \binom{|\mathcal{P}|}{\pi} = \binom{|\mathcal{P}|}{\pi_1; \cdots; \pi_k}$, where $\pi = \pi(\mathcal{P})$ is the partition induced by $\mathcal{P}$. Note that $N_{\mathcal{P}}$ depends on $\mathcal{P}$ only through π.

For instance, for $\mathcal{P} = \{\bullet, \bullet, \bullet, \star, \star, \times\}$, we have $N_{\mathcal{P}} = \binom{6}{3;2;1}$.

H.3 Monomials

Given a vector $\boldsymbol{x} = (x_1, \ldots, x_k) \in \mathbb{R}^k$ and a vector $\mathbf{P} \in [k]^r$ for some integer $r \geq 0$, we can form the monomial $\boldsymbol{x}^{\mathbf{P}} \stackrel{\text{def}}{=} \prod_{i=1}^r x_{P_i}$. Thus, if 1 appears in $\mathbf{P}$ t times, then $\boldsymbol{x}^{\mathbf{P}}$ contains a factor of x_1^t.

If we also have a multiset $\mathcal{P}$ taking values in $[k]$, then we can also form the monomial $\boldsymbol{x}^{\mathcal{P}} \stackrel{\text{def}}{=} \prod_{\alpha \in \mathcal{P}} x_\alpha$. In this notation, we would have $\boldsymbol{x}^{\mathbf{P}} = \boldsymbol{x}^{\tilde{\mathbf{P}}}$.

H.4 Tensors

Given vectors $x \in \mathbb{R}^k, y \in \mathbb{R}^l$, their tensor $x \otimes y$ is the vector in $\mathbb{R}^{k \times l}$ with entries $(x \otimes y)_{(i,j)} = x_i y_j$, where the index ranges over $(i, j) \in [k] \times [l]$. We will use angle brackets $\langle -, - \rangle$ to denote standard inner product on all Euclidean spaces, so that

$$\langle x \otimes y, x' \otimes y' \rangle = \langle x, x' \rangle \langle y, y' \rangle \tag{47}$$

if $x, x' \in \mathbb{R}^k, y, y' \in \mathbb{R}^l$.

We can tensor a vector $x \in \mathbb{R}^k$ with itself $p - 1$ times to form a *tensorial power* $x^{\otimes p} \in \mathbb{R}^{k^p}$ of x. Its entries would be indexed by elements $\mathbf{P}$ of $[k]^p$, with $(x^{\otimes p})_{\mathbf{P}} = x^{\mathbf{P}}$. Like in Eq. (47), we have

$$\langle x^{\otimes p}, y^{\otimes p} \rangle = \langle x, y \rangle^p.$$

for any x, y in the same Euclidean space.

This is particularly relevant as we will often encounter the expression $(\sum_\alpha x_\alpha)^p$ which can be written as

$$\left(\sum_\alpha x_\alpha \right)^p = \langle 1, x \rangle^p = \langle 1^{\otimes p}, x^{\otimes p} \rangle = \sum_{\mathbf{P} \in [k]^p} x^{\mathbf{P}}. \tag{48}$$

By Definition H.7, we can further rewrite this as

$$\left(\sum_\alpha x_\alpha \right)^p = \sum_{\mathcal{P}} N_{\mathcal{P}} x^{\mathcal{P}} \tag{49}$$

where $\mathcal{P}$ ranges over all multisets of size $|\mathcal{P}| = p$ taking value in $[k]$.

Spectral Norms for Tensors Let A be a r-th order tensor, $A \in \mathbb{R}^{k_1 \times \ldots \times k_r}$. We define its spectral norm, $\|A\|$, as follows:

$$\|A\| = \sup_{x_1 \in \mathbb{R}^{k_1},\, \|x_1\|=1} \cdots \sup_{x_r \in \mathbb{R}^{k_r},\, \|x_r\|=1} = \left| \left\langle A, \bigotimes_{i=1}^{r} x_i \right\rangle \right|. \tag{50}$$

Note that this definition is consistent with the definition of spectral norm of matrices. For any r-th order tensor A and vectors $x_1 \in \mathbb{R}^{k_1}, \ldots x_r \in \mathbb{R}^{k_r}$, we have the following inequality:

$$\left| \left\langle A, \bigotimes_{i=1}^{r} x_i \right\rangle \right| \leq \|A\| \prod_{i=1}^{r} \|x_i\|. \tag{51}$$

H.5 Higher Order Differentiation and Taylor Expansion

Scalar Function For a smooth multivariate scalar function $f(x) = f(x_1, \ldots, x_k)$ and a vector $\mathbf{P} \in [k]^r$ with multiset $\mathcal{P} \stackrel{\text{def}}{=} \tilde{\mathbf{P}}$, we write

$$\partial_x^{\mathbf{P}} f(x) = \partial_x^{\mathcal{P}} f(x) \stackrel{\text{def}}{=} \frac{\partial^r}{\partial x_{P_1} \cdots \partial x_{P_r}} f(x).$$

We also define $\nabla_x^p f(x) \in \mathbb{R}^{k^p}$ as the pth order tensor containing all order-p derivatives of f evaluated at x:

$$(\nabla_x^p f(x))_{\mathbf{P}} = \partial_x^{\mathbf{P}} f(x).$$

This is a highly symmetric tensor as $(\nabla_x^p f(x))_{\mathbf{P}} = (\nabla_x^p f(x))_{\mathbf{Q}}$ if $\tilde{\mathbf{P}} = \tilde{\mathbf{Q}}$ as multisets. In all notations, we will drop the subscript, e.g., $\partial^{\mathbf{P}} = \partial^{\mathbf{P}}$, when the variables of the differentiation are clear from context.

Note that we have used a superscript notation for $\partial^{\mathbf{P}}$ and $\partial^{\mathcal{P}}$ to emphasize that they denote a "product of (1st order) partial derivatives operators." In particular, $\partial^{\mathbf{P}} f(x)$ and $\partial^{\mathcal{P}} f(x)$ should be distinguished from the vector of 1st order partial derivatives $(\frac{\partial}{\partial x_{P_i}} f(x))_{i=1}^r$. They will always be scalar quantities when f is scalar, whereas we always use ∇^p for when we want to think of the collection of (higher order) partial derivatives as a vector or tensor.

In this notation, f's Taylor expansion around $x = 0$ can be written as

$$f(0) + \langle \nabla f(0), x \rangle + \frac{1}{2!} \langle \nabla^2 f(0), x^{\otimes 2} \rangle + \cdots + \frac{1}{p!} \langle \nabla^p f(0), x^{\otimes p} \rangle + \cdots .$$

Of course, in general this Taylor expansion is not exact unless f is analytic in the appropriate neighborhood. For our purpose, the Taylor expansion with remainder is more useful, as it is exact:

Lemma H.8 (Taylor expansion with remainder)**.** *If $f : \mathbb{R}^k \to \mathbb{R}$, $f(x) = f(x_1, \ldots, x_k)$ is C^{p+1} (i.e., $\nabla^p f(x)$ has continuous partial derivatives), then*

$$f(x) = f(0) + \langle \nabla f(0), x \rangle + \cdots + \frac{1}{p!} \langle \nabla^p f(0), x^{\otimes p} \rangle + R_{p+1},$$

where the remainder R_{p+1} is

$$R_{p+1} \stackrel{\text{def}}{=} \frac{1}{p!} \int_0^1 (1-t)^p \langle \nabla^{p+1} f(tx), x^{\otimes (p+1)} \rangle dt$$

Vector Function For the vector case $f : \mathbb{R}^k \to \mathbb{R}^l, f(x) = (f_1(x), \ldots, f_l(x))$, all of the above considerations apply to the components of f in parallel. For example,

$$\partial^{\mathbf{P}} f(x) = \partial^{\mathcal{P}} f(x) \stackrel{\text{def}}{=} (\partial^{\mathcal{P}} f_1(x), \ldots, \partial^{\mathcal{P}} f_l(x)) \in \mathbb{R}^l$$
$$\nabla^p f(x) \stackrel{\text{def}}{=} (\nabla^p f_1(x), \ldots, \nabla^p f_l(x)) \in \mathbb{R}^{l \times k^p}$$

H.6 Higher Order Chain Rule

Lemma H.9 ([9]). *Suppose $f : \mathbb{R}^l \to \mathbb{R}, y : \mathbb{R}^k \to \mathbb{R}^l$ are C^k and consider their composition $f(y) = f(y(x_1, \dots, x_k))$. Then for any vector $\mathbf{P} \in [k]^r$,*

$$\partial_x^{\mathbf{P}} f(y) = \sum_{\tau} \langle \nabla_y^{|\tau|} f(y), \bigotimes_{\mathcal{P} \in \mathbf{P}[\tau]} \partial_x^{\mathcal{P}} y \rangle \tag{52}$$

where τ ranges over partitions of $\{1, \dots, r\}$ and $\mathbf{P}[\tau]$ is the partition $\{\{P_i : i \in \mathcal{S}\} : \mathcal{S} \in \tau\}$ of $\tilde{\mathbf{P}}$.

Eq. (52) may look somewhat scary at first, but when we apply it, the range of τ (in the outer sum) and the range of $\mathcal{P}$ (in the inner tensor product) are both $O(1)$-sized, as $n \to \infty$. So when we want to bound $\partial_x^{\mathbf{P}} f(y)$ (which is the only time we will use Eq. (52)), what we end up caring about is only the individual norms of $\nabla_y^{|\tau|} f(y)$ and $\partial_x^{\mathcal{P}} y$. May this fact assuage the reader intimidated by the density of Eq. (52).

Nevertheless, let us take some time to digest Eq. (52).

To unpack the tensor notation, notice that 1) $\nabla_y^{|\tau|} f(y) \in \mathbb{R}^{l^{|\tau|}}$, 2) $\partial_x^{\mathcal{P}} y = (\partial_x^{\mathcal{P}} y_1, \dots, \partial_x^{\mathcal{P}} y_l) \in \mathbb{R}^l$, so 3) $\bigotimes_{\mathcal{P} \in \mathbf{P}[\tau]} \partial_x^{\mathcal{P}} y$, being a tensor over $|\mathbf{P}[\tau]| = |\tau|$ vectors, is in $\mathbb{R}^{l^{|\tau|}}$ as well. So, for example, when $|\tau| = 1$, there is only one partition, consisting of the whole of $[k]$, and $\mathbf{P}[\tau] = \{\tilde{\mathbf{P}}\}$, so that the corresponding term in Eq. (52) is

$$\langle \nabla_y f(y), \partial_x^{\mathbf{P}} y \rangle = \sum_{j=1}^{l} \frac{\partial f}{\partial y_j} \frac{\partial^r y_j}{\partial x^{\mathbf{P}}}.$$

As a helping example, consider the case when $r = 2$ and $\mathbf{P} = (1, 2)$. Then we have

$$\partial_x^{\mathbf{P}} f(y) = \partial_{x_1} \partial_{x_2} f(y) = \partial_{x_1} \left(\langle \nabla_y f(y), \partial_{x_2} y \rangle \right) = \langle \nabla_y f(y), \partial_{x_1} \partial_{x_2} y \rangle + \langle \nabla_y^2 f(y), \partial_{x_1} y \otimes \partial_{x_2} y \rangle.$$

Here the first term corresponds to $\tau = \{[k]\}$; in this case, $\mathbf{P}[\tau] = \{\tilde{\mathbf{P}}\} = \{\{x_1, x_2\}\}$. The second term corresponds to $\tau = \{\{1\}, \{2\}\}$; in that case, $\mathbf{P}[\tau] = \{\{x_1\}, \{x_2\}\}$.

Note that Eq. (52) is not equivalent to the look-alike equation where we allow τ to range over partitions of the multiset $\tilde{\mathbf{P}}$ and letting $\mathcal{P}$ in the inner tensor product range over $\mathcal{P} \in \tau$. This alternative equation would lead to too few terms in the sum over τ, as is apparent when one considers $\mathbf{P} = (1, \dots, 1)$.

Applying Eq. (51) to Eq. (52), we get

Lemma H.10. *In the same setting as in Lemma H.9, we have*

$$|\partial_x^{\mathbf{P}} f(y)| \le \sum_{\tau} \|\nabla_y^{|\tau|} f(y)\| \cdot \prod_{\mathcal{P} \in \mathbf{P}[\tau]} \|\partial_x^{\mathcal{P}} y\|. \tag{53}$$

We will in particular need to apply this repeatedly to the case when $f(y) = \prod_j y_j$:

Lemma H.11. *Suppose $y : \mathbb{R}^k \to \mathbb{R}^l$ is C^k, with $y(x)$ denoting the value of y on input x. Then for any vector $\mathbf{P} \in [k]^r$,*

$$\left\| \partial_x^{\mathbf{P}} \prod_j y_j \right\| \le \sum_{t=0}^{l-1} \sqrt{l^{l-t-1} \|y\|_{2t}^{2t}} \cdot \sum_{\tau} \prod_{\mathcal{P} \in \mathbf{P}[\tau]} \|\partial_x^{\mathcal{P}} y\|. \tag{54}$$

where τ ranges over partitions of $\{1, \dots, r\}$ of size $l - t$ and $\mathbf{P}[\tau]$ is as in Lemma H.9. Here $\|y\|_0^0$ is by convention defined to be l.

The proof below actually shows the l^{l-t-1} in Eq. (54) can be refined to the falling factorial $(l - 1) \cdots (t + 1)$ but we will not need this. Also note that there are no partitions of size $> r$, so that the summand vanishes when $t < l - r$.

Proof. For $f(y) = \prod_j y_j$, by Eq. (53), we just need to show

$$\|\nabla_y^{|\tau|} f(y)\|^2 \le l^{|\tau|-1} \|y\|_{2(l-|\tau|)}^{2(l-|\tau|)}.$$

Each nonzero entry of $\nabla_y^{|\tau|} f(y)$ has the form y^S for some size-$(l - |\tau|)$ subset (i.e., having no duplicates) S of $[l]$. By power-mean inequality, $|y^S|^2 \le \frac{1}{|S|} \sum_{j \in S} |y_j|^{2|S|}$. Taking sum over all possible S of size $(l - |\tau|)$, we have

$$\sum_S |y^S|^2 \le \binom{l}{|\tau|} \frac{1}{l} \sum_{j=1}^{l} |y_j|^{2(l-|\tau|)}$$

Finally, each y^S appears $|\tau|!$ times in $\nabla_y^{|\tau|} f(y)$, so

$$\|\nabla_y^{|\tau|} f(y)\|^2 = |\tau|! \sum_S |y^S|^2 \le l^{|\tau|-1} \sum_{j=1}^{l} |y_j|^{2(l-|\tau|)}$$

as desired. □

H.7 Smoothness Profile of A Function

Definition H.12 (Smoothness Profile)**.** Consider a smooth (C^∞) function $f : \mathbb{R}^k \to \mathbb{R}$. If there is a sequence $\mathcal{C}$ of positive reals $(C_1, p_1), (C_2, p_2), \ldots$ such that for any $\mathbf{P} \in [k]^r$, we have $\left|\partial^{\mathbf{P}} f(x)\right| \le C_r(1 + |x_1|^{p_r} + \cdots + |x_k|^{p_r})$, then we say $\mathcal{C}$ is a *smoothness profile*, or just *profile* for short, of f.

Obviously, f is polynomially smooth (Definition 3.5) iff it has a profile.

As one could guess, the "expected smoothness" of a program's vectors (under perturbation of its matrices) will depend on the nonlinearities ϕ^i of the program *only* through the profiles of ϕ^i. Thus, there are smoothness guarantees that are uniform over all programs whose nonlinearities have the same profiles.

I The Basic Moment Argument

Throughout this work, we need to study many different sums of random variables $X = \sum_{\alpha=1}^{n} x_\alpha$. In particular, we will repeatedly use a basic argument to bound its moment $\mathbb{E}\, X^p$ for even p, which is formally codified in Lemma I.4 at the end of this section. But to motivate this eventual formulation, it's nice to start with a simple example.

Simple Motivating Example The simplest example would be when x_α are iid random variables that don't depend on n (but eventually we will need to cover the case of general non-iid distributions depending on n, with small correlations among x_α). Then for even p, by Eq. (49),

$$\mathbb{E}\, X^p = \sum_{\mathcal{P}} N_{\mathcal{P}} \mathbb{E}\, x^{\mathcal{P}} \tag{55}$$

where $\mathcal{P}$ ranges over all multisets of size $|\mathcal{P}| = p$ taking value in $[n]$ and $N_{\mathcal{P}}$ is the multinomial coefficient defined in Definition H.7. In our case, with each x_α being iid, it's clear that $\mathbb{E}\, x^{\mathcal{P}} = \mathbb{E}\, x^{\mathcal{P}'}$ if the multisets $\mathcal{P}, \mathcal{P}'$ have the samed induced partition: $\pi(\mathcal{P}) = \pi(\mathcal{P}')$. For example, $\mathbb{E}\, x_1 x_1 x_2 x_3 = \mathbb{E}\, x_1 x_1 x_3 x_4 = \mathbb{E}\, x_4 x_4 x_n x_{n-1}$ because the corresponding multisets all have the same induced partition $\{2, 1, 1\}$ of 4.

For any fixed partition τ of p, we can ask how many multisets $\mathcal{P}$ are there with $\pi(\mathcal{P}) = \pi$. The answer is quite straightforward but requires a bit of explanation notationally: we think of τ as another multiset and *take its induced partition* $\pi(\tau)$ of the integer $|\tau|$; then the answer is the multinomial coefficient $\binom{n}{\pi(\tau)}$. For the above example of $\tau = \{2, 1, 1\}$, we have $\pi(\tau) = \{1, 2\}$, because a) 1 in $\pi(\tau)$ is the multiplicity of 2 in τ and b) 2 in $\pi(\tau)$ is the multiplicity of 1 in τ. So there are $\binom{n}{1;2}$ multisets $\mathcal{P}$ with $\pi(\mathcal{P}) = \pi$.

Therefore, we can further rewrite Eq. (55) as a sum over partitions τ of integer p:

$$\mathbb{E}\, X^p = \sum_{\tau} \binom{n}{\pi(\tau)} N_{\mathcal{P}} \mathbb{E}\, x^{\mathcal{P}} \tag{56}$$

where for every τ, $\mathcal{P}$ is any choice of multiset with $\pi(\mathcal{P}) = \tau$. The advantage of this representation is that $\mathbb{E}\, X^p$ is now clearly a polynomial in n. This is because 1) τ ranges over a set that does not depend on n; 2) $N_{\mathcal{P}}$ does not depend on n; and 3) the distribution of x_α does not depend on n by our assumption. Thus, *as $n \to \infty$, we can obtain a bound on X^p by deriving the leading term of this polynomial.* Because $\binom{n}{\pi(\tau)} = \Theta(n^{|\tau|})$ (e.g., $\binom{n}{\pi(\{1,1,2\})} = \binom{n}{2;1} = \Theta(n^3)$), this means focusing on the the partitions τ with the largest $|\tau|$. In general, of course, this is just $\tau = \{1, \ldots, 1\}$ where $|\tau| = p$, corresponding to monomials like $x_1 x_2 \cdots x_p$. But in the cases we are concerned with, $\mathbb{E}\, x^{\mathcal{P}}$ for this τ will vanish, so we need to look at lower order terms.

For example, assume further that x_α has mean 0 and variance 1. Then $\mathbb{E}\, x^{\mathcal{P}} = 0$ for any $\mathcal{P}$ that contains an element appearing with multiplicity 1 (e.g., $\mathbb{E}\, x_1 x_2 x_2 = 0$ because it's equal to $(\mathbb{E}\, x_1)(\mathbb{E}\, x_2^2) = 0 \cdot (\mathbb{E}\, x_2^2) = 0$). Therefore, the leading term in the polynomial Eq. (55) corresponds exactly to the partition $\tau = \{2, \ldots, 2\}$ (where $|\tau| = p/2$) of p (corresponding to monomials like $\mathbb{E}\, x_1^2 \cdots x_{p/2}^2$ which is equal to 1 by our variance-1 assumption), and any other τ must either have smaller $|\tau|$ or it has $\mathbb{E}\, x^{\mathcal{P}} = 0$. Therefore, in this case, we deduce

$$\mathbb{E}\, X^p = \Theta(n^{p/2}), \quad \text{as } n \to \infty. \tag{57}$$

We can of course say more precise things about $\mathbb{E}\, X^p$ under these assumptions, which we record below.

Proposition I.1. *If $x_\alpha, \alpha \in [n]$, are sampled iid from a distribution with kth moment ν_k, then for any even integer p,*

$$\mathbb{E}\left(\sum_{\alpha=1}^{n} x_\alpha\right)^p = \sum_{\tau} \binom{n}{\pi(\tau)} \binom{p}{\tau} \nu^\tau \tag{58}$$

summing over partitions τ of integer p and any n.

Here $\nu^\tau = \prod_{k \in \tau} \nu_k$ following the multiset notation in Appendix H.3.

Slightly more generally, we have

Proposition I.2. *If $x_\alpha, \alpha \in [n]$, are sampled independently and the distribution of x_α has kth moment bounded above by ν_k, then for any even integer p,*

$$\mathbb{E}\left(\sum_{\alpha=1}^{n} x_\alpha\right)^p \leq \sum_{\tau} \binom{n}{\pi(\tau)} \binom{p}{\tau} \nu^\tau \tag{59}$$

summing over partitions τ of integer p and any n.

General Case Now, when we consider x_α that are not iid and may correlate amongst themselves, Eq. (55) will continue to hold but in general Eq. (56) will not. Furthermore, $\mathbb{E}\, x^{\mathcal{P}}$ in the general case can also depend on n. So even if there's enough symmetry to obtain Eq. (56), we no longer have a polynomial expression of $\mathbb{E}\, X^p$ in n.

Nevertheless, most of the time, our objective is to show that $\mathbb{E}\, X^p$ is $O(1)$ as $n \to \infty$, and the above arguments can be easily adapted to work given the following condition:

Definition I.3 (Small Moment Condition)**.** Consider a sequence x of random vectors $(x(n) \in \mathbb{R}^n)_{n=1}^{\infty}$. We say x satisfies the *Small Moment Condition (SMC)* if for every even integer p, there exists a constant C such that

$$\mathbb{E}\, x(n)^{\mathcal{P}} \leq C n^{-|\mathrm{uniq}(\mathcal{P})|} \tag{60}$$

for every multiset $\mathcal{P}$ of size p and for any n, where $\mathrm{uniq}(\mathcal{P})$ is the set of unique elements of $\mathcal{P}$.

Lemma I.4. *Consider a sequence x of random vectors $(x(n) \in \mathbb{R}^n)_{n=1}^{\infty}$. If x satisfies the Small Moment Condition (Definition I.3), then for any real $p \geq 1$,*

$$\mathbb{E}\left|\sum_{\alpha=1}^{n} x(n)_\alpha\right|^p = O(1) \quad \textit{as } n \to \infty. \tag{61}$$

Here the constant in $O(1)$ depends only on p and the constant C in Definition I.3.

Proof. Note that, by the power-mean inequality, it suffices to prove this for even integer p. Let $X(n) \stackrel{\text{def}}{=} \sum_{\alpha=1}^{n} x(n)_\alpha$. Below we suppress the argument (n) notationally. Note that $|\mathrm{uniq}(\mathcal{P})| = |\pi(\mathcal{P})|$, for $\pi(\mathcal{P})$ defined in Definition H.5. By Eq. (55), we have, for every n,

$$\mathbb{E}\, X^p \le \sum_{\mathcal{P}} N_{\mathcal{P}} C n^{-|\pi(\mathcal{P})|} \le C' \sum_{\mathcal{P}} n^{-|\pi(\mathcal{P})|}$$

summing over multisets $\mathcal{P}$ of size p taking value in $[n]$. where C is the constant in Definition I.3 and $C' = C \max_{\mathcal{P}} N_{\mathcal{P}}$. Then the same symmetry argument leading to Eq. (56) yields

$$\mathbb{E}\, X^p \le C' \sum_{\tau} \binom{n}{\pi(\tau)} n^{-|\tau|} = \sum_{\tau} O(1) = O(1),$$

where τ ranges over all partitions of integer p. □

As an example, if $x_\alpha = x(n)_\alpha, \alpha \in [n]$, are sampled independently and the distribution of x_α has kth moment bounded above by $\nu_k / n^{k/2}$, where ν_k are independent of n, then $\mathbb{E}\, x^{\mathcal{P}} \le \nu^{\pi(\mathcal{P})} n^{-p/2}$ (where $\nu^{\pi(\mathcal{P})} = \prod_{k \in \pi(\mathcal{P})} \nu_k$). If each x_α further has mean 0, then as discussed above Eq. (57), $\mathbb{E}\, x^{\mathcal{P}} = 0$ if $|\pi(\mathcal{P})| > p/2$. For $\mathcal{P}$ with $|\pi(\mathcal{P})| \le p/2$, we have $\mathbb{E}\, x^{\mathcal{P}} \le \nu^{\pi(\mathcal{P})} n^{-p/2} \le C n^{-|\pi(\mathcal{P})|/2}$ where $C = \max_\tau \nu^\tau$ taken over all partitions τ of integer p. Thus SMC (Definition I.3) is satisfied, and Lemma I.4 applies to yield

Proposition I.5. *Consider a sequence x of random vectors $(x(n) \in \mathbb{R}^n)_{n=1}^{\infty}$. Suppose $x_\alpha = x(n)_\alpha, \alpha \in [n]$, are sampled independently. Assume there are $\nu_k, k = 1, 2, \ldots$, independent of n such that $\mathbb{E}\, |x_\alpha|^k \le \nu_k n^{-k/2}$ for all n and k. Then x satisfies Small Moment Condition (Definition I.3), and for any $p \ge 1$,*

$$\mathbb{E} \left| \sum_{\alpha=1}^{n} x(n)_\alpha \right|^p = O(1) \quad \text{as } n \to \infty. \tag{62}$$

Here the constant in $O(1)$ depends on p and $\nu_1, \ldots, \nu_p$ (i.e., the first p moment bounds only).

J A Priori Moment Controls

Our main result in this section is Lemma J.3, which bounds the moments of derivatives of vector entries in the program. We shall come to it after stating some definitions.

Oblivious Constants In this and following sections, we need to reason carefully about constants hidden in big-O expressions. In particular, we isolate the following notion of *oblivious constant*, which roughly means that the constant does not depend on the fine details of a program beyond some finite number of smoothness and moment bounds.

For the first time reader, exactly understanding this notion is not a priority. So it may help to skip ahead to Lemma J.3, keeping in mind just this intuitive understanding of *oblivious constants*, and come back only after absorbing key ideas of our proofs.

Definition J.1. Consider a program T in Setup J.2, nonlinearities $\psi^1, \ldots, \psi^l$ for some $l \ge 0$, and multisets $\mathcal{P}_1, \ldots, \mathcal{P}_r$ for some $r \ge 0$. In the context of a bound or a big-O expression, we say a constant C is *$(\mathcal{P}_1, \ldots, \mathcal{P}_r)$-oblivious wrt T and $\psi^1, \ldots, \psi^l$* if all of the following hold.

1. C does not depend on n
2. C depends on each $\mathcal{P}_i$ only through its size $|\mathcal{P}_i|$
3. For each combination of $|\mathcal{P}_1|, \ldots, |\mathcal{P}_r|$, there is an integer $K > 1$ such that
   (a) For any sequence $\nu_2, \nu_3, \ldots \ge 0$ such that $\mathbb{E}\, |a|^k \le \nu_k n^{-k/2}$ for all matrix entries a of T, our constant C can be taken to depend on the distributions of matrix entries in T only through $\nu_2, \ldots, \nu_K$
   (b) For any profile $((C_0, p_0), (C_1, p_1), \ldots)$ (Definition H.12) satisfied by all nonlinearities ϕ^i of the program T as well as $\psi^1, \ldots, \psi^l$, our constant C can be taken to depend on $\{\phi^i\}_i$ and $\{\psi^j\}_j$ only through $(C_0, p_0), \ldots, (C_K, p_K)$

(c) For any sequence $R_1, R_2, \ldots$ such that $\mathbb{E}\,|c^i|^q \leq R_q$ for all initial scalars c^i (i.e., where $i \in [M_0]$), all integers $q \geq 1$, and all n, our constant C can be taken to depend on $\{c^i\}_{i=1}^{M_0}$ only through $R_1, \ldots, R_K$.

(d) Furthermore, C is a oblivious function of $\nu_2, \ldots, \nu_K$, $(C_0, p_0), \ldots, (C_K, p_K)$, and $R_1, \ldots, R_K$.

We will just say $(\mathcal{P}_1, \ldots, \mathcal{P}_r)$*-oblivious* if the program and $\psi^1, \ldots, \psi^l$ are clear from context. Finally, any of the sets $\mathcal{P}_i$ can be an integer p, which just stands for the multiset $\{\underbrace{1, \ldots, 1}_{p}\}$.[24]

Thus, if an expression $\Lambda = \Lambda(\mathcal{P}, T, \psi)$ is bounded by a $\mathcal{P}$-oblivious constant wrt a program T and additional nonlinearity ψ, then

$$\sup_{\mathcal{P}, T, \psi} \Lambda \leq F(p, \mathcal{C}, (\nu_j)_j, (R_j)_j)$$

for some function F, where 1) $\mathcal{P}$ ranges over all multisets of size p, 2) ψ and all nonlinearities of T range over functions satisfying the profile $\mathcal{C}$, 3) the distributions of all matrix entries of T range of those that satisfy the moment bounds given by $(\nu_j)_j$, and 4) all initial scalars of T range over those that satisfy the moment bounds given by $(R_j)_j$.

Smoothness and Moment Control We relax Setup 3.6 slightly for our main result in this section.

Setup J.2. *Consider Setup 3.6, but relax 3*) to allow the variances of matrix entries to differ from n^{-1} but still bounded above by $\nu_2 n^{-1}$ for some $\nu_2 > 0$ common to all matrix entries.*

Lemma J.3 (Expected Smoothness of Vectors in a Program)**.** *Consider a Tensor Program under Setup J.2. Then, for any polynomially smooth $\psi : \mathbb{R}^M \to \mathbb{R}$, any $p \geq 1$, and any multiset $\mathcal{P}$ taking values in the program's matrix entries $\{A^i_{\alpha\beta}\}_{\alpha,\beta,i}$,*

$$\sup_{\alpha \in [n]} \mathbb{E}\left|\partial^{\mathcal{P}} \psi(g^1_\alpha, \ldots, g^M_\alpha; c^1, \ldots, c^M)\right|^p = O(1) \quad \text{as } n \to \infty. \tag{63}$$

where constant in the big-O is $(p, \mathcal{P})$-oblivious wrt ψ and the program.

We are slightly deviating from the partial derivative notation in Appendix H.5 as $\mathcal{P}$ now directly specify the variables to take derivatives against, rather than their indices.

One can interpret this result as saying that: any higher order derivative (with respect to weights) of any entry of $\psi(\cdots)$ will typically not explode to ∞ with n. The fact that the hidden constant is $(p, \mathcal{P})$-oblivious will be important when we prove our main result Theorem 5.2.

Remark J.4. As a sanity check, we discuss some features of this result before moving on to the proof. First, notice that the $\sup$ is outside the expectation. Were it the other way around, then we typically would expect some (function of) $\log n$ factors on the RHS of the bound.

Second, notice that if each matrix A^i has iid entries (instead of the more general case covered here where entries can come from different distributions), then by symmetry, the above expectation for all α would be identical, so the supremum is extraneous. But in general, this supremum is not extraneous.

Third, we assume the less stringent Setup J.2 instead of Setup 3.6 not just because we can but also because we need this in our inductive proof: we will need to reason about programs where some matrix entries are shrunken to 0, a condition that Setup J.2 captures but Setup 3.6 does not.

Remark J.5. This will hold for programs with variable dimensions, with the addendum that the constants $B_{p,|\mathcal{P}|}$ also can depend on hidden width ratios.

By applying power mean inequality to Lemma J.3, we also easily get

Lemma J.6. *Consider the same setting as Lemma J.3. Then*

$$\mathbb{E}\left|\partial^{\mathcal{P}} \frac{1}{n} \sum_{\alpha \in [n]} \psi(g^1_\alpha, \ldots, g^M_\alpha; c^1, \ldots, c^M)\right|^p = O(1) \quad \text{as } n \to \infty. \tag{64}$$

for the same hidden constant as in Eq. (63)*.*

However, later we will see that this bound is unnecessarily loose when $\mathcal{P}$ is not empty (Lemma K.6).

[24] In this case of $\mathcal{P}_i$ being an integer, condition 2 is then trivially satisfied, so the important condition is condition 3.

J.1 Proof of Lemma J.3: Induction Setup

In everything below, by *constant* we always mean something independent of n that may or may not depend on other data.

Fix the number of initial vectors M_0, as well as the sequence of scaled moment bounds $\nu_2, \nu_3, \ldots$, the profile $\mathcal{C}$, and the initial scalar moment bounds $R_1, R_2, \ldots$ as discussed in Setup J.2. We will prove the following claims simultaneously *for all programs that have M_0 initial vectors/scalars and satisfy the above constraints (specified by $(\nu_k)_k, \mathcal{C}, (R_k)_k$).* We do so by induction on the vector index j:

Claim 1(j) For any integer $p \geq 0$, there is a sequence of constants $B_{\mathrm{j},p,0}, B_{\mathrm{j},p,1}, \ldots$ such that

$$\sup_{\alpha \in [n]} \mathbb{E} \left|\partial^{\mathcal{P}} g_\alpha^{\mathrm{j}}\right|^p \leq B_{\mathrm{j},p,|\mathcal{P}|}$$

for any multiset $\mathcal{P}$ of the program's matrix entries. Furthermore, $B_{\mathrm{j},p,|\mathcal{P}|}$ is $(\mathrm{j}, p, \mathcal{P})$-oblivious.

Claim 2(j) For any polynomially smooth $\psi : \mathbb{R}^{2\mathrm{j}} \to \mathbb{R}$, any integer $p \geq 0$, there is a sequence of constants $B^{\psi}_{\mathrm{j},p,0}, B^{\psi}_{\mathrm{j},p,1}, \ldots$ such that

$$\sup_{\alpha \in [n]} \mathbb{E} \left|\partial^{\mathcal{P}} \psi(g_\alpha^1, \ldots, g_\alpha^{\mathrm{j}}; c^1, \ldots, c^{\mathrm{j}})\right|^p \leq B^{\psi}_{\mathrm{j},p,|\mathcal{P}|}$$

for any multiset $\mathcal{P}$ of the program's matrix entries. Furthermore, $B^{\psi}_{\mathrm{j},p,|\mathcal{P}|}$ is $(\mathrm{j}, p, \mathcal{P})$-oblivious wrt to ψ and the program.

Note that $|\cdot|^0$ always equal 1 by convention in both claims. Of course, Claim $2(M)$ would yield Lemma J.3.

Obviously, Claim 1(j) is a special case of Claim 2(j), but our induction proof will go like this

$$\text{Claim } 2(\mathrm{j}-1) \implies \text{Claim } 1(\mathrm{j})$$
$$\text{Claim } 1(1, \ldots, \mathrm{j}) \text{ and Claim } 2(1, \ldots, \mathrm{j}-1) \implies \text{Claim } 2(\mathrm{j})$$

Before we begin the induction proof, we first record several consequences of the claims above.

Proposition J.7. *Recall ϕ^i denotes the ith nonlinearity of the program. Claim 2$(i-1)$ implies that*

$$\mathbb{E} \left|\partial^{\mathcal{P}} c^i\right|^p \leq B^{\phi^i}_{i-1,p,|\mathcal{P}|}$$

for any multiset $\mathcal{P}$ of the program's matrix entries.

Proof. Unwinding the definition of c^i (Eq. (4)), we have

$$\begin{aligned}
\mathbb{E} \left|\partial^{\mathcal{P}} c^i\right|^p &= \mathbb{E} \left|\frac{1}{n} \sum_{\alpha=1}^{n} \partial^{\mathcal{P}} \phi^i(g_\alpha^1, \ldots, g_\alpha^{i-1}; c^1, \ldots, c^{i-1})\right|^p \\
&\leq \frac{1}{n} \sum_{\alpha=1}^{n} \mathbb{E} \left|\partial^{\mathcal{P}} \phi^i(g_\alpha^1, \ldots, g_\alpha^{i-1}; c^1, \ldots, c^{i-1})\right|^p && \text{applying Lemma H.2} \\
&\leq \sup_{\alpha \in [n]} \mathbb{E} \left|\partial^{\mathcal{P}} \phi^i(g_\alpha^1, \ldots, g_\alpha^{i-1}; c^1, \ldots, c^{i-1})\right|^p \leq B^{\phi^i}_{i,p,|\mathcal{P}|} && \text{applying Claim } 2(i-1).
\end{aligned}$$

□

Proposition J.8. *Consider any polynomially smooth $\psi : \mathbb{R}^{2\mathrm{j}} \to \mathbb{R}$ and any integers $p \geq 1$, $k \geq 0$. Recall that $\nabla^k \psi : \mathbb{R}^{2\mathrm{j}} \to \mathbb{R}^{(2\mathrm{j})^k}$ is the function that computes the tensor of ψ's kth-order partial derivatives. Then Claim 1$(1, \ldots, \mathrm{j})$ and Claim 2$(1, \ldots, \mathrm{j}-1)$ together imply the following L^p norm bound: There is a constant C, (j, p, k)-oblivious wrt ψ and the program, such that*

$$\sup_{\alpha \in [n]} \mathbb{E} \left\|\nabla^k \psi(g_\alpha^1, \ldots, g_\alpha^{\mathrm{j}}; c^1, \ldots, c^{\mathrm{j}})\right\|_p^p \leq C$$

Note the form of this bound is very intuitive, since $\nabla^k \psi$ is just a polynomially bounded function, but taking values in a multi- but constant-dimensional space $\mathbb{R}^{(2\mathrm{j})^k}$ instead of $\mathbb{R}$. The proof is just routine manipulation using Lemma H.2 and applications of Claim 1 and Claim 2.

Proof. Let (C, q) be the the kth element of ψ's profile, i.e., such that for all input vectors $v \in \mathbb{R}^{2\mathrm{j}}$ and for all $\mathbf{U} \in [2\mathrm{j}]^k$,

$$|\partial_v^{\mathbf{U}} \psi(v)| \leq C(1 + \|v\|_q^q). \tag{65}$$

Let $u \stackrel{\text{def}}{=} (g_\alpha^1, \ldots, g_\alpha^{\mathrm{j}}; c^1, \ldots, c^{\mathrm{j}}) \in \mathbb{R}^{2\mathrm{j}}$. Then

$$\|\nabla_u^k \psi(u)\|_p^p \leq (2\mathrm{j})^k \sup_{\mathbf{U}} |\partial_u^{\mathbf{U}} \psi(u)|^p$$

where $\mathbf{U}$ ranges over all vectors in $[2\mathrm{j}]^k$. Thus it suffices to bound $\sup_{\alpha \in [n]} \mathbb{E} \sup_{\mathbf{U}} |\partial_u^{\mathbf{U}} \psi(u)|^p$ by a constant that depends on ψ only through C and q.

Applying Lemma H.1,

$$\mathbb{E} |\partial_u^{\mathbf{U}} \psi(u)|^p \leq \mathbb{E} \left[C(1 + \|u\|_q^q)\right]^p \leq C' \mathbb{E} \left(1 + \|u\|_{pq}^{pq}\right)$$

where C' is a constant depending only on C, q, p, j continuously. Therefore it remains to show that $\mathbb{E} \|u\|_{pq}^{pq}$ is bounded by a constant independent of α. But, unwinding the definition of u,

$$\mathbb{E} \|u\|_{pq}^{pq} = \mathbb{E} |g_\alpha^1|^{pq} + \cdots + |g_\alpha^{\mathrm{j}}|^{pq} + |c^1|^{pq} + \cdots + |c^{\mathrm{j}}|^{pq}$$

By Claim $1(1, \ldots, \mathrm{j})$,

$$\sup_{\alpha \in [n]} \mathbb{E} |g_\alpha^i|^{pq} \leq B_{i,pq,0}$$

For $i \leq \mathrm{j}$, by Claim $2(i-1)$ and Proposition J.7, we have

$$\mathbb{E} |c^i|^{pq} \leq B_{i-1,pq,0}^{\phi^i}$$

So $\mathbb{E} \|u\|_{pq}^{pq}$ is indeed bounded by a constant independent of α and (j, p, k)-oblivious. □

Proposition J.9. *Consider any integer $p \geq 0$, polynomially smooth $\psi : \mathbb{R}^{2\mathrm{j}} \to \mathbb{R}$ and any multisets $\mathcal{P}, \mathcal{U}$ of matrix entries. Let $z_\alpha \stackrel{\text{def}}{=} \partial^{\mathcal{P}} \psi(g_\alpha^1, \ldots, g_\alpha^{\mathrm{j}}; c^1, \ldots, c^{\mathrm{j}})$. Assume Claim 2(j). Then for any multiset $\mathcal{N}$ taking values in $[n]$,*

$$\mathbb{E} |\partial^{\mathcal{U}} z^{\mathcal{N}}|^p \leq C$$

for some constant C that is $(\mathrm{j}, p, \mathcal{N}, \mathcal{P}, \mathcal{U})$-oblivious wrt ψ and the program.

The statement of this bound is a bit more complicated than the previous ones, but again the content is intuitive. It says that some interleaved composition of 1) taking a constant number of partial derivatives and 2) taking a product over a constant number of "neuron index" $\alpha \in [n]$ will still result in an $O(1)$ quantity. The proof is again routine manipulation using Lemma H.2 and standard inequalities after applying the higher order product rule bound in Lemma H.11.

Proof. Let $l \stackrel{\text{def}}{=} |\mathcal{N}|$ and $v \in \mathbb{R}^l$ be the vector $(z_\alpha)_{\alpha \in \mathcal{N}}$ for an arbtirary ordering of $\mathcal{N}$ (including multiplicity). Thus $z^{\mathcal{N}} = \prod_j v_j$. Fix a ordering $\mathbf{U}$ of $\mathcal{U}$. By Lemma H.11 and Lemma H.2, there is a constant G depending only on $|\mathcal{U}|$ and p such that

$$|\partial^{\mathcal{U}} z^{\mathcal{N}}|^p \leq G \sum_{t=0}^{l-1} \sqrt{l^{l-t-1} \|v\|_{2t}^{2tp}} \sum_\tau \sqrt{\prod_{\mathcal{Q}} \|\partial^{\mathcal{Q}} v\|^{2p}}$$

where τ ranges over partitions of $\{1, \ldots, |\mathcal{U}|\}$ of size $l - t$ and $\mathcal{Q}$ ranges over $\mathbf{U}[\tau]$ as in Lemma H.9. Here $\|v\|_0^0$ is by convention defined to be l. By Jensen's (concave) inequality, we then have

$$\mathbb{E} |\partial^{\mathcal{U}} z^{\mathcal{N}}|^p \leq G \sum_{t=0}^{l-1} \sqrt{l^{l-t-1} \mathbb{E} \|v\|_{2t}^{2tp}} \sum_\tau \sqrt{\mathbb{E} \prod_{\mathcal{Q}} \|\partial^{\mathcal{Q}} v\|^{2p}}$$

Since the sizes of the ranges of t and of τ both depend only on $l = |\mathcal{N}|$ and $|\mathcal{U}|$, it suffices to show that both $\mathbb{E} \|v\|_{2t}^{2tp}$ and $\mathbb{E} \prod_{\mathcal{Q}} \|\partial^{\mathcal{Q}} v\|^{2p}$ have bounds that are $(\mathrm{j}, p, \mathcal{N}, \mathcal{P}, \mathcal{U})$-oblivious wrt ψ and the program.

Bounding $\mathbb{E}\,\|v\|_{2t}^{2t}$ Again, by Lemma H.2, $\mathbb{E}\,\|v\|_{2t}^{2tp} = O(\mathbb{E}\,\|v\|_{2tp}^{2tp})$, so it suffices to bound the latter. Now for all $t \in \{0, \dots, l-1\}$,

$$\mathbb{E}\,\|v\|_{2tp}^{2tp} = \sum_{\alpha \in \mathcal{N}} \mathbb{E}\,|z_\alpha|^{2tp} \le l \sup_{\alpha \in [n]} \mathbb{E}\,|z_\alpha|^{2tp} \le l B^{\psi}_{\mathrm{j},2tp,|\mathcal{P}|}$$

by Claim 2(j). This obviously satisfies the desired property.

Bounding $\mathbb{E}\prod_{\mathcal{Q}} \|\partial^{\mathcal{Q}} v\|^2$ Let $s \stackrel{\text{def}}{=} l - t$, so that the product $\prod_{\mathcal{Q}}$ iterates over s elements. Then by Hölder's Inequality,

$$\mathbb{E}\prod_{\mathcal{Q}} \|\partial^{\mathcal{Q}} v\|^{2p} \le \sqrt[s]{\prod_{\mathcal{Q}} \mathbb{E}\,\|\partial^{\mathcal{Q}} v\|^{2sp}}$$

By Lemma H.1, for a constant R depending on only l and s, we have

$$\begin{aligned}
\mathbb{E}\,\|\partial^{\mathcal{Q}} v\|^{2sp} &\le R\mathbb{E}\,\|\partial^{\mathcal{Q}} v\|_{2sp}^{2sp} \le lR \sup_{\alpha \in [n]} \mathbb{E}\,|\partial^{\mathcal{Q}} z_\alpha|^{2sp} \\
&= lR \sup_{\alpha \in [n]} \mathbb{E}\,|\partial^{\mathcal{Q}} \partial^{\mathcal{P}} \psi(g_\alpha^1, \dots, g_\alpha^{\mathrm{j}}; c^1, \dots, c^{\mathrm{j}})|^{2sp} \\
&\le lRB^{\psi}_{\mathrm{j},2sp,|\mathcal{Q}|+|\mathcal{P}|}
\end{aligned}$$

by Claim 2(j). From this, it's clear that $\mathbb{E}\prod_{\mathcal{Q}} \|\partial^{\mathcal{Q}} v\|^2$ has the desired property as well. □

J.2 Base Case: Claim 1$(1, \dots, M_0)$ and Claim 2$(1, \dots, M_0)$

Here we consider the case of $\mathrm{j} = 1, \dots, M_0$.

When $|\mathcal{P}| > 0$, Claim 1(j) and Claim 2(j) are trivially true since there's no dependence on the matrices A^i yet.

Now assume $|\mathcal{P}| = 0$. Then Claim 1(j) follows from standard Gaussian moment expressions. For Claim 2(j), we note $x \stackrel{\text{def}}{=} \psi(g^1, \dots, g^{\mathrm{j}}; c^1, \dots, c^{\mathrm{j}})$ has

$$\begin{aligned}
|x_\alpha| &\le C(1 + |g_\alpha^1|^q + \dots + |g_\alpha^{\mathrm{j}}|^q + |c^1|^q + \dots + |c^{\mathrm{j}}|^q) \\
&\le C(1 + |g_\alpha^1|^q + \dots + |g_\alpha^{\mathrm{j}}|^q + \mathrm{j}R_q)
\end{aligned}$$

where C, q come from a profile of ψ and R_q is the qth moment bound on all the initial scalars. Then again we can apply standard Gaussian moment expressions to derive Claim 2(j).

J.3 Claim 1$(1, \dots, \mathrm{j})$ and Claim 2$(1, \dots, \mathrm{j}-1)$ Imply Claim 2(j)

Here we assume Claim 1$(1, \dots, \mathrm{j})$ and Claim 2$(1, \dots, \mathrm{j}-1)$ for $\mathrm{j} \ge M_0 + 1$ and derive Claim 2(j).

Let $u \stackrel{\text{def}}{=} (g_\alpha^1, \dots, g_\alpha^{\mathrm{j}}; c^1, \dots, c^{\mathrm{j}}) \in \mathbb{R}^{2\mathrm{j}}$ and let $\mathbf{P}$ be any ordering of $\mathcal{P}$. By Lemma H.10, we have

$$\partial^{\mathbf{P}} \psi(u) = \sum_{\tau} \|D^{|\tau|}\| \cdot \prod_{\mathcal{Q}} \|\partial^{\mathcal{Q}} u\|$$

where $D^k \stackrel{\text{def}}{=} \nabla_u^k \psi(u) \in \mathbb{R}^{(2\mathrm{j})^k}$, τ ranges over partitions of $\{1, \dots, |\mathcal{P}|\}$, and, for each τ, $\mathcal{Q}$ ranges over the elements (which are multisets) of the partition $\mathbf{P}[\tau] = \{\{P_i : i \in \mathcal{S}\} : \mathcal{S} \in \tau\}$ of $\tilde{\mathbf{P}}$.

Thus, applying Lemma H.2,

$$\begin{aligned}
|\partial^{\mathbf{P}} \psi(u)|^p &\le \left(\sum_{\tau} \|D^{|\tau|}\| \cdot \prod_{\mathcal{Q}} \|\partial^{\mathcal{Q}} u\| \right)^p \\
&\le G \sum_{\tau} \|D^{|\tau|}\|^p \cdot \prod_{\mathcal{Q}} \|\partial^{\mathcal{Q}} u\|^p
\end{aligned}$$

where G is the constant from Lemma H.2 that depends only on p and $|\mathcal{P}|$ (through the number of partitions of $\{1, \ldots, |\mathcal{P}|\}$). Taking expectation and applying another Cauchy-Schwarz gives

$$\mathbb{E}\,|\partial^{\mathbf{P}}\psi(u)|^p \leq G \sum_{\tau} \sqrt{\mathbb{E}\,\|D^{|\tau|}\|^{2p}} \cdot \sqrt{\mathbb{E} \prod_{\mathcal{Q}} \|\partial^{\mathcal{Q}} u\|^{2p}}$$

Since the number of partitions of $\{1, \ldots, |\mathcal{P}|\}$ (which is the range of τ) depends only on $|\mathcal{P}|$, it suffices to prove that both $\mathbb{E}\,\|D^{|\tau|}\|^{2p}$ and $\mathbb{E} \prod_{\mathcal{Q}} \|\partial^{\mathcal{Q}} u\|^{2p}$ are bounded by constants that are $(\mathrm{j}, p, \mathcal{P})$-oblivious wrt ψ and the program and are independent of α.

Bounding $\mathbb{E}\,\|D^{|\tau|}\|^{2p}$ This follows directly from Proposition J.8 (which is a straightforward bound by replacing each partial derivative of ψ with its polynomial upper bound).

Bounding $\mathbb{E} \prod_{\mathcal{Q}} \|\partial^{\mathcal{Q}} u\|^{2p}$ First, note this quantity clearly does not depend on ψ.

Let $k = |\tau|$. Recall $\mathcal{Q}$ ranges over the k multisets $\{P_i : i \in \mathcal{S}\}$ as $\mathcal{S}$ ranges over elements of the partition τ. By Hölder's Inequality and Lemma H.1,

$$\mathbb{E} \prod_{\mathcal{Q}} \|\partial^{\mathcal{Q}} u\|^{2p} \leq \prod_{\mathcal{Q}} \sqrt[k]{\mathbb{E}\,\|\partial^{\mathcal{Q}} u\|^{2pk}} \leq R \prod_{\mathcal{Q}} \sqrt[k]{\mathbb{E}\,\|\partial^{\mathcal{Q}} u\|_{2pk}^{2pk}}$$

where R is a constant depending only on $2pk$ and j (coming from Lemma H.1). Since $|\mathcal{Q}|$ and $k = |\tau|$ are both bounded by $|\mathbf{P}|$, it suffices to show $\mathbb{E}\,\|\partial^{\mathcal{Q}} u\|^{2pk}$ is bounded by a constant independent of α and is $(\mathrm{j}, p, k, \mathcal{Q})$-oblivious wrt ψ and the program. Now,

$$\begin{aligned}
\|\partial^{\mathcal{Q}} u\|_{2pk}^{2pk} &= \sum_{i=1}^{\mathrm{j}} (\partial^{\mathcal{Q}} g_\alpha^i)^{2pk} + (\partial^{\mathcal{Q}} c^i)^{2pk} \\
&\leq \sum_{i=1}^{\mathrm{j}} B_{i,2pk,|\mathcal{Q}|} + B_{i-1,2pk,|\mathcal{Q}|}^{\phi^i}
\end{aligned}$$

by Claim $1(1, \ldots, \mathrm{j})$ and Proposition J.7. This bound indeed is independent of α and has the required oblivious property.

J.4 Claim 2$(\mathrm{j}-1)$ Implies Claim 1(j)

Assume $\mathrm{j} \geq M_0 + 1$. Let $y_\beta \stackrel{\text{def}}{=} \phi^{\mathrm{j}}(g_\beta^1, \ldots, g_\beta^{\mathrm{j}-1}; c^1, \ldots, c^{\mathrm{j}-1})$. Recall $g_\alpha^{\mathrm{j}} = \sum_{\beta=1}^n W_{\alpha\beta}^{\mathrm{j}} y_\beta$, where W^{j} is one of the program's matrices A^i or their transposes.

Reduction via Product Rule Then by the product rule of differentiation,

$$\partial^{\mathcal{P}} g_\alpha^{\mathrm{j}} = \left(\sum_{\beta=1}^n W_{\alpha\beta}^{\mathrm{j}} \partial^{\mathcal{P}} y_\beta \right) + remainder$$

where $remainder$ is a sum of at most $|\mathcal{P}|$ elements of the form $\partial^{\mathcal{P}'} y_\beta$ where $\mathcal{P}'$ is $\mathcal{P}$ with some element removed. Therefore, by Lemma H.2,

$$|\partial^{\mathcal{P}} g_\alpha^{\mathrm{j}}|^p \leq R \left[\left| \sum_{\beta=1}^n W_{\alpha\beta}^{\mathrm{j}} \partial^{\mathcal{P}} y_\beta \right|^p + |remainder|^p \right]$$

where R depends only on p. We can easily bound $\mathbb{E}\,|remainder|^p$ by a $(\mathrm{j}, p, \mathcal{P})$-oblivious constant independent of α using Claim $2(\mathrm{j}-1)$ and Lemma H.2. Thus, it suffices to bound $\left| \sum_{\beta=1}^n W_{\alpha\beta}^{\mathrm{j}} \partial^{\mathcal{P}} y_\beta \right|^p$ by a constant with the same property.

Plan: Show Small Moment Condition (SMC) Holds To do so, we will show the vector with entries $w_\beta z_\beta$ where $w_\beta \stackrel{\text{def}}{=} W^{\text{j}}_{\alpha\beta}$ and $z_\beta \stackrel{\text{def}}{=} \partial^{\mathcal{P}} y_\beta$ satisfies the Small Moment Condition (Definition I.3) and apply Lemma I.4 to it. In particular, we will assume WLOG that p is an even integer and prove that for every multiset $\mathcal{N}$ of $[n]$ of size $|\mathcal{N}| = p$, we have

$$\mathbb{E}\, w^{\mathcal{N}} z^{\mathcal{N}} \le C n^{-|\mathrm{uniq}(\mathcal{N})|} \tag{66}$$

for a constant C that is $(\text{j}, \mathcal{N}, \mathcal{P})$-oblivious, where $\mathrm{uniq}(\mathcal{N})$ is the set of unique elements of $\mathcal{N}$.

Taylor Expansion of $z^{\mathcal{N}}$ Fix $\mathcal{N}$. We now consider $z^{\mathcal{N}}$ as a function of w_β for unique elements $\beta \in \mathrm{uniq}(\mathcal{N})$ of $\mathcal{N}$ (keeping other weight entries fixed). Let $v = (w_\beta)_{\beta \in \mathrm{uniq}(\mathcal{N})} \in \mathbb{R}^{|\mathrm{uniq}(\mathcal{N})|}$ be the vector of such elements, so that we write $z^{\mathcal{N}} = z^{\mathcal{N}}(v)$ as function of v. By Lemma H.8, we Taylor expand $z^{\mathcal{N}}$ to the rth order, for some r to be determined later:

$$\begin{aligned} z^{\mathcal{N}} &= z^{\mathcal{N}}(0) + \langle \nabla z^{\mathcal{N}}(0), v\rangle + \cdots + \frac{1}{r!}\langle \nabla^r z^{\mathcal{N}}(0), v^{\otimes r}\rangle + R_{r+1} \\ &= R_{r+1} + \sum_{s=0}^{r} \frac{1}{s!}\langle \nabla^s z^{\mathcal{N}}(0), v^{\otimes s}\rangle \\ \text{where} \quad R_{r+1} &\stackrel{\text{def}}{=} \frac{1}{r!}\int_0^1 (1-t)^r \langle \nabla^{r+1} z^{\mathcal{N}}(tv), v^{\otimes (r+1)}\rangle dt \end{aligned}$$

While the tensors appearing in this expansion may seem at first like "large objects", note that in terms of n, the tensors have constant sizes, so their norms are entirely determined by how their entries scale with n. Intuitively, the derivative tensors $\nabla^s z^{\mathcal{N}}$ will have $O(1)$ entry sizes, by induction hypothesis, while $v^{\otimes s}$ has size $O(n^{-s/2})$. But before we keep following this logic of naive bounds, it pays to notice there are a lot of cancellation.

Cancellation Using Independence and Zero-Mean Because now $\nabla^s z^{\mathcal{N}}(0)$ no longer depends on and thus is independent (as a random variable) from v (and thus $w^{\mathcal{N}}$)[25], taking expectation we now have

$$\mathbb{E}\, w^{\mathcal{N}} z^{\mathcal{N}} = \mathbb{E}\, w^{\mathcal{N}} R_{r+1} + \sum_{s=0}^{r} \frac{1}{s!}\langle \mathbb{E}\, \nabla^s z^{\mathcal{N}}(0), \mathbb{E}\, w^{\mathcal{N}} v^{\otimes s}\rangle$$

where $w^{\mathcal{N}} v^{\otimes s}$ is the scalar multiplication of the scalar $w^{\mathcal{N}}$ with the tensor $v^{\otimes s}$.

Now suppose

$$\mathcal{N} \text{ has exactly } k \text{ elements that appear singly (i.e., have multiplicity 1) in } \mathcal{N}.$$

Then $\mathbb{E}\, w^{\mathcal{N}} v^{\otimes s}$ will be 0 for all $s < k$: indeed, every entry of $w^{\mathcal{N}} v^{\otimes s}$ is a monomial $w^{\mathcal{N}'}$ that will have some w_β appearing by itself in the product (i.e., has degree 1), so that

$$\mathbb{E}\, w^{\mathcal{N}'} = \mathbb{E}\, w_\beta \mathbb{E}\, w^{\mathcal{N}' \setminus \{\beta\}} = 0$$

using the fact that w_β is zero-mean and independent from $w^{\mathcal{N}' \setminus \{\beta\}}$.

Therefore, we will take r (the order of the Taylor expansion) to be $k-1$, so that

$$\mathbb{E}\, w^{\mathcal{N}} z^{\mathcal{N}} = \mathbb{E}\, w^{\mathcal{N}} R_{k+1}.$$

Unwinding the definition of R_{k+1} and using Cauchy-Schwarz, we have

$$\mathbb{E}\, w^{\mathcal{N}} z^{\mathcal{N}} \le \frac{1}{(k-1)!}\int_0^1 (1-t)^{k-1} \sqrt{\mathbb{E}\, \|\nabla^k z^{\mathcal{N}}(tv)\|^2 \cdot \mathbb{E}\, \|w^{\mathcal{N}} v^{\otimes k}\|^2}\, dt \tag{67}$$

[25] this was the main purpose of the Taylor expansion

Constructing the SMC Constant Let $\mathcal{N}_1$ be the subset of elements of $\mathcal{N}$ with multiplicity 1 (so that $|\mathcal{N}_1| = k$) and let $\mathcal{N}' = \mathcal{N} \setminus \mathcal{N}_1$. We will show that

$$\mathbb{E}\, w^{\mathcal{N}} z^{\mathcal{N}} \leq B_k n^{-|\mathrm{uniq}(\mathcal{N})|} \tag{68}$$

for some constant B_k that is $(\mathrm{j}, \mathcal{N}_1, \mathcal{N}', \mathcal{P})$-oblivious. In particular, this means B_k depends on $\mathcal{N}$ only through $|\mathcal{N}| = |\mathcal{N}_1| + |\mathcal{N}'| = p$ and $|\mathcal{N}_1| = k$. Then the constant C in Eq. (66) can be taken as $\max_k B_k$ where k ranges from 0 to p.

In light of Eq. (67), to prove Eq. (68), it thus suffices to show that

$$\mathbb{E}\, \|\nabla^k z^{\mathcal{N}}(tv)\|^2 = O(1)$$
$$\mathbb{E}\, \|w^{\mathcal{N}} v^{\otimes k}\|^2 = O(n^{-2|\mathrm{uniq}(\mathcal{N})|})$$

where the big-Os hide $(\mathrm{j}, \mathcal{N}_1, \mathcal{N}', \mathcal{P})$-oblivious constants that furthermore are independent of $t \in [0, 1]$.

Bounding $\mathbb{E}\, \|w^{\mathcal{N}} v^{\otimes k}\|^2$ Each entry of the tensor $w^{\mathcal{N}} v^{\otimes k}$ is just a product of $|\mathcal{N}| + k = p + k$ matrix entries, whose expected square norm can be bounded by $\nu_{2(k+p)} n^{-(k+p)}$, where $\nu_{2(k+p)}$ is the scaled moment bound on the matrix entries we fixed at the beginning of this proof. There are $|\mathrm{uniq}(\mathcal{N})|^k \leq p^k$ entries in this tensor, so

$$\mathbb{E}\, \|w^{\mathcal{N}} v^{\otimes k}\|^2 \leq p^k \nu_{2(k+p)} n^{-(k+p)}$$

Now note that, by the definition of k, we have $k + p \geq 2|\mathrm{uniq}(\mathcal{N})|$. Thus

$$\mathbb{E}\, \|w^{\mathcal{N}} v^{\otimes k}\|^2 = O(n^{-2|\mathrm{uniq}(\mathcal{N})|})$$

where the constant in $O(-)$ is $(p, \mathcal{N}_1, \mathcal{N}', \mathcal{P})$-oblivious.

Bounding $\mathbb{E}\, \|\nabla^k z^{\mathcal{N}}(tv)\|^2$. Recall that all bounds in this proof (Appendix J) are oblivious wrt the program, so they only depend on the program through the bounds $(\nu_q)_q, (R_q)_q, \mathcal{C}$ fixed at the beginning of this proof. Notice that $\nabla^k z^{\mathcal{N}}(tv)$ is just $\nabla^k z^{\mathcal{N}}$ computed in the program where the matrix entries $\{W^{\mathrm{j}}_{\alpha\beta} : \beta \in \mathcal{N}\}$ (which are the entries of v) are scaled *down* and such a program satisfies the exact same data (in particular the moment bounds given by $(\nu_q)_q$).

Thus, by Proposition J.9, *every* entry of $\nabla^k z^{\mathcal{N}}(tv)$ has the same $(\mathrm{j}, \mathcal{N}_1, \mathcal{N}', \mathcal{P})$-oblivious bound C on its expected square norm, *uniformly* over $t \in [0, 1]$. Because $\nabla^k z^{\mathcal{N}}(tv)$ has $|\mathrm{uniq}(\mathcal{N})|^k \leq p^k$ entries,

$$\mathbb{E}\, \|\nabla^k z^{\mathcal{N}}(tv)\|^2 \leq C p^k$$

which is $(\mathrm{j}, \mathcal{N}_1, \mathcal{N}', \mathcal{P})$-oblivious and independent of $t \in [0, 1]$, as desired.

K Program Transformations

K.1 Backpropagation Program

Given a program T as in Eq. (4) and a polynomially smooth function $\psi : \mathbb{R}^{2M} \to \mathbb{R}$, we can create a new program T_ψ that extends T, which we call the *backpropagation program of T with respect to ψ*, or just *backprogram* for short. Intuitively, T_ψ will compute the gradients of

$$c \stackrel{\text{def}}{=} \frac{1}{n} \sum_{\alpha=1}^{n} \psi(g^1_\alpha, \ldots, g^M_\alpha; c^1, \ldots, c^M) \tag{69}$$

with respect to all vectors in the program. In the context of this work, the importance of the backprogram construction is to easily express the partial derivative $\delta = \frac{\partial c}{\partial A^j_{\alpha\beta}}$ (Proposition K.2), which easily shows that $\delta = O(n^{-1})$ *instead of* $O(1)$ *as suggested by Proposition J.7* (see Lemma K.6). This will be crucial in the proof of Theorem 5.2.

Explicitly, T_ψ is constructed as follows. It has the same matrices A^i as T. The first M vectors and scalars g^i and c^i for $i = 1, \ldots, M$ are the same as in T. It additionally has new vectors and scalars constructed after them. We will first describe the *mathematical objects* that will be computed by these new vectors and scalars, before discussing how to represent them in the form of Eq. (4).

Notation In this context, we use Sans Serif font to represent these mathematical objects, to distinguish them from objects in the program itself. Recall that ϕ^i is the nonlinearity used in iteration i in the original program T (as well as in the new program T_ψ). For $M \geq j > i$, we will write $\mathsf{x}^j_{i;} \in \mathbb{R}^n$ and $\mathsf{x}^j_{;i} \in \mathbb{R}^n$ for the vectors with entries

$$(\mathsf{x}^j_{i;})_\alpha \stackrel{\text{def}}{=} \partial_{g^i}\phi^j(g^1, \ldots, g^{j-1}; c^1, \ldots, c^{j-1})$$
$$(\mathsf{x}^j_{;i})_\alpha \stackrel{\text{def}}{=} \partial_{c^i}\phi^j(g^1, \ldots, g^{j-1}; c^1, \ldots, c^{j-1}).$$

In other words, $\mathsf{x}^j_{i;}$ is the partial derivative $\partial_{g^i}\phi^j$ with respect to the argument g^i and likewise $\mathsf{x}^j_{;i}$ is the partial derivative $\partial_{c^i}\phi^j$ with respect to the argument c^i, both evaluated on the original set of inputs for ϕ^j. For convenience, we will also write $\mathsf{x}^{M+1}_{i;}$ and $\mathsf{x}^{M+1}_{;i}$ for the partial derivatives of ψ, i.e.,

$$(\mathsf{x}^{M+1}_{i;})_\alpha \stackrel{\text{def}}{=} \partial_{g^i}\psi(g^1, \ldots, g^M; c^1, \ldots, c^M)$$
$$(\mathsf{x}^{M+1}_{;i})_\alpha \stackrel{\text{def}}{=} \partial_{c^i}\psi(g^1, \ldots, g^M; c^1, \ldots, c^M).$$

In addition, for a given vector $v \in \mathbb{R}^n$, we write $\langle v \rangle \stackrel{\text{def}}{=} \frac{1}{n}\sum_{\alpha=1}^n v_\alpha$ for the average of the entries of v.

Mathematical Idea We iteratively construct the vectors $\mathsf{dx}^i, \mathsf{dg}^i \in \mathbb{R}^n$ for decreasing $i = M, M-1, \ldots, M_0+1$ as follows.

$$\mathsf{dx}^{M+1} \stackrel{\text{def}}{=} 1 \in \mathbb{R}^n$$
$$\mathsf{dg}^i \stackrel{\text{def}}{=} \sum_{k=i+1}^{M+1} \mathsf{x}^k_{i;} \odot \mathsf{dx}^k + \sum_{M+1 \geq j > k > i} \langle \mathsf{dx}^j \odot \mathsf{x}^j_{;k} \rangle x^k_{i;} \tag{70}$$
$$\mathsf{dx}^i \stackrel{\text{def}}{=} W^{i\top}\mathsf{dg}^i \tag{71}$$

where $W^{i\top}$ in the last line is the transpose of the same matrix used in iteration i of the original program T, as in Eq. (4). For example, the first few dg^i and dx^i looks like the following.

$$\begin{aligned}
\mathsf{dx}^{M+1} &= 1 \\
\mathsf{dg}^M &= \mathsf{x}^{M+1}_{M;} \\
\mathsf{dx}^M &= W^{M\top}\mathsf{dg}^M \\
\mathsf{dg}^{M-1} &= \mathsf{x}^{M+1}_{M-1;} + \mathsf{x}^M_{M-1;} \odot \mathsf{dx}^M + \langle \mathsf{x}^{M+1}_{;M} \rangle \mathsf{x}^M_{M-1;} \\
\mathsf{dx}^{M-1} &= W^{M-1\top}\mathsf{dg}^{M-1}
\end{aligned}$$

One can verify the following statement using the chain rule.

Proposition K.1. *For $i = M, M-1, \ldots, M_0+1$, the vector dx^i (Eq. (71)) equals the scaled total gradient $n\nabla_{x^i}c$ of c (scaled up by n) against the vector x^i defined in Eq. (4), and likewise dg^i (Eq. (70)) equals $n\nabla_{g^i}c$.*

The vectors dx^i and dg^i make it easy to express the partial derivative $\frac{\partial c}{\partial A^i_{\alpha\beta}}$ as well:

Proposition K.2. *For any matrix A^j of program T, for ψ, c as in Eq. (69) and dg^i as in Eq. (70), we have*

$$\frac{\partial c}{\partial A^j_{\alpha\beta}} = \sum_{i:W^i=A^j} \frac{1}{n}(\mathsf{dg}^i)_\alpha (x^i)_\beta + \sum_{i:W^i=A^{j\top}} \frac{1}{n}(\mathsf{dg}^i)_\beta (x^i)_\alpha,$$

where in the first sum we iterate over indices i such that $W^i = A^j$ and in the second, i such that $W^i = A^{j\top}$, both in the context of the original program T.

Proof. Each A^j is used in the computation of c only through any W^i that equals A^j or its transpose. Thus

$$\frac{\partial c}{\partial A^j_{\alpha\beta}} = \sum_{i:W^i=A^j} \frac{\partial c}{\partial W^i_{\alpha\beta}} + \sum_{i:W^i=A^{j\top}} \frac{\partial c}{\partial W^i_{\beta\alpha}}$$

But, from Proposition K.1 and chain rule, one easily calculates

$$\frac{\partial c}{\partial W^i_{\alpha\beta}} = \frac{1}{n}(\mathsf{dg}^i)_\alpha (x^i)_\beta.$$

□

Program T_ψ Construction Here we will construct the vectors g^a for $a = M+1, M+2, \ldots$ for the new program T_ψ. To avoid confusion with vectors in the original program, we use superscript index a, b, e for talking about the new vectors, while i, j, k are reserved for indices of the old program. Our goal is to express each dx^i as some g^a with dg^i being the corresponding x^a (c.f. Eq. (4)). To do so, as is apparent from Eq. (70), we need to express $\langle \mathsf{dx}^j \odot \mathsf{x}^j_{;k} \rangle$ for each $M+1 \geq j > k$ as scalars c^b as well.

So the strategy is to, in descending order of i, alternatingly express dx^i, then the scalars $\langle \mathsf{dx}^j \odot \mathsf{x}^j_{;k} \rangle$ for $M+1 \geq j > k > i-1$, then express dg^{i-1} through Eq. (70) and finally dx^{i-1} through Eq. (71), and repeat. At the end, the vectors of T_ψ contain $g^1, \ldots, g^M$ (same as in T) and $\mathsf{dx}^{M+1} \ldots, \mathsf{dx}^{M_0+1}$ (new vectors) and the scalars contain $c^1, \ldots, c^M$ (same as in T) and $\{\langle \mathsf{dx}^j \odot \mathsf{x}^j_{;k} \rangle\}_{M+1 \geq j > k > M_0}$ (new scalars). There are also other "junk" vectors (resp. scalars) that we don't care about, which arise when we only want to express some scalar (resp. vectors). This shows

Proposition K.3. *If T has $M - M_0$ iterations, then T_ψ has at most $(M - M_0)^2$ iterations.*

Combining this with the fact that the nonlinearities of T_ψ are just compositions of polynomials with first order derivatives of those of T, we deduce the following

Proposition K.4. *Any $(\mathcal{P}_1, \ldots, \mathcal{P}_r)$-oblivious constant wrt T_ψ and $\psi^1, \ldots, \psi^l$ is also $(\mathcal{P}_1, \ldots, \mathcal{P}_r)$-oblivious wrt T and $\psi^1, \ldots, \psi^l$.*

It's clear from Eq. (70) that dg^i can be expressed as some nonlinearity applied to vectors of T_ψ. Beyond this fact, the exact construction of T_ψ is not important for our purposes, so we will not detail it further here.

Proposition K.5. *For any program T and c, ψ as in Eq. (69), each $\mathsf{dg}^i = n\nabla_{g^i} c$ can be expressed as some nonlinearity ϕ applied to the vectors and scalars of T_ψ. If all nonlinearities of T are polynomially smooth, then ϕ is polynomially smooth as well.*

Matrix Derivative Bound

Lemma K.6. *Consider a program in Setup J.2. Then for any $p \geq 1$, any nonempty multiset $\mathcal{P}$ taking values in the program's matrix entries $\{A^i_{\alpha\beta}\}_{\alpha,\beta,i}$, and any scalar c of the program,*

$$\mathbb{E}\left|\partial^{\mathcal{P}} c\right|^p = O(n^{-p}) \quad \text{as } n \to \infty. \tag{72}$$

Furthermore, the constant in the big-O is $(p, \mathcal{P})$-oblivious wrt the program.

Note importantly that $\mathcal{P}$ has to be nonempty for this to hold, since without taking derivatives, c can definitely be $\Theta(1)$ (for example, if its limit $\mathring{c}$ is nonzero). This lemma improves on Lemma J.6 drastically when $\mathcal{P}$ is nonempty. The reason that Lemma J.6 is so loose in such cases is that, in the sum over $\alpha \in [n]$ in Eq. (64), only a constant number of α really achieves the sup bound in Lemma J.3. So the naive way Lemma J.6 converts the sup bound to an average bound turned out to leave a lot of room.

Proof. Suppose c is the ith scalar. For $i \leq M_0$, the derivative is 0, so consider the case where $i > M_0$. For brevity, write ψ for ϕ^i (the nonlinearity used to create c^i). Construct the backprogram T_ψ with respect to ψ. By Proposition K.4, it suffices to show the bound for a constant $(p, \mathcal{P})$-oblivious wrt T_ψ.

Since $\mathcal{P}$ is nonempty, there is an element a of $\mathcal{P}$. Write $\mathcal{P}' = \mathcal{P} \setminus \{a\}$. Then by Proposition K.2,

$$\partial^{\mathcal{P}} c = \partial^{\mathcal{P}'}(\partial_a c) = \frac{1}{n}\sum \partial^{\mathcal{P}'}[(\mathsf{dg}^j)_\alpha (x^j)_\beta],$$

where the sum is over some collection of (j, α, β) of size at most $M - M_0$ (an upper bound on how many times a has been used in the original program).

Then a routine combination of Lemma H.2, product rule (Lemma H.11), and Cauchy-Schwarz reduces our problem to bounding $\mathbb{E}\,|\partial^{\mathcal{Q}}(\mathsf{dg}^j)_\alpha|^{2p}$ and $\mathbb{E}\,|\partial^{\mathcal{Q}}(x^j)_\beta|^{2p}$ for all subsets $\mathcal{Q}$ of $\mathcal{P}'$ by a $(p, \mathcal{Q})$-oblivious constant independent of α and β. This is precisely provided by Lemma J.3 applied to the backprogram T_ψ. □

Lemma K.7. *Consider a program in Setup J.2. Let $\mathcal{Q}_1, \ldots, \mathcal{Q}_r$ be nonempty and let $\mathcal{P}$ be potentially empty multisets of matrix entries. Then for any scalar c of the program,*

$$\mathbb{E}\left|\partial^{\mathcal{P}} \prod_{i=1}^{r} \partial^{\mathcal{Q}_i} c\right|^p = O(n^{-rp})$$

where the hidden constant is $(p, \mathcal{Q}_1, \ldots, \mathcal{Q}_r, \mathcal{P})$-oblivious wrt the program.

Proof. The product rule (Lemma H.11), Lemma H.2, and Hölder's Inequality show that the LHS is, within a constant factor depending only on p and $|\mathcal{P}|$, bounded by a sum of $O(1)$ number (depending only on p) of terms, each of which is a product over r elements of the form

$$\mathbb{E}\,|\partial^{\mathcal{Q}} c|^{rp}$$

for multisets $\mathcal{Q}$ with size at most $|\mathcal{P}| + \max_i |\mathcal{Q}_i|$. Then the desired result follows from Lemma K.6.

□

L Proof of Theorem 5.2

We will prove the following more specific version of Theorem 5.2, which says more about the hidden constant. Recall the dot derivative notation from Section 5 as well as the notion of *oblivious constants* in Definition J.1.

Theorem L.1. *Consider a program in Setup 3.6 and its interpolation (Definition 5.1), and let c be a scalar in it. Then for any finite $p \geq 1$,*

$$\sup_t \mathbb{E}\,\dot{c}(t)^p = O(n^{-p/2})$$

where the hidden constant is p-oblivious wrt the program.

Proof. By the power-mean inequality, it suffices to show this for any even integer p. Let $a = a(t)$ be the vector of all matrix entries in the program, which is an interpolation of the Gaussian and non-Gaussian matrix entries. Let N be its dimension (which is equal to n^2 times the number of matrices in the program). We will use κ to index a's entries. Denote its derivative against t by $\dot{a} \in \mathbb{R}^N$, as in Section 5. For brevity, we will suppress the argument (t) below.

Let $D \stackrel{\text{def}}{=} \nabla_a c \in \mathbb{R}^N$, so that for any multiset $\mathcal{P}$ taking values in $[N]$, $D^{\mathcal{P}} = \prod_{\kappa \in \mathcal{P}} \frac{\partial c}{\partial a_\kappa}$. By chain rule, we have

$$\dot{c} = \langle D, \dot{a} \rangle = \langle 1, D \odot \dot{a} \rangle.$$

From here on, the proof follows an outline similar to that of Appendix J.4, except in how the cancellation happens.

Proof Plan: Small Moment Condition. We will show that the N-dimensional vector $\sqrt{n} D \odot \dot{a}$ (which has entries $\sqrt{n} D_\kappa \dot{a}_\kappa$) satisfies the Small Moment Condition (Definition I.3): For any even $p \geq 0$ and any multiset $\mathcal{P}$ of size p taking values in $[N]$, we need to show there's a $\mathcal{P}$-oblivious constant C wrt the program that is also independent of t such that

$$n^{p/2} D^{\mathcal{P}} \dot{a}^{\mathcal{P}} \leq C N^{-|\mathrm{uniq}(\mathcal{P})|}. \tag{73}$$

Then by Lemma I.4, we have

$$n^{p/2} \mathbb{E}\,|\dot{c}|^p = \mathbb{E}\,|\langle 1, \sqrt{n} D \odot \dot{a} \rangle|^p = O(1)$$

where the hidden constant is independent of $t \in [0, 1]$, which yields Theorem L.1.

Taylor Expansion. Now fix $\mathcal{P}$. Let $v \stackrel{\text{def}}{=} (a_\kappa)_{\kappa \in \mathrm{uniq}(\mathcal{P})}$ be the vector of unique a_κ for $\kappa \in \mathcal{P}$ (for any ordering of $\mathrm{uniq}(\mathcal{P})$). We think of $D^{\mathcal{P}}$ as a function $D^{\mathcal{P}}(v)$ of v. For an integer r to be specified later, we Taylor expand $D^{\mathcal{P}}$ in v around 0 to the rth order (Lemma H.8): with ∇^i denoting differentation wrt v,

$$D^{\mathcal{P}}(v) = D^{\mathcal{P}}(0) + \langle \nabla D^{\mathcal{P}}(0), v\rangle + \cdots + \frac{1}{r!}\langle \nabla^r D^{\mathcal{P}}(0), v^{\otimes r}\rangle + R_{r+1}$$

$$\text{where} \quad R_{r+1} \stackrel{\text{def}}{=} \frac{1}{r!}\int_0^1 (1-\zeta)^r \langle \nabla^{r+1} D^{\mathcal{P}}(\zeta v), v^{\otimes (r+1)}\rangle d\zeta.$$

Cancellation Using Independence and Zero-Mean Now notice $\nabla^i D^{\mathcal{P}}(0)$ is independent from $v^{\otimes i}$ and $\dot{a}^{\mathcal{P}}$. Therefore, for $i = 0, \ldots, r$,

$$\mathbb{E}\,\langle \nabla^i D^{\mathcal{P}}(0), \dot{a}^{\mathcal{P}} v^{\otimes i}\rangle = \langle \mathbb{E}\,\nabla^i D^{\mathcal{P}}(0), \mathbb{E}\,\dot{a}^{\mathcal{P}} v^{\otimes i}\rangle.$$

Furthermore, notice $\mathbb{E}\,\dot{a}^{\mathcal{P}} v^{\otimes i} = 0$ for small values of i. Indeed, if

$$k \text{ is the number of elements in } \mathcal{P} \text{ appearing exactly once}$$

and $i < 2k$, then every entry of the expected tensor $\mathbb{E}\,\dot{a}^{\mathcal{P}} v^{\otimes i}$ has the form

$$(\mathbb{E}\,\dot{a}_\kappa)\cdot(other) \quad \text{or} \quad (\mathbb{E}\,\dot{a}_\kappa a_\kappa)\cdot(other)$$

for some a_κ where $other$ is the expectation of some other monomial in a and $\dot{a}$ independent from a_κ and $\dot{a}_\kappa$. Both cases evaluate to 0 by Lemma 5.3. Thus, we now choose r (the degree of Taylor expansion) to be $2k-1$. Then

$$\mathbb{E}\,\dot{a}^{\mathcal{P}} D^{\mathcal{P}}(v) = \mathbb{E}\,\dot{a}^{\mathcal{P}} R_{2k}.$$

By Lemma H.2 and Cauchy-Schwarz, unwinding the definition of R_{2k}, we now have

$$\mathbb{E}\,\dot{a}^{\mathcal{P}} D^{\mathcal{P}}(v) \le \frac{1}{(2k-1)!}\int_0^1 (1-\zeta)^{2k-1}\sqrt{\mathbb{E}\,\|\nabla^{2k} D^{\mathcal{P}}(\zeta v)\|^2 \cdot \mathbb{E}\,\|\dot{a}^{\mathcal{P}} v^{\otimes 2k}\|^2}\, d\zeta \tag{74}$$

Let $\mathcal{P}_1$ be the subset of $\mathcal{P}$'s elements appearing in $\mathcal{P}$ exactly once, so that $|\mathcal{P}_1| = k$. Let $\mathcal{P}' = \mathcal{P} \setminus \mathcal{P}_1$. So it suffices to show

$$\sqrt{\mathbb{E}\,\|\dot{a}^{\mathcal{P}} v^{\otimes 2k}\|^2} = O(n^{-(2k+p)/2})$$
$$\sqrt{\mathbb{E}\,\|\nabla^{2k} D^{\mathcal{P}}(\zeta v)\|^2} = O(n^{-p})$$

where the hidden constants a) are $(\mathcal{P}_1, \mathcal{P}')$-oblivious and b) are independent of t (the interpolation variable) and ζ (the integration variable in the Taylor remainder R_{2k}). If these are shown, then, plugging into Eq. (74),

$$n^{p/2}\mathbb{E}\,\dot{a}^{\mathcal{P}} D^{\mathcal{P}}(v) = O(n^{-k-p}) = O(n^{-2|\mathrm{uniq}(\mathcal{P})|}) = O(N^{-|\mathrm{uniq}(\mathcal{P})|})$$

where the second equality follows because k is the number of elements of $\mathcal{P}$ with multiplicity 1 and $p = |\mathcal{P}|$, thus $k + p \ge 2|\mathrm{uniq}(\mathcal{P})|$. The hidden constant here will depend on $k = |\mathcal{P}_1|$, but we can take the max of such constants over all $k = 0, 1, \ldots, p$ to arrive at the desired $\mathcal{P}$-oblivious constant in Eq. (73).

Bounding $\dot{a}^{\mathcal{P}} v^{\otimes 2k}$. Each entry of $\dot{a}^{\mathcal{P}} v^{\otimes 2k}$ is a degree $2k + |\mathcal{P}| = 2k + p$ monomial in $\dot{a}$ and a. Thus, by Lemma 5.3, its expected square norm is bounded by $O(n^{-(2k+p)})$ uniformly over all entries, where the hidden constant depends only on the scaled moment bound ν_{2k+p} of the program's matrix entries. Because $\dot{a}^{\mathcal{P}} v^{\otimes k}$ has $|\mathrm{uniq}(\mathcal{P})|^{2k} \le p^{2k} = O(1)$ entries, we have

$$\mathbb{E}\,\|\dot{a}^{\mathcal{P}} v^{\otimes 2k}\|^2 = O(n^{-(2k+p)})$$

where the hidden constant is $(\mathcal{P}_1, \mathcal{P}')$-oblivious, as desired.

Bounding $\nabla^{2k} D^{\mathcal{P}}(\zeta v)$. We can think of $\nabla^{2k} D^{\mathcal{P}}(\zeta v)$ as $\nabla^{2k} D^{\mathcal{P}}$ computed on a program where the matrix entries in v are multiplied by a factor of $0 \le \zeta \le 1$. This program would then satisfy the same scaled moment bounds $(\nu_j)_{j=2}^{\infty}$ as the original program, uniformly over all t. Thus all of our oblivious bounds would apply to $\nabla^{2k} D^{\mathcal{P}}(\zeta v)$. In particular, Lemma K.7 tells us each entry of $\nabla^{2k} D^{\mathcal{P}}(\zeta v)$ has expected square norm bounded uniformly (over all entries) by $O(n^{-2|\mathcal{P}|}) = O(n^{-2p})$ with a $(\mathcal{P}_1, \mathcal{P}')$-oblivious constant independent of ζ and t. Finally, because there are $|\mathrm{uniq}(\mathcal{P})|^{2k} \le p^{2k} = O(1)$ entries, we get

$$\mathbb{E}\, \|\nabla^{2k} D^{\mathcal{P}}(\zeta v)\|^2 = O(n^{-2p})$$

$(\mathcal{P}_1, \mathcal{P}')$-obliviously and uniformly over ζ and t as desired.

□

M L^p Convergence From Almost Sure Convergence

The following is a standard lemma.

Lemma M.1. *For any nonnegative random variable $X \in \mathbb{R}$ and deterministic $B \ge 0$, we have*

$$\mathbb{E}\, X \mathbb{I}(X > B) = B \Pr[X > B] + \int_B^{\infty} \Pr[X > t] dt$$

Proof. Integration by parts. □

Lemma M.2. *Suppose a sequence of random variables $(X_n)_{n=1}^{\infty}$ converges almost surely to 0. Then for any $p \in [1, \infty)$, X_n converges to 0 in L^p as well if for some $q > p$, there exists a constant C such that $\mathbb{E}\, |X_n|^q < C$ for all n.*

This lemma's proof is a standard truncation trick.

Proof. For any $B > 0$, we have

$$\mathbb{E}\, |X_n|^p = \mathbb{E}\, |X_n|^p \mathbb{I}(|X_n|^p \le B) + \mathbb{E}\, |X_n|^p \mathbb{I}(|X_n|^p > B)$$

The random variable $|X_n|^p \mathbb{I}(|X_n|^p \le B)$ converges almost surely to 0, so by dominated convergence, it also converges in mean:

$$\mathbb{E}\, |X_n|^p \mathbb{I}(X_n \le B) \to 0.$$

This convergence happens for any fixed $B > 0$. So it suffices to show that $\sup_n \mathbb{E}\, |X_n|^p \mathbb{I}(|X_n|^p > B)$ becomes arbitrarily small as $B \to \infty$.

By Markov's Inequality, we have

$$\Pr[|X_n|^p > t] = \Pr[|X_n|^q > t^{q/p}] \le \frac{\mathbb{E}\, |X_n|^q}{t^{q/p}}$$

By Lemma M.1,

$$\begin{aligned} \mathbb{E}\, |X_n|^p \mathbb{I}(|X_n|^p > B) &= B \Pr[|X_n|^p > B] + \int_B^{\infty} \Pr[|X_n|^p > t] dt \\ &\le \frac{\mathbb{E}\, |X_n|^q}{B^{q/p-1}} + \int_B^{\infty} \frac{\mathbb{E}\, |X_n|^q}{t^{q/p}} dt. \end{aligned}$$

Since $\mathbb{E}\, |X_n|^q < C$ and $q/p > 1$ by assumption, this quantity goes to 0 as $B \to \infty$, as desired. □

Theorem M.3. *Consider a program in Setup J.2 and a scalar c in the program. If c converges almost surely to a limit $\mathring{c}$, then c converges in L^p to $\mathring{c}$ as well for every $p \in [1, \infty)$.*

Proof. This follows from Lemma M.2 and Lemma J.6. □

N Empirical validation

In the present section, we validate our main result, Theorem 3.7, by empirically checking some of its corollaries mentioned in Section 4.

N.1 On non-Gaussian biases and input and output layers

Here we first note a very important subtlety concerning distribution universality. When expressing neural network computations (i.e. forward and backward passes) as a tensor program, we use matrices for initial hidden weights and vectors for biases and initial input and output weights. Indeed, since for input and output layers, only one dimension grows to infinity, they cannot be expressed as $n \times n$ matrices in the program for which both dimensions go to infinity. Instead, they are expressed as a set of vectors.

As noted in Section 4, we model non-Gaussian vector variables as images of Gaussian ones under elementwise nonlinear maps. This requires adding a nonlinearity to a program, which may alter the limit in Theorem 3.7.

This means that the NNGP and NTK kernels of Corollaries 4.3 and 4.4 are not necessarily distribution-invariant. However both corollaries are still true as they merely state that these kernels exist.

N.2 Setup

We perform our experiments with vanilla RNN, vanilla GRU network, and a simple Transformer. We build on the code acompanying [29] and [30].

As for experiments aimed to validate NNGP correspondence, Corollary 4.3, we compute correlation matrices on two sentences, "The brown fox jumps over the dog" and "The quick brown fox jumps over the lazy dog", embedded using GloVe embeddings [22]. We compute the empirical kernel for different distributions and measure the relative Frobenius norm distance between this empirical kernel and the analytic kernel for Gaussian initialization. We compute mean and standard deviations for this relative Frobenius norm distance using 100 random initializations.

As for experiments aimed to validate convergence to a kernel method, Corollary 4.4, we compute empirical kernels on two "sentences", the first 5 pixels of the first CIFAR10 image and the first 5 pixels of the second CIFAR10 image. We compute it for different distributions and measure the relative Frobenius norm distance between this empirical kernel and the analytic neural tangent kernel. Same as for NNGP correspondence, we compute mean and standard deviations for this relative Frobenius norm distance using 100 random initializations.

Following Setup 3.6, we consider distributions with zero mean and variance $1/n$. The distributions we consider are Gaussian, Gaussian truncated at 2σ, uniform, Laplace, and the other one which we call *Cubecauchy*. The Cubecauchy distribution is a distribution of a cubic root of a Cauchy random variable. Such a random variable has finite mean and variance, but does not have any other moments. Since it does not have all moments, it does not follow Setup 3.6, which makes our Theorem 3.7 inapplicable. However, as we shall see shortly, the empirical kernels we consider still converge to the same limit suggesting that existence of all moments is not necessary.

N.3 Results

We start with empirically validating NNGP correspondence, Corollary 4.3. We first swap only hidden weights with their non-Gaussian counterparts. In this case, our Theorem 3.7 predicts that the limit should not depend on matrix entry distributions. As we see in Fig. 1, this is indeed the case. Moreover, the empirical kernel that corresponds to the Cubecauchy distribution still converges to the same limit as for the other distributions, thus suggesting that coincidence of only the first two moments are necessary for the Master theorem to hold.

What happens if we swap all other trainable parameters, i.e. biases, input weights, embeddings, with their non-Gaussian counterparts? As discussed above, the Master theorem still holds, but it requires modifying the program itself which may alter the limit. As we observe in Fig. 2, sometimes it is indeed the case (left plot), while sometimes the limit could be the same (right plot). We note that even

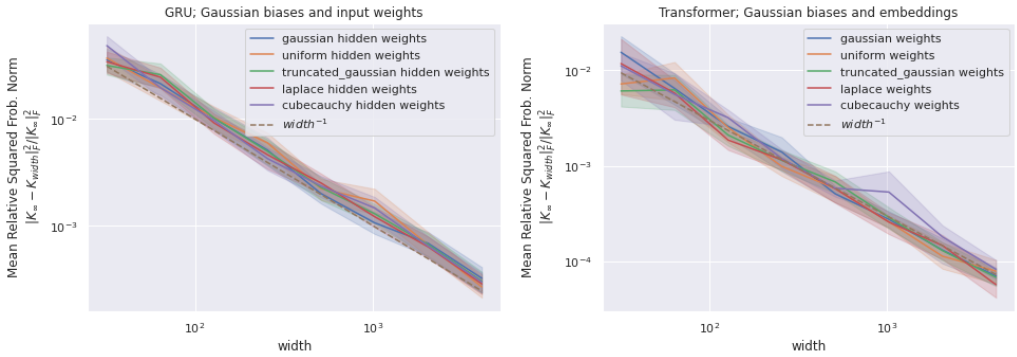

Figure 1: Relative Frobenius norm distance between empirical NNGP kernels for different distributions and the analytic kernel for Gaussian initialization. All trainable parameters expressed with vectors (i.e. biases, embeddings, and input weights) are assumed to be Gaussian. Left: a simple GRU network, right: a simple transformer, see Appendix N.2. As we see, NNGP kernels for all distributions considered converge to the same analytic kernel.

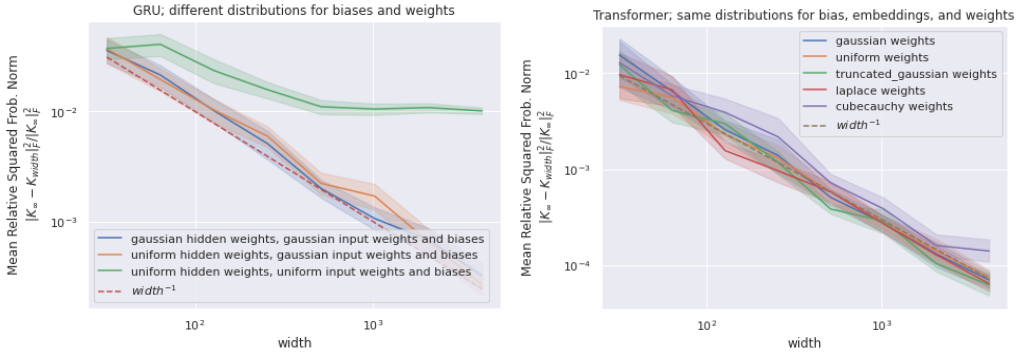

Figure 2: Relative Frobenius norm distance between empirical NNGP kernels for different distributions and the analytic kernel for Gaussian initialization. We assume all trainable parameters to have the same distribution. Left: a simple GRU network, right: a simple transformer, see Appendix N.2. As we see, in some cases (left, green line), the limit NNGP kernel may depend on the parameter distribution since we had to modify the corresponding tensor program in order to model non-Gaussian vector variables, see Appendix N.1. Note that the limit kernel still exists and there is no contradiction with Corollary 4.3.

Figure 3: Relative Frobenius norm distance between empirical neural tangent kernels for different distributions and the analytical neural tangent kernel for Gaussian initialization. We assume all trainable parameters to have the same distribution. Left: a simple recurrent neural network, right: a simple transformer, see Appendix N.2. As we see, the limit NTK does not depend on the parameter distribution, even when we use non-Gaussian vectors, see discussion in Appendix N.1.

when the limit is different from the one resulted from Gaussian weights, Fig. 2 still demonstrates that the limit exists even for non-Gaussian ones, therefore validating Corollary 4.3.

Next, we validate neural tangent kernel convergence at initialization, checking a part of Corollary 4.4. As we see in Fig. 3, even when all trainable parameters are initialized with non-Gaussians, the limit kernel still remains the same.