Appendix organization The appendix is organized as follows:
In Appendix A we expands upon the examples given in Section 3, while adding some additional details.
In Appendix B we introduce the formal version of the NETSOR$\top$,NETSOR$\top^+$ programs.
In Appendix C we introduce the graphical notation of NETSOR$\top^+$ and demonstrate other examples of architectures or computations expressible in Tensor Programs.
In Appendix D we prove our main result.

Notations For the readers convenience we restate the notations described in Section 2, along with some additional ones which will be used throughout the appendix. We will consider SGD with batch size 1 and learning rate of 1 (WLOG).We use ξ_t to denote the input and $\mathcal{L}_t$ to denote the loss function (absorbing the label) at step t. More generally, subscript t on any symbol means *time* t. However, for brevity, we abuse notation and shorthand f_t for $f_t(\xi_t)$, and, for any (pre-)activation x, x_t for $x_t(\xi_t)$. We will also write χ_t for the loss derivative $\mathcal{L}_t'(f_t)$. For any vector $x(\xi)$ we define $\delta x_{t+1}(\xi) \overset{\text{def}}{=} \sqrt{n}\big(x_{t+1}(\xi) - x_t(\xi)\big)$ and $dx(\xi) \overset{\text{def}}{=} \sqrt{n}\frac{\partial f(\xi)}{\partial x(\xi)}$. We will track the evolution of f on an arbitrary input $\tilde{\xi}$.[22] Similar to above, we shorthand $\tilde{x}_t, \tilde{f}_t$ for $x_t(\tilde{\xi}), f_t(\tilde{x})$. In general, omitting the time index t for any time dependent quantity implies its value at initialization. (i.e $x(\xi) = x_0(\xi), f(\xi) = f_0(\xi)$). Finally, we use $\triangleq$ to imply equality of symbols (i.e $W^1 \triangleq W^2$ iff W^1, W^2 represent the same variable, as opposed to equality in value).

A. Additional Examples

In this section we flesh out the examples given in Section 3 of the main text with the purpose of adding additional clarity, while maintaining the intuitive arguments as presented in each example to perform these calculations. The rigorous justification for these calculations will be given in the following section with the formal introduction of the Tensor Program framework.

Recall that our objective is to derive Claim 3.1 by tracking the coordinate distribution of each (pre-)activations vector $x(\xi), dx(\xi) \overset{\text{def}}{=} \sqrt{n}\frac{\partial f(\xi)}{\partial x(\xi)}, \delta x(\xi) \overset{\text{def}}{=} \sqrt{n}\big(x_{t+1}(\xi) - x_t(\xi)\big)$.

Claim 3.1. *In the large width limit, $\tilde{f}_t = f_t(\tilde{\xi})$ changes by*

$$\lim_{n\to\infty} \tilde{f}_{t+1} - \tilde{f}_t = -\mathring{\chi}_t \mathring{\mathcal{K}}(\tilde{\xi}, \xi_t) \tag{3}$$

at step t, where $\mathring{\mathcal{K}}$ is the limiting NTK of the architecture and $\mathring{\chi}_t = \mathcal{L}_t'(\lim_n f_t)$ is the loss derivative.

In our calculations we will rely on the following rules relating to the coordinates of any $\mathbb{R}^n$ (pre-)activation vector $x(\xi)$ in the large width regime, which we will later formalize:

- $x_{t+1}(\xi) - x_t(\xi)$ has $\Theta(\frac{1}{\sqrt{n}})$ coordinates.
- $\delta x_{t+1}(\xi)$ has $\Theta(1)$ coordinates.
- $x_t(\xi) = x(\xi) + o(1)$. Consequently $Z^{x_t(\xi)} = Z^{x(\xi)}$.
- If $x(\xi) = \phi\big(y(\xi)\big)$ for some vector $y(\xi) \in \mathbb{R}^n$, then by Taylor approximation $\delta x_{t+1}(\xi) = \sqrt{n}\big(\phi(y_t(\xi) + \frac{\delta y_{t+1}(\xi)}{\sqrt{n}}) - \phi(y_t(\xi))\big) \approx \phi'\big(y_t(\xi)\big) \odot \delta y_{t+1}(\xi)$. Consequently $Z^{\delta x_{t+1}(\xi)} = \phi'(Z^{y_t(\xi)})Z^{\delta y_{t+1}(\xi)}$.

Remark A.1. We write Z^x to denote the limit coordinate distribution of $x \in \mathbb{R}^n$ conditioned on the output function f at initialization. Consequently we write $\mathbb{E}\, X^x Z^y$ to express a conditional expectation given the output function f. See Appendix B for the formal statement.

A.1. 1 hidden layer

Recall our model is of the form:

$$f = V^\top x, \quad x = \phi(h), \quad h = U\xi$$

where $\xi \in \mathbb{R}^d$, $U = \frac{u}{\sqrt{d}} \in \mathbb{R}^{n\times d}$, $V = \frac{v}{\sqrt{n}} \in \mathbb{R}^{n\times 1}$, for trainable parameter tensor u, initialized iid from $\mathcal{N}(0,1)$, and $d = 1$.

[22]It might help to think of $\tilde{\xi}$ as some *test sample*, but it can also fall in the training set.

Deriving The NTK The infinite width NTK of this architecture is given by:

$$\mathcal{K}(\xi,\tilde{\xi}) = \langle \nabla_u f(\xi), \nabla_u f(\tilde{\xi})\rangle = \xi^\top \tilde{\xi} \frac{dh(\xi)^\top dh(\tilde{\xi})}{n} \tag{27}$$

$$dh(\xi) \overset{\text{def}}{=} \sqrt{n}\frac{\partial f(\xi)}{\partial h(\xi)} = \phi'(h(\xi)) \odot v. \tag{28}$$

Therefore, by LLN it follows:[23]

$$\mathring{\mathcal{K}}(\xi,\tilde{\xi}) = \xi^\top \tilde{\xi} \lim_{n\to\infty} \frac{dh(\xi)^\top dh(\tilde{\xi})}{n} = \xi^\top\tilde{\xi}\, \mathbb{E}\, \phi'(Z^{h(\tilde{\xi})})\phi'(Z^{h(\xi)}). \tag{29}$$

Getting Claim 3.1 To show that Claim 3.1 holds with the kernel in Eq. (29), we track the coordinate distribution $Z^{\delta \tilde{x}_{t+1}}$ at each step of SGD. At step t, the update to the weights $u_{t+1} - u_t$ is given by the gradient of the loss with respect to u_t:

$$u_{t+1} - u_t = -\chi_t \frac{dh_t \xi_t^\top}{\sqrt{n}}, \quad dh_t = \phi'(h_t) \odot v \tag{30}$$

Recall that $\delta\tilde{h}_{t+1} = \sqrt{n}(\tilde{h}_{t+1} - \tilde{h}_t) = \sqrt{n}(u_{t+1}\tilde{\xi} - u_t\tilde{\xi})$ and $\delta\tilde{x}_{t+1} = \sqrt{n}(\tilde{x}_{t+1} - \tilde{x}_t)$. It therefore follows:

$$\delta\tilde{h}_{t+1} = -\chi_t\xi_t^\top\tilde{\xi}\phi'(h_t)\odot v, \quad \delta\tilde{x}_{t+1} = \sqrt{n}\big(\phi(\tilde{h}_t + \frac{\delta\tilde{h}_{t+1}}{\sqrt{n}}) - \phi(\tilde{h}_t)\big). \tag{31}$$

Since $h_t = \Theta(1)$ and $\delta\tilde{h}_{t+1} = \Theta(1)$, for large n we may Taylor expand ϕ to first order around $\tilde{h}_t$:

$$\delta\tilde{x}_{t+1} \approx \sqrt{n}\big(\phi(\tilde{h}_t) + \frac{1}{\sqrt{n}}\phi'(\tilde{h}_t)\odot\delta\tilde{h}_{t+1} - \phi(\tilde{h}_t)\big) \tag{32}$$

$$= \phi'(\tilde{h}_t)\odot\delta\tilde{h}_{t+1} \tag{33}$$

$$= -\chi_t\xi_t^\top\tilde{\xi}\phi'(\tilde{h}_t)\odot\phi'(h_t)\odot v. \tag{34}$$

Again since $\delta h_t(\xi) = \Theta(1)$, it follows that $h_t(\xi) = h(\xi) + \sum_{s=1}^t \frac{\delta h_s(\xi)}{\sqrt{n}} = h(\xi) + o(1)$. Hence, in the infinite width limit the coordinate distribution of $h_t(\xi)$ is identical to the coordinate distribution of $h(\xi)$ (i.e $Z^{h_t(\xi)} = Z^{h(\xi)}$). Using Eq. (34), the coordinate distribution of $\delta\tilde{x}_{t+1}$ is given by:

$$Z^{\delta\tilde{x}_{t+1}} = -\mathring{\chi}_t\xi_t^\top\tilde{\xi}\phi'(Z^{\tilde{h}_t})\phi'(Z^{h_t})Z^v. \tag{35}$$

In the large width limit, the change in the output is simply given by $\tilde{f}_{t+1} - \tilde{f}_t = \frac{v^\top\delta\tilde{x}_{t+1}}{n} = \mathbb{E}\, Z^{\delta\tilde{x}_{t+1}}Z^v$. Using Eq. (35) and the independence of Z^v from the other random variables,[24]

$$\mathbb{E}\, Z^{\delta\tilde{x}_{t+1}}Z^v = -\,\mathbb{E}\,\mathring{\chi}_t\xi_t^\top\tilde{\xi}\phi'(Z^{\tilde{h}_t})\phi'(Z^{h_t})(Z^v)^2 \tag{36}$$

$$= -\mathring{\chi}_t\xi_t^\top\tilde{\xi}\,\mathbb{E}\,\phi'(Z^{\tilde{h}_t})\phi'(Z^{h_t}) \tag{37}$$

$$= -\mathring{\chi}_t\mathring{\mathcal{K}}(\xi_t,\tilde{\xi}). \tag{38}$$

A.2. 2 hidden layers

Recall our model is of the form:

$$f = V^\top x, \quad x = \phi(h), \quad h = Wz$$
$$z = \phi(g), \quad g = U\xi$$

where $U = \frac{u}{\sqrt{d}} \in \mathbb{R}^{n\times d}$, $W = \frac{w}{\sqrt{n}} \in \mathbb{R}^{n\times n}$, $V = \frac{v}{\sqrt{n}} \in \mathbb{R}^{n\times 1}$, for trainable parameters u, w, initialized iid from a normal distribution. As before we assume the last layer is not trained, and $d = 1$.

[23]While in Remark A.1, we said $\mathbb{E}$ denotes expectation conditioned on $\lim f_0$, the NTK here does not actually depend on $\lim f_0$.

[24]Again, as in Footnote 10, the expectations in Appendices A.1 and A.2 should more rigorously be interpreted as expectation conditional on the function values of $\lim f_0$.

Deriving The NTK The infinite width NTK is given by:

$$\mathcal{K}(\xi,\tilde{\xi}) = \langle \nabla_u f(\xi), \nabla_u f(\tilde{\xi})\rangle + \langle \nabla_w f(\xi), \nabla_w f(\tilde{\xi})\rangle \tag{39}$$

$$= \xi^\top \tilde{\xi}\frac{dg(\xi)^\top dg(\tilde{\xi})}{n} + \frac{z(\xi)^\top z(\tilde{\xi})}{n}\frac{dh(\xi)^\top dh(\tilde{\xi})}{n} \tag{40}$$

$$dh(\xi) \overset{\text{def}}{=} \sqrt{n}\frac{\partial f(\xi)}{\partial h(\xi)} = \phi'\big(h(\xi)\big) \odot v \tag{41}$$

$$dg(\xi) \overset{\text{def}}{=} \sqrt{n}\frac{\partial f(\xi)}{\partial g(\xi)} = \phi'\big(g(\xi)\big) \odot \big(W^\top dh(\xi)\big). \tag{42}$$

Naively using LLN on Eq. (39) (and Z^v being independent from everything else) should result in:

$$\mathcal{K}(\xi,\tilde{\xi}) = \xi^\top\tilde{\xi}\,\mathbb{E}\left[Z^{dg(\xi)}Z^{dg(\tilde{\xi})}\right] + \mathbb{E}\left[Z^{z(\xi)}Z^{z(\tilde{\xi})}\right]\mathbb{E}\left[\phi'(Z^{h(\xi)})\phi'(Z^{h(\tilde{\xi})})\right]. \tag{43}$$

Evaluating the term $\mathbb{E}\left[Z^{dg(\xi)}Z^{dg(\tilde{\xi})}\right]$ however presents a challenge since $dh(\xi)$ depends on both W and $W^\top$. As it turns out, at initialization we may naively assume that $W^\top, W$ are independent (formally known in the literature as gradient independence assumption, or GIA) [25], we arrive using simple LLN arguments to:

$$\mathbb{E}\left[Z^{dg(\xi)}Z^{dg(\tilde{\xi})}\right] = \mathbb{E}[\phi'(Z^{g(\xi)})\phi'(Z^{g(\tilde{\xi})})]\,\mathbb{E}[\phi'(Z^{h(\xi)})\phi'(Z^{h(\tilde{\xi})})]. \tag{44}$$

Plugging Eq. (44) into Eq. (43) we arrive at the correct expression for the infinite width NTK.

Getting Claim 3.1 To show that Claim 3.1 holds at any step t (where we may not assume that GIA holds), we track the distributions of the vectors $g(\xi), z(\xi), h(\xi), x(\xi)$ throughout training.

At any step t the weights are updated according to:

$$u_{t+1} - u_t = -\chi_t\frac{dg_t\xi_t^\top}{\sqrt{n}}, \quad w_{t+1} - w_t = -\chi_t\frac{dh_t z_t^\top}{n}. \tag{45}$$

The update $\delta\tilde{g}_{t+1} \overset{\text{def}}{=} \sqrt{n}(\tilde{g}_{t+1} - \tilde{g}_t), \delta\tilde{z}_{t+1} \overset{\text{def}}{=} \sqrt{n}(\tilde{z}_{t+1} - \tilde{z}_t)$ are given by:

$$\delta\tilde{g}_{t+1} = -\chi_t dg_t\xi_t^\top\tilde{\xi}, \quad \delta\tilde{z}_{t+1} = \sqrt{n}\big(\phi(\tilde{g}_t + \frac{\delta\tilde{g}_{t+1}}{\sqrt{n}}) - \phi(\tilde{g}_t)\big)\big). \tag{46}$$

As before, with large n we have that $\bullet_{t+1}(\xi) - \bullet_t(\xi) \sim \Theta(\frac{1}{\sqrt{n}})$ and $\delta\bullet_{t+1}(\xi) \sim \Theta(1)$ coordinates for $\bullet$ replaced by $\{g, z, h, x\}$. And so after Taylor expanding $\phi(\tilde{g}_t + \frac{\delta\tilde{g}_{t+1}}{\sqrt{n}})$ around $\tilde{g}_t$:

$$\delta\tilde{z}_{t+1} \approx \phi'(\tilde{g}_t)\odot\delta\tilde{g}_{t+1}. \tag{47}$$

In a similar fashion, using Eqs. (41) and (46) the updates $\delta\tilde{z}_{t+1}, \delta\tilde{x}_{t+1}, \delta\tilde{x}_{t+1}$ take the form:

$$\delta\tilde{z}_{t+1} \approx \phi'(g_t)\odot\delta\tilde{g}_{t+1} = -\chi_t\xi_t^\top\tilde{\xi}\phi'(g_t)\odot\phi'(\tilde{g}_t)\odot\big(W^\top dh_t\big) \tag{48}$$

$$\delta\tilde{h}_{t+1} \approx W\delta\tilde{z}_{t+1} - \chi_t\frac{z_t^\top\tilde{z}_t}{n}\phi'(h_t)\odot v - \frac{1}{\sqrt{n}}\sum_{s=0}^{t}\chi_s\frac{z_s^\top\delta\tilde{z}_{t+1}}{n}\phi'(h_s)\odot v \tag{49}$$

$$\delta\tilde{x}_{t+1} \approx \phi'(\tilde{h}_t)\odot\delta\tilde{h}_{t+1}. \tag{50}$$

where we used Eq. (45) and

$$\delta\tilde{h}_{t+1} = W_{t+1}\delta\tilde{z}_{t+1} + \sqrt{n}(W_{t+1} - W_t)\tilde{z}_t \tag{51}$$

$$= W\delta\tilde{z}_{t+1} + \sum_{s=0}^{t}(W_{s+1} - W_s)\delta\tilde{z}_{t+1} + \sqrt{n}(W_{t+1} - W_t)\tilde{z}_t \tag{52}$$

[25]For a rigorous justification of the GIA assumption see (Yang, 2020a)

to get Eq. (49). Based on Eqs. (41) and (48) to (50), the corresponding coordinate distributions take the form:

$$Z^{dh_t} = \phi'(Z^{h_t})Z^v \tag{53}$$

$$Z^{\delta\tilde{z}_{t+1}} = -\mathring{\chi}_t \xi_t^\top \tilde{\xi} \phi'(Z^{g_t})\phi'(Z^{\tilde{g}_t})Z^{W^\top dh_t} \tag{54}$$

$$Z^{\delta\tilde{h}_{t+1}} = Z^{W\delta\tilde{z}_{t+1}} - \mathring{\chi}_t \mathbb{E}[Z^{z_t}Z^{\tilde{z}}]\phi'(Z^{h_t})Z^v \tag{55}$$

$$Z^{\delta\tilde{x}_{t+1}} = \phi'(Z^{\tilde{h}_t})Z^{\delta\tilde{h}_{t+1}}. \tag{56}$$

As before, the functional update is given by $\lim_{n\to\infty} \tilde{f}_{t+1} - \tilde{f}_t = \lim_{n\to\infty} \frac{v^\top \delta\tilde{x}_{t+1}}{n} = \mathbb{E}\, Z^v Z^{\delta\tilde{x}_{t+1}}$. Plugging Eqs. (55) and (56):

$$\mathbb{E}\, Z^v Z^{\delta\tilde{x}_{t+1}} = -\mathring{\chi}_t \,\mathbb{E}\left[Z^{z_t}Z^{\tilde{z}}\right] \mathbb{E}\left[\phi'(Z^{h_t})\phi'(Z^{\tilde{h}_t})\right] - \mathbb{E}\left[\phi'(Z^{h_t})Z^{W\delta\tilde{z}_{t+1}}Z^v\right]. \tag{57}$$

To compute the second term of the RHS of Eq. (57), we use Claim 3.2, reproduced below.

Claim 3.2. *Based on the above discussion and some easy calculations, $\delta\tilde{z}_{t+1}$ can be written as $\Phi(W^\top dh_t)$ for some $\Phi : \mathbb{R} \to \mathbb{R}$ applied coordinatewise (which will depend on other vectors not of the form $W^\top \bullet$). Then it turns out*[25]

$$Z^{W\delta\tilde{z}_{t+1}} = G + Z^{dh_t}\,\mathbb{E}\,\frac{\partial Z^{\delta\tilde{z}_{t+1}}}{\partial Z^{W^\top dh_t}}, \tag{17}$$

where G is some Gaussian variable independent from Z^v, and $\frac{\partial Z^{\delta\tilde{z}_{t+1}}}{\partial Z^{W^\top dh_t}} \overset{def}{=} \Phi'(Z^{W^\top dh_t})$.

Applying Claim 3.2 to get the expression for $Z^{W\delta\tilde{z}_{t+1}}$:

$$Z^{W\delta\tilde{z}_{t+1}} = G + Z^{dh_t}\,\mathbb{E}\,\frac{\partial Z^{\delta\tilde{z}_{t+1}}}{\partial Z^{W^\top dh_t}} \tag{58}$$

$$= G - \phi'(Z^{h_t})Z^v \mathring{\chi}_t \xi_t^\top \tilde{\xi}\,\mathbb{E}\left[\phi'(Z^{g_t})\phi'(Z^{\tilde{g}_t})\right] \tag{59}$$

As before, for $h \in \{g, z, h, x\}$ it holds that $h_t(\xi) = h(\xi) + \sum_{s=1}^{t} \frac{\delta h_s(\xi)}{\sqrt{n}} = h(\xi) + o(1)$, and $Z^{h_t(\xi)} = Z^{h(\xi)}$. Plugging Eq. (58) into Eq. (57) yields Claim 3.1.

B. Tensor Programs: the Formal Version

We briefly review the formal definition of Tensor Programs below, but readers needing more explanation and intuition should see (Yang, 2020b). We will directly describe NETSOR⊤⁺ programs, which generalizes NETSOR⊤.

Definition B.1. A NETSOR$\top^+$ program is a sequence of $\mathbb{R}^n$-vectors and $\mathbb{R}$-scalars inductively generated via one of the following ways from an initial set $\mathcal{C}$ of random scalars, $\mathcal{V}$ of random $\mathbb{R}^n$ vectors, and a set $\mathcal{W}$ of random $\mathbb{R}^{n\times n}$ matrices (which will be sampled with iid Gaussian entries in Setup B.2)

MATMUL same as MATMUL in Definition 4.1.

NONLIN$^+$ Given $\phi : \mathbb{R}^k \times \mathbb{R}^l \to \mathbb{R}$, previous scalars $\theta_1, \ldots, \theta_l \in \mathbb{R}$ and vectors $x^1, \ldots, x^k \in \mathbb{R}^n$, we can generate a new vector

$$\psi(x^1, \ldots, x^k; \theta_1, \ldots, \theta_l) \in \mathbb{R}^n \tag{60}$$

where $\psi(-; \theta_1, \ldots, \theta_l)$ applies coordinatewise to each "α-slice" $(x^1_\alpha, \ldots, x^k_\alpha)$.

MOMENT Given same setup as above, we can also generate a new scalar

$$\frac{1}{n}\sum_{\alpha=1}^{n} \psi(x^1_\alpha, \ldots, x^k_\alpha; \theta_1, \ldots, \theta_l) \in \mathbb{R}. \tag{61}$$

A NETSOR$\top$ program is just a NETSOR$\top^+$ program without scalars, without the usage of MOMENT, and without parameters $\theta_1, \ldots, \theta_l$ in NONLIN$^+$.

We will typically randomly sample the initial matrices, vectors, and scalars of the program as follows.

Setup B.2. *1) For each initial $W \in \mathcal{W}$, we sample iid $W_{\alpha\beta} \sim \mathcal{N}(0, \sigma_W^2/n)$ for some variance σ_W^2 associated to W, independent of other $W' \in \mathcal{W}$; 2) for some multivariate Gaussian $Z^{\mathcal{V}} = \left\{Z^h : h \in \mathcal{V}\right\} \in \mathbb{R}^{\mathcal{V}}$, we sample the initial set of vectors $\mathcal{V}$ like $\{h_\alpha : h \in \mathcal{V}\} \sim Z^{\mathcal{V}}$ iid for each $\alpha \in [n]$. 3) For each initial scalar $\theta \in \mathcal{C}$, we require $\theta \xrightarrow{\text{a.s.}} \mathring{\theta}$ for some deterministic $\mathring{\theta} \in \mathbb{R}$.*

The following constructs a random variable Z^h for every vector h and a deterministic scalar $\mathring{\theta}$ for every scalar θ in the program. The interpretation is that h will have iid coordinates distributed like Z^h, and θ will converge to $\mathring{\theta}$ as $n \to \infty$.

Definition B.3 (Z^h and $\mathring{\theta}$)**.** Given a NETSOR$\top^+$ program, we recursively define Z^h for each vector h and $\mathring{\theta}$ for each scalar θ as follows.

ZINIT If $h \in \mathcal{V}$, then Z^h is defined as in Setup B.2. We also set $\hat{Z}^h \stackrel{\text{def}}{=} Z^h$ and $\dot{Z}^h \stackrel{\text{def}}{=} 0$.

ZNONLIN$^+$ Given $\psi : \mathbb{R}^k \times \mathbb{R}^l \to \mathbb{R}$, previous scalars $\theta_1, \ldots, \theta_l \in \mathbb{R}$ and vectors $x^1, \ldots, x^k \in \mathbb{R}^n$, we have

$$Z^{\psi(x^1,\ldots,x^k;\theta_1,\ldots,\theta_l)} \stackrel{\text{def}}{=} \psi(Z^{x^1}, \ldots, Z^{x^k}; \mathring{\theta}_1, \ldots, \mathring{\theta}_l). \tag{62}$$

ZMOMENT Given same setup as above and scalar $\theta = \frac{1}{n}\sum_{\alpha=1}^n \psi(x_\alpha^1, \ldots, x_\alpha^k; \theta_1, \ldots, \theta_l)$, then

$$\mathring{\theta} \stackrel{\text{def}}{=} \mathbb{E}\,\psi(Z^{x^1}, \ldots, Z^{x^k}; \mathring{\theta}_1, \ldots, \mathring{\theta}_l). \tag{63}$$

Here $\mathring{\theta}_1, \ldots, \mathring{\theta}_l$ are deterministic, so the expectation is taken over $Z^{x^1}, \ldots, Z^{x^k}$.

ZMATMUL $Z^{Wx} \stackrel{\text{def}}{=} \hat{Z}^{Wx} + \dot{Z}^{Wx}$ for every matrix W (with $\mathcal{N}(0, \sigma_W^2/n)$ entries) and vector x, where

ZHAT $\hat{Z}^{Wx}$ is a Gaussian variable with zero mean. Let $\mathcal{V}_W$ denote the set of all vectors in the program of the form Wy for some y. Then $\{\hat{Z}^{Wy} : Wy \in \mathcal{V}_W\}$ is defined to be jointly Gaussian with zero mean and covariance

$$\operatorname{Cov}\left(\hat{Z}^{Wx}, \hat{Z}^{Wy}\right) \stackrel{\text{def}}{=} \sigma_W^2\,\mathbb{E}\,Z^x Z^y, \quad \text{for any } Wx, Wy \in \mathcal{V}_W. \tag{64}$$

Furthermore, $\{\hat{Z}^{Wy} : Wy \in \mathcal{V}_W\}$ is mutually independent from $\{\hat{Z}^v : v \in \mathcal{V} \cup \bigcup_{\bar{W} \neq W} \mathcal{V}_{\bar{W}}\}$, where $\bar{W}$ ranges over $\mathcal{W} \cup \{A^\top : A \in \mathcal{W}\}$.

ZDOT We can always unwind $Z^x = \Phi(\cdots)$, for some arguments $(\cdots) = (\{\hat{Z}^{W^\top y^i}\}_{i=1}^k, \{\hat{Z}^{z^i}\}_{i=1}^j; \{\mathring{\theta}_i\}_{i=1}^l)$, $z^i \notin \mathcal{V}_{W^\top}$ (where $\mathcal{V}_{W^\top}$ is defined in ZHAT), and deterministic function $\Phi : \mathbb{R}^{k+j+l} \to \mathbb{R}$. Define $\partial Z^x / \partial \hat{Z}^{W^\top y^i} \stackrel{\text{def}}{=} \partial_i \Phi(\cdots)$. Then we set

$$\dot{Z}^{Wx} \stackrel{\text{def}}{=} \sigma_W^2 \sum_{i=1}^k Z^{y^i}\,\mathbb{E}\,\frac{\partial Z^x}{\partial \hat{Z}^{W^\top y^i}}, \tag{65}$$

There is some nuance in this definition, so see Remark B.5 and B.6.

The following theorem ties the symbolic nature of the Zs to the analytic nature of a Tensor Program.

Theorem B.4 (NETSOR$\top^+$ Master Theorem, c.f. Theorem E.15 of (Yang, 2020b))**.** *Fix a Tensor Program initialized accordingly to Setup B.2. Adopt Assumption B.8. Then*

1. *For any fixed k and any pseudo-Lipschitz $\psi : \mathbb{R}^k \to \mathbb{R}$, as $n \to \infty$,*

$$\frac{1}{n}\sum_{\alpha=1}^n \psi(h_\alpha^1, \ldots, h_\alpha^k) \xrightarrow{\text{a.s.}} \mathbb{E}\,\psi(Z^{h^1}, \ldots, Z^{h^k}), \tag{66}$$

for any vectors $h^1, \ldots, h^k$ in the program, where Z^{h^i} are as defined in Definition B.3.

2. *Any scalar θ in the program tends to $\mathring{\theta}$ almost surely, where $\mathring{\theta}$ is as defined in Definition B.3.*

Remark B.5 (Partial derivative). The partial derivative in ZDOT should be interpreted as follows. By a simple inductive argument, Z^x for every vector x in the program is defined *uniquely* as a deterministic function $\varphi(\hat{Z}^{x^1}, \ldots, \hat{Z}^{x^k})$ of some $x^1, \ldots, x^k$ in $\mathcal{V}$ or introduced by MATMUL (notationally, we are suppressing the possible dependence on limit scalars $\mathring{\theta}_1, \ldots, \mathring{\theta}_l$). For instance, if in a program we have $A \in \mathcal{W}, v \in \mathcal{V}, y = Av, x = A^\top y$, then $Z^x = \hat{Z}^x + \hat{Z}^v$, so φ is given by $\varphi(a, b) = a + b$. Then

$$\partial Z^x / \partial \hat{Z}^{x^i} \stackrel{\text{def}}{=} \partial_i \varphi(\hat{Z}^{x^1}, \ldots, \hat{Z}^{x^k}), \quad \text{and} \quad \partial Z^x / \partial \hat{Z}^z \stackrel{\text{def}}{=} 0 \text{ for any } z \notin \{x^1, \ldots, x^k\}.$$

Note this definition depends on the precise way the program is written, not just on the underlying mathematics. For example, if $y, z \in \mathcal{V}$ and $x = \phi(W(y+z))$, then $Z^x = \phi(\hat{Z}^{W(y+z)})$ so that $\partial Z^x / \partial \hat{Z}^{Wy} = \partial Z^x / \partial \hat{Z}^{Wz} = 0$. If instead, we have $x = \phi(Wy + Wz)$, then $Z^x = \phi(\hat{Z}^{Wy} + \hat{Z}^{Wz})$ so that $\partial Z^x / \partial \hat{Z}^{W(x+y)} = 0$. However, in both cases, $\dot{Z}^{W^\top x} = (Z^y + Z^z) \mathbb{E}\, \phi'(\hat{Z}^{W(y+z)})$.

Remark B.6 (Partial derivative expectation). The quantity $\mathbb{E} \frac{\partial Z^x}{\partial \hat{Z}^{W^\top y}}$ is well defined if Z^x is differentiable in $\hat{Z}^{W^\top y}$. However, even if this is not the case, e.g. if $x = \theta(W^\top y)$ where θ is the Heavyside step function, we can still define this expectation by leveraging Stein's lemma:

In ZDOT, suppose $\{W^\top y^i\}_{i=1}^k$ are all elements of $\mathcal{V}_{W^\top}$ introduced before x. Define the matrix $C \in \mathbb{R}^{k \times k}$ by $C_{ij} \stackrel{\text{def}}{=} \mathbb{E}\, Z^{y^i} Z^{y^j}$ and define the vector $b \in \mathbb{R}^k$ by $b_i \stackrel{\text{def}}{=} \mathbb{E}\, \hat{Z}^{W^\top y^i} Z^x$. If $a = C^+ b$ (where C^+ denotes the pseudoinverse of C), then in ZDOT we may set

$$\sigma_W^2 \, \mathbb{E} \frac{\partial Z^x}{\partial \hat{Z}^{W^\top y^i}} = a_i. \tag{67}$$

This definition agrees with the partial derivative expectation by Stein's lemma when the latter is well defined. Theorem B.4 holds with this broader definition of partial derivative expectation.

Pseudo-Lipschitz functions are, roughly speaking, functions whose weak derivatives are polynomially bounded.

Definition B.7. A function $f : \mathbb{R}^k \to \mathbb{R}$ is called *pseudo-Lipschitz* of degree d if $|f(x) - f(y)| \leq C\|x - y\|(1 + \sum_{i=1}^k |x_i|^d + |y_i|^d)$ for some C. We say f is pseudo-Lipschitz if it is so for any degree.

Here are some basic properties of pseudo-Lipschitz functions:

- The norm $\|\cdot\|$ in Definition B.7 can be any norm equivalent to the ℓ_2 norm, e.g. $\ell_p, p \geq 1$, norms. Similarly, $\sum_{i=1}^k |x_i|^d + |y_i|^d$ can be replaced by $\|x\|_p^d + \|y\|_p^d$, for any $p \geq 1$.
- A pseudo-Lipschitz function is polynomially bounded.
- A composition of pseudo-Lipschitz functions of degrees d_1 and d_2 is pseudo-Lipschitz of degree $d_1 + d_2$.
- A pseudo-Lipschitz function is Lipschitz on any compact set.

We adopt the following assumption for the Master Theorem Theorem B.4.

Assumption B.8. *Suppose*

1. *If a function $\phi(;-) : \mathbb{R}^{0+l} \to \mathbb{R}$ with only parameter arguments is used in* MOMENT*, then ϕ is continuous in those arguments.*
2. *Any other function $\phi(-;-) : \mathbb{R}^{k+l} \to \mathbb{R}$ with parameters (where $k > 0$) used in* NONLIN *or* MOMENT *is pseudo-Lipschitz in all of its arguments (both inputs and parameters).*

Statement 1 in Assumption B.8 essentially says that if we have scalars $\theta_1, \ldots, \theta_l$ in the program, then we can produce a new scalar by applying a continuous function (a weaker restriction than a pseudo-Lipschitz function) to them. Indeed, if $\theta_1, \ldots, \theta_l$ converge almost surely, then this new scalar does too. In our setting, statement 1 is used to allow any loss function whose derivative is continuous.

Other versions of the Master Theorem can be found in (Yang, 2020b), for example, versions where the we do not assume any smoothness condition at all on the nonlinearities beyond that they be polynomially bounded, in exchange for assuming what's called a *rank stability* condition. This rank stability should be generically true, but checking it rigorously is subtle, so we are content with the pseudo-Lipschitz condition in this paper.

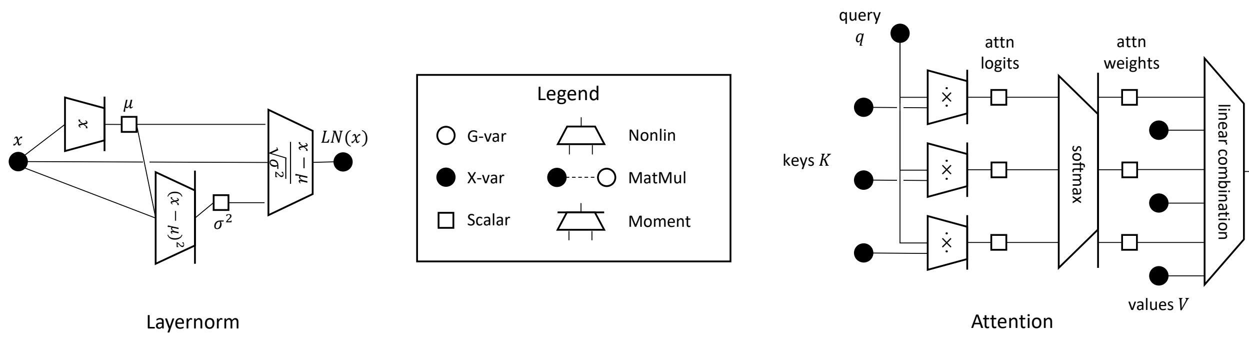

Figure 4: Layernorm and attention can be implemented with NETSOR$\top^+$.

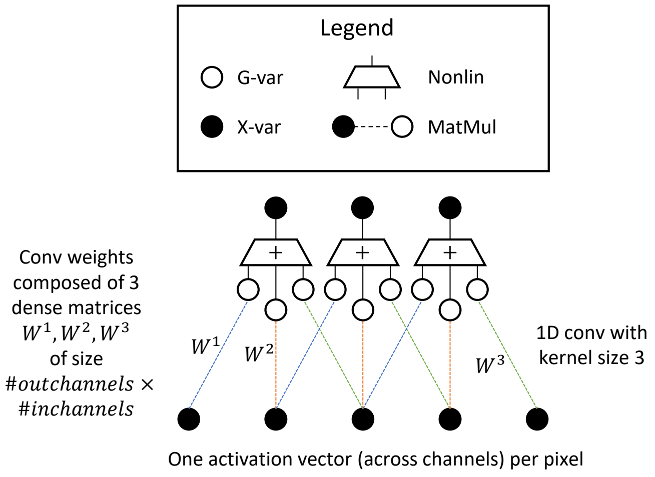
Legend
G-var
X-var
Nonlin
MatMul
Conv weights composed of 3 dense matrices W^1, W^2, W^3 of size #outchannels × #inchannels
W^1
W^2
W^3
1D conv with kernel size 3
One activation vector (across channels) per pixel

Figure 5: Convolution can be implemented with NETSOR$\top$.

C. More Diagrams

We can augment the graphical form of NETSOR$\top$ to accomodate the MOMENT instruction in NETSOR$\top^+$. See Fig. 4 for an example for layernorm and attention. In short, we denote scalar variables with a square, in contrast to the circle for vector variables, and we use a "bar-gate" to denote the MOMENT, where the function in the gate corresponds to ψ in MOMENT.

In addition, for more examples of the expressivity of NETSOR$\top$, Figs. 5 and 6 demonstrate convolution and MLP backpropagation in NETSOR$\top$.

D. Proof of Main Result

We dedicate the following section to prove Theorem 5.3. We will begin by proving a simplified version under the same assumptions as Section 5.1, as reproduced below:

Setup D.1 (Representable NN in NTK Parametrization)**.** *Suppose a neural network $f \in \mathbb{R}$ is represented by a* NETSOR$\top$ *program (in the sense of Definition 4.2) whose* NONLIN *all have polynomially bounded derivatives.*[26] *Adopt the NTK parametrization: for every matrix parameter $W \in \mathbb{R}^{n\times n}$ of f, we factor $W = \frac{1}{\sqrt{n}}w$ where w is the trainable parameter; likewise, for each input layer matrix $U^i \in \mathbb{R}^{n\times d}$, we factor $U^i = \frac{1}{\sqrt{d}}u^i$, and likewise the output matrix $V = \frac{1}{\sqrt{n}}v$. We randomly initialize all trainable parameters iid as $\mathcal{N}(0,1)$. Furthermore, we assume the following:*

A1. Input and output layers $\{u^i\}, v$, as well as biases are not trained (only $\mathbb{R}^{n\times n}$ weight matrices are trained).

A2. The forward pass does not use both a matrix and its transpose (in different MATMULs*).*

[26]More generally, we can allow any pseudo-Lipschitz function here, but for simplicity we go with the statement in the main text.

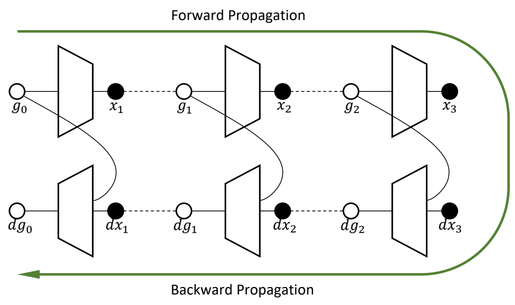

Figure 6: Backpropagation (of an MLP f) can be implemented in NETSOR$\top$. However, NETSOR$\top$ cannot express the loss derivative, so we cannot unroll multiple steps of SGD in NETSOR$\top$, unlike NETSOR$\top^+$.

A3. The output is a scalar $f \in \mathbb{R}$.

A4. We assume the last layer embedding is a G-var.

Our main result is to show that the SGD training of such a neural network described in Setup 5.2 reduces to kernel gradient descent with kernel $\mathring{\mathcal{K}}$ in the infinite-width limit.

Theorem D.2 (NTKTRAIN is Architecturally Universal)**.** *Consider training a network f described in Setup D.1 via SGD with batch-size 1 and (WLOG) learning rate 1. Let ξ_t be the input and $\mathcal{L}_t : \mathbb{R} \to \mathbb{R}$ be the loss function (absorbing the label) at time t. Suppose $\mathcal{L}_t$ is continuous for all t. Then, for any ξ and t, $f_t(\xi)$ converges almost surely to a random variable $\mathring{f}_t(\xi)$ as width $\to \infty$, such that*

$$\mathring{f}_{t+1}(\xi) - \mathring{f}_t(\xi) = -\mathring{\mathcal{K}}(\xi, \xi_t)\mathcal{L}_t'(\mathring{f}_t(\xi_t)) \tag{68}$$

where $\mathring{\mathcal{K}}$ is the infinite-width NTK (at initialization) of the neural network.

D.1. SGD as a NETSOR$\top^+$ Program

SGD is comprised of a sequence of forward and backward passes computed on some architecture. WLOG, let π_0 denote the *reduced program* implementing the body of network f, and let $x(\xi)$ denote the final embedding such that $f(\xi) = V^\top x(\xi)$, we will now show how the SGD procedure on π_0 can be implemented by a NETSOR$\top^+$ program.

D.1.1. FIRST FORWARD PASS

While π_0 implements the embeddings $x(\xi)$ by definition, the outputs $f(\xi)$ cannot be implemented trivially in a program since that at initialization $f(\xi) = \frac{v^\top x(\xi)}{\sqrt{n}}$ is not deterministic, and converges non-trivially to a GP, violating the requirements of a scalar type in a NETSOR$\top^+$ program which require all scalar types to converge to a deterministic limit as $n \to \infty$. Nevertheless, we can still easily express evolution of f *conditioned on (i.e. fixing) the values of f at initialization*. More formally, let $\mathtt{f} = [f(\xi_0), f(\xi_1), ..., f(\xi_{D-1})]^\top \in \mathbb{R}^D$ denote a fixed vector of outputs, and let $X = [x(\xi_0), x(\xi_1), ..., x(\xi_{D-1})]^\top \in \mathbb{R}^{n\times D}$ denote a fixed embedding matrix such that $\mathtt{f} = \frac{X^\top v}{\sqrt{n}}$. The distribution of v when conditioned on $\mathtt{f}$ and X is given by (see e.g. (Yang, 2020b, Sec K.2))

$$v \overset{\mathrm{d}}{=}_{\mathtt{f},X} \sqrt{n}X^+\mathtt{f} + \Pi\mathtt{v} \tag{69}$$

where X^+ is the pseudo-inverse of X, $\mathtt{v}$ is an independent copy of v and Π is the projection operator projecting unto the orthogonal complement of the space spanned by X. Namely:

$$X^+ = \frac{1}{n}X(\frac{X^\top X}{n})^+, \quad \Pi = I - X^+X^\top \tag{70}$$

Denote $\Sigma = \frac{X^\top X}{n} \in \mathbb{R}^{D\times D}, \mu = \frac{X^\top v}{n} \in \mathbb{R}^D$. Define

$$\hat{v} \overset{\text{def}}{=} X(\frac{\Sigma^+\mathtt{f}}{\sqrt{n}}) + \mathtt{v} - X\Sigma^+\mu. \tag{71}$$

Then we see via Eq. (69) that

$$v \overset{\mathrm{d}}{=}_{\mathtt{f},X} \hat{v}. \tag{72}$$

Given $\mathtt{v}$ and (the columns of) X as vectors and $\mathtt{f}$ as scalars in a program, $\hat{v}$ may be defined in the same program via NONLIN, where $\frac{\Sigma^+\mathtt{f}}{\sqrt{n}}$ and $\Sigma^+\mu$ (both finite-dimensional) provide coefficients for the linear combination over (columns of) X. Formally, to express the evolution of f conditioned on $f_0 = \mathtt{f}$ at initialization, the program will calculate the first forward pass up to X, calculate the loss derivatives χ assuming $f_0 = \mathtt{f}$, and then proceed with the backward pass and later forward/backward passes with v replaced by $\hat{v}$.

However, since $\frac{\Sigma^+\mathtt{f}}{\sqrt{n}}, \mu \xrightarrow{\text{a.s.}} 0$ and $\Sigma^+ \xrightarrow{\text{a.s.}} \mathring{\Sigma}^+$ (by rank stability, c.f. (Yang, 2020b, Lemma L.11)), these coefficients of the linear combination converge to 0, so that $Z^{\hat{v}} = Z^{\mathtt{v}}$. Intuitively, this means that the distribution of v conditioned on the equality $\mathtt{f} = X^\top v/\sqrt{n}$ is asymptotically the same as no conditioning as $n \to \infty$. Thus, for the limit calculation of δf_t and other quantities, it ends up not mattering whether we use $\hat{v}$ or v.

Computing The Loss Derivatives The loss derivative $\chi(\xi) = \frac{\partial \mathcal{L}(f(\xi))}{\partial(f(\xi))}$ after the first forward pass given $f(\xi)$ can be implemented with MOMENT instructions using $\psi(;f(\xi)) = \mathcal{L}'(f(\xi))$.

D.1.2. IMPLEMENTING SGD

Under SGD, the update at step $t+1$ to any weight $w \in \mathbb{R}^{n\times n}$ is given by:

$$w_{t+1} - w_t = -\chi_t \sum_{\mathtt{g},\mathtt{h}:\mathtt{g}=W\mathtt{h}} \frac{d\mathtt{g}_t \mathtt{h}_t^\top}{n}. \tag{73}$$

where the summation in Eq. (73) is over all pairs of vectors $\mathtt{g}, \mathtt{h}$ in program π_0 satisfying $\mathtt{g} = W\mathtt{h}$ (there can be multiple such pairs since π_0 may reuse the same matrix W).

To write the full unrolled SGD as a NETSOR$\top^+$ program, we will need to implement the error signal $dg_t \overset{\text{def}}{=} \sqrt{n}\frac{\partial f_t}{\partial g_t}$ for each G-var g at time t. To accomplish this, we recall the notion of paths in program π_0:

Definition 5.5 (Paths)**.** In a NETSOR$\top$ program, a path p starts with an X-var and ends with a G-var, alternating between X- and G-vars along the path. We write p^0 for the starting X-var, p^1 for the following G-var, and so on, as well as p^{-1} for the ending G-var (see Fig. 2 for a graphical illustration). For odd i, let W^{p^i} denote the defining matrix of G-var p^i. For two equal length paths p, q, we write $p \cong q$ (path p is isomorphic to path q) if for all odd i, W^{p^i} is the same matrix as W^{q^i}.[27] In other words, we say path p is isomorphic to path q if their sequences of MATMUL matrices are identical, (but the NONLIN don't have to be, see Fig. 3 for a graphical illustration). Let $|p|$ denote the number of vectors in p (this is always an even number).

Note that a path p represents a series of nodes independent of an input, and can be instantiated as $p(\xi)$ by an input ξ, resulting in a series of instantiated G-vars and X-vars $p^i(\xi)$.

For any G-var $g = Wh$, we can write the error term dg as the summation of errors signals over paths p:

$$dg(\xi) = \sum_{p:p^{-1}=x,p^1=g} J^p(\xi) \tag{74}$$

$$\text{where} \quad J^p = (\frac{\partial p^2}{\partial p^1})^\top (\frac{\partial p^{-2}}{\partial p^{-3}})^\top ... (\frac{\partial p^{-1}}{\partial p^{-2}})^\top v \tag{75}$$

(Here again, J^p represents a symbolic computation that can be instantiated with an input $J^p(\xi)$). Note that J^p can be defined recursively:

$$J^p = (\frac{\partial p^2}{\partial p^1})^\top (\frac{\partial p^3}{\partial p^2})^\top J^{p:3} \tag{76}$$

[27]Here we are talking about equality of symbols rather than equality of values of those symbols.

where $J_{p:k}, k \leq |p|$ is defined as:

$$J^{p:k} \overset{\text{def}}{=} \begin{cases} (\frac{\partial p^{k+1}}{\partial p^k})^\top (\frac{\partial p^{-2}}{\partial p^{-3}})^\top ... (\frac{\partial p^{-1}}{\partial p^{-2}})^\top v & k < |p| \\ v & k = |p| \end{cases} \tag{77}$$

Recall that each path p starts with an X-var p^0, and alternates between G and X vars. Let W^{p^3} denote the defining weight matrix of G-var p^3 (i.e $p^3 = W^{p^3} p^2$), and let $p^2 = \psi(..., p^1, ...)$. Then we can re-write Eq. (76) as:

$$J^p = \psi'(..., p^1, ...) \odot J^{p:2}, \quad J^{p:2} = (W^{p^3})^\top J^{p:3} \tag{78}$$

Note that Eq. (78) can be written in NETSOR$\top$ language using MATMUL instructions using the transposed weights, and NONLIN instructions using ψ', which is pseudo-Lipschitz by Setup D.1.

Recall that π_0 is the program defining the network architecture. We now write the unrolled SGD of this network in a new program π. Below, recall that lack of time subscript means $t = 0$ (e.g. W means W_0, the initialized value). In addition, feel free to revisit the notations explained before Appendix A.

- If $g = Wh \in \pi_0$, then:

$$\delta\tilde{g}_{t+1} = \sqrt{n}(W_{t+1}\tilde{h}_{t+1} - W_t\tilde{h}_t) = W\delta\tilde{h}_{t+1} + \sqrt{n}(W_{t+1} - W_t)\tilde{h}_t + \sum_{s=0}^{t}(W_{s+1} - W_s)\delta\tilde{h}_{t+1}. \tag{79}$$

where, using Eq. (73), we have

$$\sqrt{n}(W_{t+1} - W_t)\tilde{h}_t = -\chi_t \sum_{\mathtt{g},\mathtt{h}:\mathtt{g}=W\mathtt{h}} d\mathtt{g}_t \frac{\mathtt{h}_t^\top \tilde{h}_t}{n} \tag{80}$$

$$\sum_{s=0}^{t}(W_{s+1} - W_s)\delta\tilde{h}_{t+1} = -\sum_{s=0}^{t} \frac{\chi_s}{\sqrt{n}} \sum_{\mathtt{g},\mathtt{h}:\mathtt{g}=W\mathtt{h}} d\mathtt{g}_s \frac{\mathtt{h}_s^\top \delta\tilde{h}_t}{n} \tag{81}$$

Tensor Program implementation Eqs. (79) to (81) may be easily implemented using NETSOR$\top^+$ instructions. For instance, Eq. (80) (assuming the sum sums over a single pair $\{\mathtt{h}, \mathtt{g}\}$) may be implemented using MOMENT and NONLIN$^+$ instructions as follows: the term $\frac{\mathtt{h}_t^\top \tilde{h}_t}{n}$ may be implemented by a MOMENT instruction with $\psi(\tilde{h}_t, \mathtt{h}_t) = \frac{1}{n}\sum_\alpha (\tilde{h}_t)_\alpha (\mathtt{h}_t)_\alpha$. The full term is then a NONLIN$^+$instructions $\psi(d\mathtt{g}_t; \chi_t, \{\frac{\mathtt{h}_t^\top \tilde{h}_t}{n}\}_\mathtt{h})$ with scalars $\chi_t, \{\frac{\mathtt{h}_t^\top \tilde{h}_t}{n}\}_\mathtt{h}$ and vector $d\mathtt{g}_t$.

- If $g = \psi(h^1, ..., h^k) \in \pi_0$, then

$$\delta\tilde{g}_{t+1} = \sqrt{n}\left(\psi(\tilde{h}_t^1 + \frac{\delta\tilde{h}_{t+1}^1}{\sqrt{n}}, ..., \tilde{h}_t^k + \frac{\delta\tilde{h}_{t+1}^k}{\sqrt{n}}) - \psi(\tilde{h}_t^1, ..., \tilde{h}_t^k)\right). \tag{82}$$

Tensor Program implementation Eq. (82) may be implemented as a NONLIN$^+$ instruction:

$$\delta\tilde{g}_{t+1} := \psi^\star\left(\{\tilde{h}_t^i\}_{i=1}^k \cup \{\delta\tilde{h}_{t+1}^i\}_{i=1}^k; \frac{1}{\sqrt{n}}\right) \tag{83}$$

for a set of vectors $\{\tilde{h}_t^i\}_{i=1}^k, \{\delta\tilde{h}_{t+1}^i\}_{i=1}^k$ and a scalar $\frac{1}{\sqrt{n}}$, where:

$$\psi^\star(\{\mu^i\}_{i=1}^k \cup \{\nu^i\}_{i=1}^k; \theta)_\alpha \overset{\text{def}}{=} \begin{cases} \frac{\psi(\mu^1+\theta\nu^1,...,\mu^k+\theta\nu^k)_\alpha - \psi(\mu^1,...,\mu^k)_\alpha}{\theta} & \theta > 0 \\ \sum_{i=1}^k \frac{\partial\psi(\mu^1,...,\mu^k)_\alpha}{\partial\mu_\alpha^i}\nu_\alpha^i & \theta = 0. \end{cases} \tag{84}$$

Since ψ' is pseudo-Lipschitz by Setup D.1, $\psi^\star$ is pseudo-Lipschitz in all of its inputs as well.

- WLOG assume $f(\xi) = \frac{v^\top x(\xi)}{\sqrt{n}}$, then:

$$\tilde{f}_{t+1} = \frac{v^\top \delta\tilde{x}_{t+1}}{n}, \quad \chi_t = \nabla_{f_t}\mathcal{L}_t. \tag{85}$$

Tensor Program implementation the scalar type outputs $f_t(\xi)$ at $t > 0$ for any input ξ can be implemented using the MOMENT instruction. The loss derivative $\chi_t, t > 0$ given f_t can be implemented with MOMENT instructions using $\psi(-; f(\xi)) = \mathcal{L}'\big(f(\xi)\big)$ where $f(\xi)$ is treated as a scalar type as in the first forward pass.

- If $g \in \mathbb{R}^n \in \pi_0$:

$$g_{t+1}(\xi) = g(\xi) + \frac{1}{\sqrt{n}} \sum_{s=1}^{t+1} \delta g_s(\xi) \tag{86}$$

Tensor Program implementation $\tilde{g}_{t+1}$ is implemented using a NONLIN$^+$ instruction $g_{t+1}(\xi) = \psi\big(g(\xi) \cup \{\delta g_s(\xi)\}_{s=1}^{t+1}; \frac{1}{\sqrt{n}}\big)$ with $\psi(\mu \cup \{\nu_s\}_{s=1}^{t+1}; \theta) = \mu + \theta \sum_s \nu_s$.

- If $g = Wh \in \pi_0$, then using Eq. (74):

$$dg(\xi)_{t+1} = \sum_{p:p^{-1}=x, p^1=g} J^p_{t+1}(\xi) \tag{87}$$

$$J^p_{t+1} = \psi'(..., p^1_{t+1}, ...) \odot \big((W^{p^3}_{t+1})^\top J^{p:3}_{t+1}\big) \tag{88}$$

Using Eq. (76), J^p_{t+1} is implemented recursively starting from $J^{p:-2}_{t+1} = (W^{p^{-1}}_{t+1})^\top v$. Plugging in the weights update at time $t+1$ (Appendix D.1.2):

$$J^{p:-2}_{t+1} = (W^{p^{-1}}_t + W^{p^{-1}}_{t+1} - W^{p^{-1}}_t)^\top v = J^{p:-2}_t - \chi_t \sum_{\mathtt{g},\mathtt{h}:\mathtt{g}=W^{p^{-1}}_{t+1}\mathtt{h}} \frac{\mathtt{h}_t}{\sqrt{n}} \frac{d\mathtt{g}_t^\top v}{n} \tag{89}$$

$$J^p_{t+1} = \psi'(..., p^1_t + \frac{\delta p^1_{t+1}}{\sqrt{n}}, ...) \odot J^{p:2}_{t+1} \tag{90}$$

Tensor Program implementation $dg(\xi)_{t+1}$ is implemented using MOMENT and NONLIN$^+$ instructions.

For further illustration, we present in Algorithm 1 a program implementation of the first iteration of SGD on the two hidden layer MLP specified in Appendix A.2, but where only the middle layer w is trained; in Fig. 7 we have its graphical form. Naturally, the following SGD steps can be implemented in a similar fashion.

Algorithm 1 NETSOR$\top^+$ program π_1 implementing the first update $\tilde{f}_1 - \tilde{f}$. Note that in this example, ψ represents multiple functions, while ϕ is a fixed function representing the non-linearity of the MLP in Appendix A.2. Technically, we should have $\hat{v}$ as defined in Appendix D.1.1 instead of v, but as explained there, this does not affect the limit.

```
**Input:** $\mathcal{W} = \{w\}, \mathcal{V} = \{v, g = g_0(\xi_0), \tilde{g} = g_0(\tilde{\xi})\}, \mathcal{C} = \{f = f_0(\xi_0), \tilde{f} = f_0(\tilde{\xi})\}$
MOMENT: $\chi := \mathcal{L}'(f)$
NONLIN$^+$: $z := \phi(g)$
NONLIN$^+$: $\tilde{z} := \phi(\tilde{g})$
MATMUL: $h := Wg$
MATMUL: $\tilde{h} := W\tilde{g}$
NONLIN$^+$: $\tilde{z} := \phi(\tilde{g})$
MOMENT: $\theta := \psi(z, \tilde{z}) = \frac{z^\top \tilde{z}}{n}$
NONLIN: $dh := \psi(v, h) = \phi'(h) \odot v$
NONLIN$^+$: $\delta\tilde{h}_1 := \psi(dh; \theta, \chi) = -\chi\theta dh$
NONLIN$^+$: $\delta\tilde{x} := \psi(\tilde{h}, \delta\tilde{h}_1; \frac{1}{\sqrt{n}}) = \sqrt{n}\big(\phi(\tilde{h} + \frac{\delta\tilde{h}_1}{\sqrt{n}}) - \phi(\tilde{h})\big)$
MOMENT: $\tilde{f}_1 - \tilde{f} = \frac{v^\top \delta\tilde{x}}{n}$
**Output:** $\tilde{f}_1 - \tilde{f}$
```

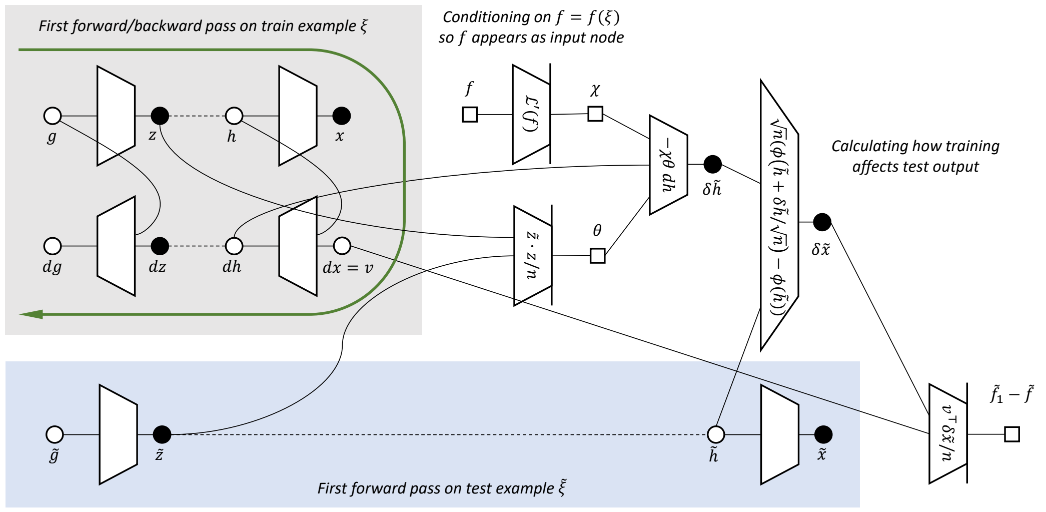
First forward/backward pass on train example ξ
Conditioning on $f = f(\xi)$
so f appears as input node
Calculating how training
affects test output
First forward pass on test example $\tilde{\xi}$
g
z
h
x
dg
dz
dh
$dx = v$
f
$\mathcal{L}'(f)$
χ
θ
$z \cdot z/n$
$-\chi\theta\, dh$
$\delta\tilde{h}$
$\sqrt{n}(\phi(\tilde{h} + \delta\tilde{h}/\sqrt{n}) - \phi(\tilde{h}))$
$\delta\tilde{x}$
$v^\top \delta\tilde{x}/n$
$\tilde{f}_1 - \tilde{f}$
$\tilde{g}$
$\tilde{z}$
$\tilde{h}$
$\tilde{x}$

Figure 7: Graphical form of NETSOR$\top^+$ program (Algorithm 1) encoding the first training step and the effect on the test set on the two hidden layer MLP specified in Appendix A.2, but where only the middle layer w is trained. Technically, we should have $\hat{v}$ as defined in Appendix D.1.1 instead of v, but as explained there, this does not affect the limit.

SGD in the Infinite Width Limit According to the NETSOR$\top^+$ rules as specified in Definition B.3, we have the following identities:

- If $g = Wh$, then using Eqs. (79) to (81): (Here $Z^{dg_t} = Z^{dg_t(\xi_t)}$, $Z^{h_t} = Z^{h_t(\xi_t)}$, and $Z^{\tilde{h}} = Z^{h_t(\tilde{\xi})}$)

$$Z^{\delta\tilde{g}_{t+1}} = Z^{W\delta\tilde{h}_{t+1}} - \chi_t \sum_{g,h:g=Wh} Z^{dg_t}\,\mathbb{E}\left[Z^{h_t} Z^{\tilde{h}}\right] \tag{91}$$

$$\text{where} \quad Z^{W\delta\tilde{h}_{t+1}} = \hat{Z}^{W\delta\tilde{h}_{t+1}} + \sum_{y} Z^{\tilde{y}}\,\mathbb{E}\,\frac{\partial Z^{\delta\tilde{h}_{t+1}}}{\partial \hat{Z}^{W^\top \tilde{y}}}. \tag{92}$$

- If $g = \psi(h^1, ..., h^k)$, then using Eqs. (82) and (84), taking the limit $1/\sqrt{n} \to 0$,

$$Z^{\delta\tilde{g}_{t+1}} = \sum_{i=1}^{k} \frac{\partial\psi(Z^{\tilde{h}^1}, ..., Z^{\tilde{h}^k})}{\partial Z^{\tilde{h}^i}} Z^{\delta\tilde{h}^i_{t+1}}. \tag{93}$$

- Using Eqs. (86), (87), (89) and (90) and taking $\frac{1}{\sqrt{n}} \to 0$, we have by ZNONLIN$^+$:

$$Z^{g_t(\xi)} = Z^{g_0(\xi)}, \quad Z^{dg_t(\xi)} = Z^{dg_0(\xi)} \quad \text{for any vector } g \in \pi_0 \text{ at time } t. \tag{94}$$

D.2. Deriving The NTK

Instantiate paths p and q on two inputs ξ, ξ' by $p = p(\xi), q = q(\xi')$ (abusing notation slightly). We define an inner product between them as follows:

$$\langle p, q\rangle \overset{\text{def}}{=} \mathbb{E}\left[Z^{p^0} Z^{q^0}\right] \prod_{i=2,even}^{|p|-2} \mathbb{E}\left[\frac{\partial Z^{p^i}}{\partial Z^{p^{i-1}}} \frac{\partial Z^{q^i}}{\partial Z^{q^{i-1}}}\right]. \tag{95}$$

where p^i, q^i are X-vars for all even i. Note that for even i, p^i is always of the form $p^i = \psi(...., p^{i-1}, ...)$ for some ψ. So the partial derivatives in Eq. (95) are just $\frac{\partial Z^{p^i}}{\partial Z^{p^{i-1}(\xi)}} = \psi'(..., Z^{p^{i-1}}, ...)$.

Our goal in this section is to prove

Proposition D.3.

$$\mathring{\mathcal{K}}(\xi, \tilde{\xi}) = \sum_{p,q: p^{-1}=q^{-1}=x, p \cong q} \langle p(\xi), q(\tilde{\xi}) \rangle. \tag{96}$$

For each weight $W \in \mathcal{W}$, the gradient of the output with respect to w is given by:

$$\nabla_w f(\xi) = \sum_{g,h: g=Wh} \frac{dg(\xi)\ h(\xi)^\top}{n} \tag{97}$$

Here, g, h represent nodes in program π_0 that can be instantiated by an input ξ.
The NTK of f can be expressed as:

$$\mathring{\mathcal{K}}(\xi, \tilde{\xi}) = \lim_{n\to\infty} \sum_{W \in \mathcal{W}} \left\langle \nabla_w f(\xi), \nabla_w f(\tilde{\xi}) \right\rangle \tag{98}$$

$$= \lim_{n\to\infty} \sum_{W \in \mathcal{W}} \sum_{g,h: g=Wh} \sum_{\mathtt{g},\mathtt{h}: \mathtt{g}=W\mathtt{h}} \frac{dg(\xi)^\top d\mathtt{g}(\tilde{\xi})}{n} \frac{h(\xi)^\top \mathtt{h}(\tilde{\xi})}{n} \tag{99}$$

$$= \sum_{W \in \mathcal{W}} \sum_{g,h: g=Wh} \sum_{\mathtt{g},\mathtt{h}: \mathtt{g}=W\mathtt{h}} \mathbb{E}\left[Z^{dg(\xi)} Z^{d\mathtt{g}(\tilde{\xi})}\right] \mathbb{E}\left[Z^{h(\xi)} Z^{\mathtt{h}(\tilde{\xi})}\right]. \tag{100}$$

Using Eqs. (74) and (78), for any G-var $g = Wh$, we can write the error term dg as the summation of errors signals over paths p:

$$dg(\xi) = \sum_{p: p^{-1}=x, p^1=g} J^p(\xi), \qquad J^p = \psi'(..., p^1, ...) \odot \left((W^{p^3})^\top J^{p:3}\right) \tag{101}$$

Hence we can write:

$$\mathbb{E}\left[Z^{dg(\xi)} Z^{dg(\tilde{\xi})}\right] = \sum_{p^1=g, p^{-1}=x, q^1=g, q^{-1}=x} \mathbb{E}\left[Z^{J^p(\xi)} Z^{J^q(\tilde{\xi})}\right] \tag{102}$$

$$Z^{J^p} = Z^{(W^{p^3})^\top J^{p:3}} \frac{\partial Z^{p^2}}{\partial Z^{p^1}} \tag{103}$$

where ψ' denotes the derivative w.r.t. p^1. By Simple GIA Check (Yang, 2020a), we have that $Z^{(W^{p^3})^\top J^{p:3}} = \hat{Z}^{(W^{p^3})^\top J^{p:3}}$ (see ZMatMul). Hence, with abuse of notation $J^p = J^p(\xi), J^q = J^q(\tilde{\xi}), p = p(\xi), q = q(\tilde{\xi})$, we have

$$\mathbb{E}\left[Z^{J^p} Z^{J^q}\right] = \mathbb{E}[\hat{Z}^{(W^{p^3})^\top J^{p:3}} \hat{Z}^{(W^{q^3})^\top J^{q:3}}] \mathbb{E}[\frac{\partial Z^{p^2}}{\partial Z^{p^1}} \frac{\partial Z^{q^2}}{\partial Z^{q^1}}]. \tag{104}$$

From the definition of ZHat, the expectation $\mathbb{E}[\hat{Z}^{(W^{p^3})^\top J^{p:3}} \hat{Z}^{(W^{q^3})^\top J^{q:3}}]$ vanishes if the weights W^{p^3} and W^{q^3} are not symbolically the same (i.e $W^{p^3} \triangleq W^{q^3}$). Then by ZHat,

$$\mathbb{E}[\hat{Z}^{(W^{p^3})^\top J^{p:3}} \hat{Z}`^{(W^{q^3})^\top J^{q:3}}] = \begin{cases} \mathbb{E}[Z^{J^{p:3}} Z^{J^{q:3}}] & \text{if } W^{p^3} \triangleq W^{q^3} \\ 0 & \text{otherwise.} \end{cases} \tag{105}$$

Applying this logic recursively, we have

$$\mathbb{E}\left[Z^{J^p} Z^{J^q}\right] = \begin{cases} \prod_{i=2,even}^{|p|-2} \mathbb{E}\left[\frac{\partial Z^{p^i}}{\partial Z^{p^{i-1}}} \frac{\partial Z^{q^i}}{\partial Z^{q^{i-1}}}\right] & \text{if } p \cong q \\ 0 & \text{otherwise.} \end{cases} \tag{106}$$

Combining with Eqs. (74), (100) and (102) proves Proposition D.3.

D.3. Getting Claim 3.1

Notation For the remainder of the proof we abbreviate $p = p(\tilde{\xi}), q = q(\xi_t), p^i = p^i(\tilde{\xi}), q^i = q^i(\xi_t)$ (i.e path p is always evaluated on $\tilde{\xi}$, while path q is always evaluated on ξ_t).
We prove Claim 3.1 by inducting on all G-vars in the network. We begin by proving the following induction hypothesis.

Definition D.4. We write $Z^x \equiv Z^y \mod \hat{Z}^{W\bullet}$ to denote that $Z^x - Z^y$ is a linear combination of $\hat{Z}^{Wu}$ for various vectors u.

Induction Hypothesis. At any time t and G-var $g = Wh$, the following holds:

$$Z^{\delta\tilde{g}_{t+1}} \equiv -\chi_t \sum_{p:p^{-1}=g} \sum_{q:q\cong p} Z^{dq^{-1}} \langle p, q\rangle \mod \hat{Z}^{W\bullet} \tag{107}$$

Here, the sum is over all paths p with endpoint g and all paths q isomorphic to p. Recall that dq^{-1} is the (scaled) gradient dy where $y = q^{-1}$ is the endpoint of q.

D.3.1. Base Case

For initial G-vars g, $\delta\tilde{g}_t = 0$ since we are not training the input layers (Assumption A1.). This proves the base case since the sum in Eq. (107) has no terms and thus is 0.

D.3.2. Inductive case

Suppose $g = Wh$, where $h = \psi(h^1, ..., h^k)$, we then have using Eq. (91):

$$Z^{\delta\tilde{g}_{t+1}} \equiv \dot{Z}^{W\delta\tilde{h}_{t+1}} + \chi_t \sum_{\mathtt{g}=W\mathtt{h}} Z^{d\mathtt{g}_t}\, \mathbb{E}\left[Z^{\mathtt{h}_t} Z^{\tilde{h}}\right] \mod \hat{Z}^{W\bullet} \tag{108}$$

$$\text{where} \quad \dot{Z}^{W\delta\tilde{h}_{t+1}} = \sum_y Z^y\, \mathbb{E}\, \frac{\partial Z^{\delta\tilde{h}_{t+1}}}{\partial \hat{Z}^{W^\top y}}. \tag{109}$$

Note $\sum_{\mathtt{g}=W\mathtt{h}} Z^{d\mathtt{g}_t}\, \mathbb{E}\left[Z^{\mathtt{h}_t} Z^{\tilde{h}}\right]$ in Eq. (108) can be written as $\sum_{\mathtt{p}:\mathtt{p}^{-1}=g,|\mathtt{p}|=2} \sum_{\mathtt{q}\cong\mathtt{p}} Z^{d\mathtt{q}^{-1}}\langle \mathtt{p}, \mathtt{q}\rangle$. Therefore, it suffices to show that

$$\dot{Z}^{W\delta\tilde{h}_{t+1}} = -\chi_t \sum_{\mathtt{p}:\mathtt{p}^{-1}=g,|\mathtt{p}|\geq 4} \sum_{\mathtt{q}\cong\mathtt{p}} Z^{d\mathtt{q}^{-1}} \langle \mathtt{p}, \mathtt{q}\rangle. \tag{110}$$

Showing Eq. (110) By Eq. (82):

$$Z^{\delta\tilde{h}_{t+1}} = \sum_{i=1}^k \frac{\partial Z^{\tilde{h}}}{\partial Z^{\tilde{h}^i}} Z^{\delta\tilde{h}^i_{t+1}}. \tag{111}$$

Since $Z^{\tilde{h}^1}, ..., Z^{\tilde{h}^k}$ do not depend on $Z^{W^\top y}$ for any y (by the assumption that we don't use both a matrix and its transpose in the forward pass), from Eq. (111) we have for any y:

$$\mathbb{E}\left[\frac{\partial Z^{\delta\tilde{h}_{t+1}}}{\partial \hat{Z}^{W^\top y}}\right] = \sum_{i=1}^k \mathbb{E}\left[\frac{\partial Z^{\tilde{h}}}{\partial Z^{\tilde{h}^i}} \frac{\partial Z^{\delta\tilde{h}^i_{t+1}}}{\partial \hat{Z}^{W^\top y}}\right]. \tag{112}$$

Applying the induction hypothesis Eq. (107) to each G-var h^i, we get

$$\frac{\partial Z^{\delta\tilde{h}^i_{t+1}}}{\partial \hat{Z}^{W^\top y}} = -\chi_t \sum_{p:p^{-1}=h^i} \sum_{q:q\cong p} \frac{\partial Z^{dq^{-1}}}{\partial \hat{Z}^{W^\top y}} \langle p, q\rangle \mod \hat{Z}^{W\bullet} \tag{113}$$

Plugging this back into $\dot{Z}^{W\delta\tilde{h}_{t+1}}$ (Eq. (109)), we get

$$\dot{Z}^{W\delta\tilde{h}_{t+1}} = -\chi_t \sum_y Z^y \sum_{i=1}^k \sum_{p:p^{-1}=h^i} \sum_{q:q\cong p} \langle p, q\rangle\, \mathbb{E}\left[\frac{\partial Z^{\tilde{h}}}{\partial Z^{\tilde{h}^i}} \frac{\partial Z^{dq^{-1}}}{\partial \hat{Z}^{W^\top y}}\right]. \tag{114}$$

Note that for any path p with $p^{-1} = h^i$, we may extend p by vectors g, h (recall $g = Wh$ and $h = \psi(h^1, ..., h^k)$). Let $\mathtt{p}$ denote this extension. If $\mathtt{q}$ is a path such that

$$\mathtt{q} \cong \mathtt{p} \quad \text{and} \quad \frac{\partial Z^{dq^{-1}}}{\partial \hat{Z}^{W^\top y}} = \frac{\partial Z^{\mathtt{q}^{-2}}}{\partial Z^{\mathtt{q}^{-3}}}, \tag{115}$$

then

$$\langle p, q\rangle \, \mathbb{E}\, \big[\frac{\partial Z^{\tilde{h}}}{\partial Z^{\tilde{h}^i}} \frac{\partial Z^{dq^{-1}}}{\partial \hat{Z}^{W^\top y}}\big] = \langle \mathtt{p}, \mathtt{q}\rangle. \tag{116}$$

Our goal now is to show q in Eq. (114) can be extended appropriately such that we may rewrite Eq. (114) as Eq. (110). This will be done through explicitly computing the term $\frac{\partial Z^{dq^{-1}}}{\partial \hat{Z}^{W^\top y}}$ in Eq. (114).

Computing $\frac{\partial Z^{dq^{-1}}}{\partial \hat{Z}^{W^\top y}}$ Suppose $\{g^1, ..., g^r\}$ are all G-vars in the program π_0 that depend on q^{-1} i.e for all $1 \leq j \leq r$, we have $g^j = W^j z^j$ where $z^j = \psi_j(..., q^{-1}, ...)$ and where W^j can be same or different matrices for different j. Note that it follows that:

$$dq^{-1} = \sum_{j=1}^{r} \psi_j'(..., q^{-1}, ...) \odot ((W^j)^\top dg^j) \tag{117}$$

$$Z^{dq^{-1}} = \sum_{j=1}^{r} \frac{\partial Z^{z^j}}{\partial Z^{q^{-1}}} Z^{(W^j)^\top dg^j} \tag{118}$$

$$\text{where} \quad Z^{(W^j)^\top dg^j} = \hat{Z}^{(W^j)^\top dg^j} + \dot{Z}^{(W^j)^\top dg^j} = \hat{Z}^{(W^j)^\top dg^j}. \tag{119}$$

Note that $\dot{Z}^{(W^j)^\top dg^j} = 0$ in Eq. (119) from the gradient independence assumption (GIA) because we pass the Simple GIA Check. This may also be easily verified by explicitly computing $\dot{Z}^{(W^j)^\top dg^j} = \sum_y Z^y \, \mathbb{E}\, \frac{\partial Z^{dg^j}}{\partial \hat{Z}^{W y}}$, and noticing that the expectation vanishes from the dependency of Z^{dg^j} on Z^v (i.e $Z^{dg^j} = Z^v Z^\mu$ for some vector μ which does not depend on v). Since y does not depend on v and the last layer v is not trained, we have $\mathbb{E}\, \frac{\partial Z^{dg^j}}{\partial \hat{Z}^{W y}} = \mathbb{E}[Z^v]\, \mathbb{E}[..] = 0$.
Since we assumed that the forward propagation does not contain both $W, W^\top$, it follows from differentiating Eq. (118) that

$$\frac{\partial Z^{dq^{-1}}}{\partial \hat{Z}^{W^\top y}} = \sum_{j=1}^{r} \frac{\partial Z^{z^j}}{\partial Z^{q^{-1}}} \frac{\partial \hat{Z}^{(W^j)^\top dg^j}}{\partial \hat{Z}^{W^\top y}} = \sum_{j : W^j \triangleq W, dg^j = y} \frac{\partial Z^{z^j}}{\partial Z^{q^{-1}}}. \tag{120}$$

If this sum over j is nonempty, then there is a unique j such that $W^j \triangleq W$ and $dg^j = y$. In such a case, we may extend the path q with g^j, z^j to form $\mathtt{q}$ satisfying Eq. (115). Plugging back into Eq. (114) we obtain Eq. (110) as desired.

Hence, we have proven the induction hypothesis.

D.3.3. Proving Claim 3.1 using the induction hypothesis

WLOG assume $f(\xi) = V^\top x(\xi)$ for some G-var $x(\xi)$. Using the induction hypothesis and the Master Theorem (Theorem B.4), we have that:

$$\lim_{n \to \infty} \tilde{f}_{t+1} - \tilde{f}_t = \mathbb{E}\, Z^v Z^{\delta \tilde{x}_{t+1}} = -\chi_t \sum_{p : p^{-1} = x} \sum_{q \cong q} \mathbb{E}\, \big[Z^v Z^{dq^{-1}}\big] \langle p, q\rangle. \tag{121}$$

Note that $Z^{dq^{-1}} = Z^v$ for any path $q : q^{-1} = x$. Hence, with Eq. (96), we have

$$\lim_{n \to \infty} \tilde{f}_{t+1} - \tilde{f}_t = -\chi_t \sum_{p : p^{-1} = x} \sum_{q \cong p} \langle p, q\rangle = -\chi_t \mathring{\mathcal{K}}(\xi_t, \tilde{\xi}) \tag{122}$$

as desired.

D.4. Relaxing Assumptions (A1.) to (A4.)

We now briefly discuss the case where Assumptions (A1.) to (A4.) are relaxed, as well as the case where f is represented by a NETSOR$\top^+$ program. As the proof of the general case follows roughly the same logic as in Setup D.1, we only discuss the meaningful differences in each case.

D.4.1. TRAINING THE FIRST AND LAST LAYERS

Recall the input and output layers are parameterized by $\{u^i\}, v$ which now depend on t. The output evolution is now given by:

$$\tilde{f}_{t+1} - \tilde{f}_t = V_{t+1}^\top \tilde{x}_{t+1} - V_t^\top \tilde{x}_t = \frac{v^\top \delta \tilde{x}_{t+1}}{n} + \frac{\sqrt{n}(v_{t+1} - v_t)^\top \tilde{x}_t}{n} + \sum_{s=0}^{t} \frac{(v_{s+1} - v_s)^\top \delta \tilde{x}_{t+1}}{n}. \tag{123}$$

Plugging $v_{t+1} - v_t = -\chi_t \frac{1}{\sqrt{n}} x_t$ into Eq. (123) and taking the limit (using NETSOR$\top^+$ rules):

$$\lim_{n\to\infty} \tilde{f}_{t+1} - \tilde{f}_t = \mathbb{E}\left[Z^v Z^{\delta \tilde{x}_{t+1}}\right] - \chi_t \mathbb{E}\left[Z^{x(\xi_t)} Z^{\tilde{x}}\right]. \tag{124}$$

Evaluating $\mathbb{E}\left[Z^v Z^{\delta \tilde{x}_{t+1}}\right]$ by induction requires altering the path definition so that each path p may start with an input ξ, and ends with a G-var (that is, a path either starts with an X-var or an input). We reuse the definition of inner product between $p \cong q$ in Eq. (95), only when both start with inputs $\xi, \tilde{\xi}$ respectively then $\mathbb{E}\left[Z^{p^0} Z^{q^0}\right]$ implies $\xi^\top \tilde{\xi}$. The remainder of the proof follows the same logic as with Setup D.1. Note that the NTK in this case would yield:

$$\mathring{\mathcal{K}}(\xi_t, \tilde{\xi}) = \sum_{q \cong q} \langle p, q\rangle + \mathbb{E}\left[Z^{x(\xi_t)} Z^{\tilde{x}}\right]. \tag{125}$$

D.4.2. $W, W^\top$ IN THE FORWARD PASS

When both $W, W^\top$ are allowed in the forward pass, the update equations for each w_t take the form:

$$w_{t+1} - w_t = -\chi_t \sum_{g,h:g=Wh} \frac{dg_t h_t^\top}{n} - \chi_t \sum_{g,h:g=W^\top h} \frac{h_t dg_t^\top}{n} \tag{126}$$

Some quick calculations using NETSOR$\top^+$rules show that for G-vars:

- If $g = Wh$:

$$Z^{\delta \tilde{g}_{t+1}} = Z^{W \delta \tilde{h}_{t+1}} - \chi_t \sum_{\mathrm{g,h:g}=W\mathrm{h}} Z^{\mathrm{dg}(\xi_t)} \mathbb{E}\left[Z^{\mathrm{h}(\xi_t)} Z^{\tilde{h}}\right]) - \chi_t \sum_{\mathrm{g,h:g}=W^\top\mathrm{h}} Z^{\mathrm{h}(\xi_t)} \mathbb{E}\left[Z^{\mathrm{dg}(\xi_t)} Z^{\tilde{h}}\right]. \tag{127}$$

- If $g = W^\top h$:

$$Z^{\delta \tilde{g}_{t+1}} = Z^{W^\top \delta \tilde{h}_{t+1}} - \chi_t \sum_{\mathrm{g,h:g}=W^\top\mathrm{h}} Z^{\mathrm{dg}(\xi_t)} \mathbb{E}\left[Z^{\mathrm{h}(\xi_t)} Z^{\tilde{h}}\right]) - \chi_t \sum_{\mathrm{g,h:g}=W\mathrm{h}} Z^{\mathrm{h}(\xi_t)} \mathbb{E}\left[Z^{\mathrm{dg}(\xi_t)} Z^{\tilde{h}}\right]. \tag{128}$$

It is straightforward to show using GIA (Yang, 2020a) that $\mathbb{E}\left[Z^{\mathrm{dg}(\xi_t)} Z^{\tilde{h}}\right] = 0$ in both cases, leaving us with a similar expression as with Setup D.1. The induction hypothesis for G-vars in this case takes one of two forms:

- If $g = Wh$ then Eq. (107) holds.
- If $g = W^\top h$ then Eq. (107) holds with $\mod \hat{Z}^{W^\top \bullet}$ replacing $\mod \hat{Z}^{W \bullet}$.

Some additional complications need to be resolved. Specifically, with setup Setup D.1 we have used in two places the fact that no transpose is used in the forward pass to prove the induction hypothesis (see Eqs. (112) and (120)). To prove the induction, and assuming $g = Wh$, we now have instead of Eq. (112) (using Eq. (111)):

$$\mathbb{E}\frac{\partial Z^{\delta \tilde{h}_{t+1}}}{\partial \hat{Z}^{W^\top y}} = \sum_{i=1}^{k} \mathbb{E}\left[\frac{\partial Z^{\tilde{h}}}{\partial Z^{\tilde{h}^i}} \frac{\partial Z^{\delta \tilde{h}^i_{t+1}}}{\partial \hat{Z}^{W^\top y}}\right] + \sum_{i=1}^{k} \mathbb{E}\left[\frac{\partial^2 Z^{\tilde{h}}}{\partial Z^{\tilde{h}^i} \partial \hat{Z}^{W^\top y}} Z^{\delta \tilde{h}^i_{t+1}}\right]. \tag{129}$$

where $\{h^i\}$ are G-vars. To evaluate the additional term on the RHS of Eq. (129), we use the induction hypothesis to express $Z^{\delta\tilde{h}^i_{t+1}}$:

$$\mathbb{E}\left[\frac{\partial^2 Z^{\tilde{h}}}{\partial Z^{\tilde{h}^i}\partial \hat{Z}^{W^\top y}} Z^{\delta\tilde{h}^i_{t+1}}\right] = -\chi_t \sum_{p:p^{-1}=h^i} \sum_{q\cong p} \langle p, q\rangle \, \mathbb{E}\left[\frac{\partial^2 Z^{\tilde{h}}}{\partial Z^{\tilde{h}^i}\partial \hat{Z}^{W^\top y}} Z^{dq^{-1}}\right]. \tag{130}$$

Using GIA (Yang, 2020a), it is straight forward to show that the expectation on the RHS of Eq. (130) vanishes, leaving us with the first term on the RHS of Eq. (129), as with Setup D.1. Note that the same logic may be applied in Eq. (120), concluding the proof.

D.4.3. Multiple outputs and arbitrary batchsize

We have used a scalar output and a batchsize of 1 throughout this paper. However, extending to multiple (finite) outputs and an arbitrary batchsize requires no additional arguments besides some additional notations. For example, the definition of path should now be altered to express dependency on multiple samples (if batchnorm is used for example). The proof however follows roughly the same logic in Setup D.1.

D.4.4. X-var embedding

We assumed in our proof that x, which represents the final embedding of f is a G-var. However, extending the proof to the case where x is an X-var is straightforward. Let $f(\xi) = V^\top x(\xi)$ where $x = \psi(h^1, ..., h^k)$ and $h^1, ..., h^k$ are G-vars. Using the induction hypothesis, along with Eq. (93) yields:

$$\lim_{n\to\infty} \tilde{f}_{t+1} - \tilde{f}_t = -\chi_t \, \mathbb{E}\left[Z^v Z^{\delta\tilde{x}_{t+1}}\right] = -\chi_t \sum_{i=1}^{k} \mathbb{E}\left[Z^v \frac{\partial Z^{\tilde{x}}}{\partial Z^{\tilde{h}^i}} Z^{\delta\tilde{h}_{t+1}}\right] \tag{131}$$

$$= -\chi_t \sum_{i=1}^{k} \mathbb{E}\left[Z^v \frac{\partial Z^{\tilde{x}}}{\partial Z^{\tilde{h}^i}} \sum_{p:p^{-1}=h^i} \sum_{q\cong p} Z^{dh^i(\xi_t)} \langle p, q\rangle\right] \tag{132}$$

$$= -\chi_t \sum_{i=1}^{k} \sum_{p:p^{-1}=h^i} \sum_{q\cong p} \langle p, q\rangle \, \mathbb{E}\left[\frac{\partial Z^{\tilde{x}}}{\partial Z^{\tilde{h}^i}} \frac{\partial Z^{x(\xi_t)}}{\partial Z^{h^i(\xi_t)}}\right] \tag{133}$$

$$= -\chi_t \mathring{\mathcal{K}}(\xi_t, \tilde{\xi}) \tag{134}$$

It is straightforward to show that the expression for $\mathring{\mathcal{K}}(\xi_t, \tilde{\xi})$ in Eq. (133) represents the NTK if this case.

D.4.5. Network specified by Netsor$\top^+$

If the network is more generally represented by a Netsor$\top^+$ program instead of just a Netsor$\top$ program, then our proof can be very simply modified to accommodate as follows: The new operation allowed in such a network is the production of a scalar through Moment, say $a = \frac{1}{n}\sum_{\alpha=1}^{n} \psi(x^1_\alpha, \ldots, x^k_\alpha; \theta^1, \ldots, \theta^l)$. By a similar inductive argument as before, we will see that 1) $x^i_t = x^i_0 + o(1)$ for all $i \in [k]$ and $\theta^j_t = \theta^j_0 + o(1)$ for all $j \in [l]$, so that $a_t = a_0 + o(1)$; 2) in the backward pass, any backpropagation through a will zero out: For example, if a is only used later in a Nonlin $z = \psi(y; a)$, then $\frac{1}{\sqrt{n}}\nabla_a f = \langle dz, \partial_a \psi(y; a)\rangle/n$ will converge to 0 because of GIA (as dz is linear in the final layer), and the error signal at x^i times $\sqrt{n}$ is the constant vector with entries $\frac{1}{\sqrt{n}}\nabla_a f$, which is $o(1)$.

Therefore, we can treat any scalar produced through Moment as a constant fixed at initialization, and the notion of path from before carries over here without change (by assuming all nonlinearities with scalar parameters to be parameterless nonlinearities where the parameters are fixed). Then the same reasoning follows.