# Tensor programs

Part I' + Part II'

Eugene Golikov

April 9, 2021

École Polytechnique Fédérale de Lausanne, Switzerland
Former researcher at DeepPavlov.ai, Moscow Institute of Physics and Technology, Russia

**Define:**

$$\text{width of a network} = \text{the minimal number of nodes in its hidden representations.}$$

**Talk subject:** neural nets in the limit of infinite width.

**Reason:**

1. Sufficiently wide nets $\approx$ infinitely wide nets;
2. Infinitely wide nets are much easier to study theoretically;
3. Infinitely wide nets enjoy a number of cool properties (below).

Given He initialization (standard) and certain parameterization, a fully-connected feedforward network w/o shared weights enjoy the following properties:

1. It converges to a **Gaussian process** at initialization as width $\to \infty$ [Matthews et al., 2018];
2. Its GD training dynamics converges to a **kernel GD with a constant kernel** as width $\to \infty$ [Jacot et al., 2018];
3. The spectrum of its **input-output jacobian** can be computed in the limit of infinite width using a **free independence principle** [Pennington et al., 2017].

**Tensor programs** series [Yang, 2019, Yang, 2020a, Yang, 2020b] prove these properties for a wide class of models as follows:

1. Introduce a wide class of models called *tensor programs;*
2. Prove a *Master theorem* about their limit behavior;
3. Deduce the properties above from the Master theorem.

**Overall talk construction strategy:**

1. Take one of the properties discussed above;
2. Illustrate it on a simple model;
3. Introduce a class of tensor programs sufficient to express this property;
4. Prove the corresponding Master theorem;
5. Deduce the property from the Master theorem;
6. Proceed with another property.

# Convergence to Gaussian processes

Consider a forward pass:

$$\underbrace{h^l = W^l x^{l-1}}_{\text{pre-activations, } \in \mathbb{R}^{n_l}}, \qquad \underbrace{x^{l-1} = \phi(h^{l-1})}_{\text{activations, } \in \mathbb{R}^{n_{l-1}}}; \qquad W^l \sim \mathcal{N}\left(0, \frac{\sigma_W^2}{n_{l-1}}\right). \tag{1}$$

In a matrix form:

$$\begin{pmatrix} h_1^l \\ h_2^l \\ \dots \\ h_{n_l}^l \end{pmatrix} = \underbrace{\begin{pmatrix} W_{11}^l & W_{12}^l & \dots & W_{1n_{l-1}}^l \\ W_{21}^l & W_{22}^l & \dots & W_{2n_{l-1}}^l \\ \dots & \dots & \dots & \dots \\ W_{n_l1}^l & W_{n_l2}^l & \dots & W_{n_ln_{l-1}}^l \end{pmatrix}}_{\text{all iid } \sim \mathcal{N}(0, \sigma_W^2/n_{l-1})} \times \begin{pmatrix} x_1^{l-1} \\ x_2^{l-1} \\ \dots \\ x_{n_{l-1}}^{l-1} \end{pmatrix}.$$

$$
\underbrace{\begin{pmatrix} h_1^l \\ h_2^l \\ \cdots \\ h_{n_l}^l \end{pmatrix}}_{\substack{\text{components tend to Gaussians} \\ \text{as } n_{l-1} \to \infty \text{ by CLT}}}
=
\underbrace{\begin{pmatrix} W_{11}^l & W_{12}^l & \cdots & W_{1n_{l-1}}^l \\ W_{21}^l & W_{22}^l & \cdots & W_{2n_{l-1}}^l \\ \cdots & \cdots & \cdots & \cdots \\ W_{n_l1}^l & W_{n_l2}^l & \cdots & W_{n_ln_{l-1}}^l \end{pmatrix}}_{\text{all iid } \sim \mathcal{N}(0, \sigma_W^2/n_{l-1})}
\times
\underbrace{\begin{pmatrix} x_1^{l-1} \\ x_2^{l-1} \\ \cdots \\ x_{n_{l-1}}^{l-1} \end{pmatrix}}_{\text{suppose all components are iid}}
.
$$

$$
\underbrace{\begin{pmatrix} h_1^l \\ h_2^l \\ \cdots \\ h_{n_l}^l \end{pmatrix}}_{\text{different components are uncorrelated}} = \underbrace{\begin{pmatrix} W_{11}^l & W_{12}^l & \cdots & W_{1n_{l-1}}^l \\ W_{21}^l & W_{22}^l & \cdots & W_{2n_{l-1}}^l \\ \cdots & \cdots & \cdots & \cdots \\ W_{n_l1}^l & W_{n_l2}^l & \cdots & W_{n_ln_{l-1}}^l \end{pmatrix}}_{\text{rows are iid}} \times \underbrace{\begin{pmatrix} x_1^{l-1} \\ x_2^{l-1} \\ \cdots \\ x_{n_{l-1}}^{l-1} \end{pmatrix}}_{\text{suppose all components are iid}} .
$$

$$\underbrace{\begin{pmatrix} h_1^l \\ h_2^l \\ \dots \\ h_{n_l}^l \end{pmatrix}}_{\substack{\text{tends to a Gaussian vector} \\ \text{with iid components as } n_{l-1} \to \infty \text{ by CLT}}} = \underbrace{\begin{pmatrix} W_{11}^l & W_{12}^l & \dots & W_{1n_{l-1}}^l \\ W_{21}^l & W_{22}^l & \dots & W_{2n_{l-1}}^l \\ \dots & \dots & \dots & \dots \\ W_{n_l1}^l & W_{n_l2}^l & \dots & W_{n_ln_{l-1}}^l \end{pmatrix}}_{\text{all iid} \sim \mathcal{N}(0, \sigma_W^2/n_{l-1})} \times \underbrace{\begin{pmatrix} x_1^{l-1} \\ x_2^{l-1} \\ \dots \\ x_{n_{l-1}}^{l-1} \end{pmatrix}}_{\text{suppose all components are iid}} .$$

$$\underbrace{\begin{pmatrix} h_1^l & \bar{h}_1^l \\ h_2^l & \bar{h}_2^l \\ \cdots & \cdots \\ h_{n_l}^l & \bar{h}_{n_l}^l \end{pmatrix}}_{\substack{\text{each row converges to} \\ \text{a multivariate Gaussian vector} \\ \text{as } n_{l-1} \to \infty \text{ by the vector CLT}}} = \underbrace{\begin{pmatrix} W_{11}^l & W_{12}^l & \cdots & W_{1n_{l-1}}^l \\ W_{21}^l & W_{22}^l & \cdots & W_{2n_{l-1}}^l \\ \cdots & \cdots & \cdots & \cdots \\ W_{n_l 1}^l & W_{n_l 2}^l & \cdots & W_{n_l n_{l-1}}^l \end{pmatrix}}_{\text{all iid} \sim \mathcal{N}(0, \sigma_W^2 / n_{l-1})} \times \underbrace{\begin{pmatrix} x_1^{l-1} & \bar{x}_1^{l-1} \\ x_2^{l-1} & \bar{x}_2^{l-1} \\ \cdots & \cdots \\ x_{n_{l-1}}^{l-1} & \bar{x}_{n_{l-1}}^{l-1} \end{pmatrix}}_{\substack{\text{suppose all components} \\ \text{of each column are iid}}}.$$

Let $\xi_{1:M}$ be a batch of $M$ inputs.

$$\begin{pmatrix} h_1^l(\xi_1) & h_1^l(\xi_2) & \dots & h_1^l(\xi_M) \\ h_2^l(\xi_1) & h_2^l(\xi_2) & \dots & h_2^l(\xi_M) \\ \dots & \dots & \dots & \dots \\ h_{n_l}^l(\xi_1) & h_{n_l}^l(\xi_2) & \dots & h_{n_l}^l(\xi_M) \end{pmatrix}$$

- Batch dimension — tends to a multivariate zero-mean Gaussian as $n_{l-1} \to \infty$;
- Neuron dimension — components tend to iid Gaussians.

**Corollary (informal)**
*A neural net converges to a GP at initialization as $n_{1:L} \to \infty$ sequentially.*

**Off-topic remark: what happens during training?**

1. When quadratic loss is optimized with gradient flow, a neural net remains a GP $\forall t > 0$;
2. The result of training corresponds to the result of GP inference iff only the readout layer is trained.

Does the previous result hold if some **weights are shared?**

Suppose we want to compute the limit distribution of $WWx$:

$$\underbrace{\begin{pmatrix} W_{11} & W_{12} & \cdots & W_{1n} \\ W_{21} & W_{22} & \cdots & W_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ W_{n1} & W_{n2} & \cdots & W_{nn} \end{pmatrix}}_{\text{all iid } \sim \mathcal{N}(0, \sigma_W^2/n)} \times \underbrace{\begin{pmatrix} W_{11} & W_{12} & \cdots & W_{1n} \\ W_{21} & W_{22} & \cdots & W_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ W_{n1} & W_{n2} & \cdots & W_{nn} \end{pmatrix}}_{\text{the same matrix}} \times \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{pmatrix}}_{\text{vector with iid components}} .$$

We can directly apply CLT if $WW$ **has iid components.**

Let us compute the limit distribution of $(WW)_\alpha$:

$$\underbrace{\begin{pmatrix} W_{\alpha 1} & W_{\alpha\alpha} & \cdots & W_{\alpha n} \end{pmatrix}}_{\text{all iid} \sim \mathcal{N}(0, \sigma_W^2/n)} \underbrace{\begin{pmatrix} W_{11} & W_{1\alpha} & \cdots & W_{1n} \\ W_{\alpha 1} & W_{\alpha\alpha} & \cdots & W_{\alpha n} \\ \cdots & \cdots & \cdots & \cdots \\ W_{n1} & W_{n\alpha} & \cdots & W_{nn} \end{pmatrix}}_{\text{all iid} \sim \mathcal{N}(0, \sigma_W^2/n)} = \underbrace{\begin{pmatrix} \sum(\text{iid w. mean} = 0) \\ W_{\alpha\alpha}^2 + \sum(\text{iid w. mean} = 0) \\ \cdots \\ \sum(\text{iid w. mean} = 0) \end{pmatrix}^\top}_{\text{all iid except for } \alpha\text{'s term}}$$

Fortunately, since $W_{\alpha\alpha} \sim \mathcal{N}(0, \sigma_W^2/n)$, $W_{\alpha\alpha}^2 x_\alpha \propto 1/n \to 0$.

**Conclusion:**

1. **CLT is still applicable** even if the same matrix $W$ is used several times;
2. $(WWx)_\alpha$ and $(W\tilde{W}x)_\alpha$ with $\tilde{W} \overset{\mathrm{d}}{=} W$ have **the same distribution** in the limit.

However, we **cannot substitute** $W$ with its iid copy $\tilde{W}$:

- $WWx$ and $Wx$ are correlated in the limit;
- $W\tilde{W}x$ and $\tilde{W}x$ are not.

Consider $h = Wx$ and $\bar{h} = W\bar{x}$ where $x$ and $\bar{x}$ **can depend on $W$.**

$$\underbrace{\begin{pmatrix} h_1 & \bar{h}_1 \\ h_2 & \bar{h}_2 \\ \cdots & \cdots \\ h_n & \bar{h}_n \end{pmatrix}}_{\substack{\text{each row converges to a zero-mean Gaussian} \\ \text{with covariance } \sigma_W^2 \mathbb{E}\,(x^\top \bar{x})/n \\ \text{as } n \to \infty \text{ by the vector CLT}}} = \underbrace{\begin{pmatrix} W_{11} & W_{12} & \cdots & W_{1n} \\ W_{21} & W_{22} & \cdots & W_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ W_{n1} & W_{n2} & \cdots & W_{nn} \end{pmatrix}}_{\text{all iid } \sim \mathcal{N}(0, \sigma_W^2/n)} \times \underbrace{\begin{pmatrix} x_1 & \bar{x}_1 \\ x_2 & \bar{x}_2 \\ \cdots & \cdots \\ x_n & \bar{x}_n \end{pmatrix}}_{\substack{\text{suppose all components} \\ \text{of each column are iid}}}.$$

$$\text{A } \textsc{Netsor} \text{ program} = (\text{a set of input vars, a sequence of commands}),$$

where variables are of three different **types:**

1. A-vars: matrices with iid Gaussian entries;
2. G-vars: vectors with *asymptotically* iid Gaussian entries;
3. H-vars: images of G-vars by coordinatewise nonlinearities.

Each command generates a new variable from the previous ones using one of the following **ops:**

1. $\mathrm{MatMul}$: $(W : \mathsf{A},\ x : \mathsf{H}) \to Wx : \mathsf{G}$;
2. $\mathrm{LinComb}$: $(\{x_i : \mathsf{G},\ a_i \in \mathbb{R}\}_{i=1}^k) \to \sum_{i=1}^k a_i x_i : \mathsf{G}$;
3. $\mathrm{Nonlin}$: $(\{x_i : \mathsf{G}\}_{i=1}^k,\ \phi : \mathbb{R}^k \to \mathbb{R}) \to \phi(x_{1:k}) : \mathsf{H}$.

**Initialization assumption:**

1. All hidden dimensions are equal to $n$;
2. $\forall W$ : A we sample $W_{\alpha\beta} \sim \mathcal{N}(0, \sigma_W^2/n)$ iid;

**Our goal:** compute the distributions of all G-vars in the program in the limit of $n \to \infty$.

**Claim:**

Let $g^{1:M}$ be a set of all G-vars in the program.

$$\begin{pmatrix} g_1^1 & g_1^2 & \dots & g_1^M \\ g_2^1 & g_2^2 & \dots & g_2^M \\ \dots & \dots & \dots & \dots \\ g_n^1 & g_n^2 & \dots & g_n^M \end{pmatrix}$$

- "Batch" dimension — converges to $\mathcal{N}(\mu, \Sigma)$ with $\mu = \{\mu(g^i)\}_{i=1}^M$ and $\Sigma = \{\Sigma(g^i, g^j)\}_{i,j=1}^M$ defined below;
- Neuron dimension — components tend to be iid Gaussians.

$(g_\alpha^1, \ldots, g_\alpha^M)$ becomes jointly Gaussian with mean and covariance **defined by CLT**[1]:

$$\mu(g) = \begin{cases} 0 & \text{if } g = Wy. \end{cases} \tag{2}$$

$$\Sigma(g, \bar{g}) = \begin{cases} \sigma_W^2 \mathbb{E}\, _Z \phi(Z)\bar{\phi}(Z) & \text{if } g = {\color{red}W}\phi(Z) \text{ and } \bar{g} = {\color{red}W}\bar{\phi}(Z); \\ 0 & \text{else.} \end{cases} \tag{3}$$

Here $Z \sim \mathcal{N}(\mu, \Sigma)$ is a set of all previous G-vars.

[1]we have suppressed the $\mathrm{LinComb}$ op for brevity.

**Theorem (Netsor Master Theorem, [Yang, 2019], informal)**
*Let the* NETSOR *program satisfy the initialization assumption and let all nonlinearities do not grow too fast. Let* $g^{1:M}$ *be a set of all G-vars in the program. Then, for any well-behaved* $\psi$,

$$\underbrace{\frac{1}{n}\sum_{\alpha=1}^{n}\psi(g_\alpha^1,\ldots,g_\alpha^M)}_{\text{empirical mean over rows}} \to \underbrace{\mathbb{E}_{Z\sim\mathcal{N}(\mu,\Sigma)}\psi(Z)}_{\substack{\text{expectation}\\ \text{over the corresponding Gaussian}}} \tag{4}$$

*a.s. as* $n\to\infty$, *where* $\mu=\{\mu(g^i)\}_{i=1}^M$ *and* $\Sigma=\{\Sigma(g^i,g^j)\}_{i,j=1}^M$.

A NETSOR program

- is able to express the **first forward pass** of a wide class of neural nets (i.e. with shared/structured weights, with BNs etc.);
- reveals its limiting Gaussian process behavior.

Questions:

1. Can we express a **backward pass** as a NETSOR program?
2. What is its limiting behavior?

# Intermedia: a Neural Tangent Kernel

Let $f(\cdot;\theta)$ be a parametric model.

Define a **neural tangent kernel** as

$$\Theta_t(\xi,\bar{\xi}) = \underbrace{\nabla_\theta^T f(\xi;\theta_t)\nabla_\theta f(\bar{\xi};\theta_t)}_{\text{"gradient similarity"}}. \tag{5}$$

It drives evolution of **model predictions;** e.g. for square loss:

$$\dot{f}(\bar{\xi};\theta_t) = (\vec{y} - f(\vec{\xi};\theta_t))^\top \Theta_t(\vec{\xi},\bar{\xi}), \quad \text{where } (\vec{\xi},\vec{y}) \text{ is a train dataset.}$$

Assuming $\Theta_t(\cdot,\cdot) \approx \Theta_0(\cdot,\cdot)$ makes the dynamics **analytically tractable.**

**Theorem ([Jacot et al., 2018], informal)**
*Suppose we have a feedforward neural net of width $n$ parameterized in a certain way. Then, as $n \to \infty$,*

1. $\Theta_0(\xi, \bar{\xi})$ *converges to a deterministic* $\mathring{\Theta}(\xi, \bar{\xi})$*;*
2. *Moreover,* $\Theta_t(\xi, \bar{\xi})$ *converges to the same* $\mathring{\Theta}(\xi, \bar{\xi})$*.*

Parameterize $W^l$ as $\omega^l/\sqrt{n_{l-1}}$:

$$\underbrace{f = \frac{1}{\sqrt{n_L}}\omega^{L+1}x^L}_{\text{the network output}}, \qquad \underbrace{x^l = \phi(h^l)}_{\text{activations}}, \quad \underbrace{h^l = \frac{1}{\sqrt{n_{l-1}}}\omega^l x^{l-1}}_{\text{preactivations}}, \quad l \le L; \qquad \omega^l_{ij} \sim \mathcal{N}(0, \sigma_W^2) \text{ iid.}$$

NTK is defined as

$$\Theta(\xi, \bar{\xi}) = \nabla_\theta^T f(\xi;\theta)\nabla_\theta f(\bar{\xi};\theta) = \sum_{l=1}^{L+1} \underbrace{\mathrm{tr}(\nabla_{\omega^l}^T f(\xi)\nabla_{\omega^l} f(\bar{\xi}))}_{\text{"layer-wise gradient similarity"}} . \tag{6}$$

$$\Theta(\xi, \bar{\xi}) = \nabla_\theta^T f(\xi;\theta) \nabla_\theta f(\bar{\xi};\theta) = \sum_{l=1}^{L+1} \underbrace{\mathrm{tr}(\nabla_{\omega^l}^T f(\xi) \nabla_{\omega^l} f(\bar{\xi}))}_{\text{"layer-wise gradient similarity"}} \ . \tag{7}$$

Weight gradient can be expressed as

$$\nabla_{\omega^l} f = \frac{1}{\sqrt{n_{l-1} n_l}} \underbrace{dh^l}_{\substack{\text{backward pass} \\ \text{up to the layer } l}} \times \underbrace{x^{l-1,\top}}_{\substack{\text{forward pass} \\ \text{up to the layer } l-1}} \ , \quad \text{where } dh^l \propto \nabla_{h_l} f. \tag{8}$$

Plug (8) into (7):

$$\Theta(\xi, \bar{\xi}) = \sum_{l=1}^{L+1} \underbrace{\left(\frac{dh^{l,\top} d\bar{h}^l}{n_l}\right)}_{\text{"backward pass similarity"}} \times \underbrace{\left(\frac{x^{l-1,\top} \bar{x}^{l-1}}{n_{l-1}}\right)}_{\text{"forward pass similarity"}} \ . \tag{9}$$

Consider the second multiplier:

$$\frac{x^{l-1,\top}\bar{x}^{l-1}}{n_{l-1}} = \frac{1}{n_{l-1}}\sum_{\alpha=1}^{n_{l-1}}\phi(h_\alpha^{l-1})\phi(\bar{h}_\alpha^{l-1}) = \underbrace{\frac{1}{n_{l-1}}\sum_{\alpha=1}^{n_{l-1}}\psi(h_\alpha^{l-1},\bar{h}_\alpha^{l-1})}_{\text{the limit is given by the Master theorem!}} \quad \text{for } \psi(x,y)=\phi(x)\phi(y).$$

**Can we compute the limit of the first multiplier in the same way?**

For simplicity, assume $n_1 = \ldots = n_L = n$. Recall $W^l = \omega^l/\sqrt{n}$.

**Relations between forward and backward passes:**

| Forward pass: | Backward pass: |
|---|---|
| $x^l = \phi(h^l) : \mathrm{Nonlin}$ | $dh^l = dx^l \odot \phi'(h^l) : \mathrm{Nonlin}$ |
| $h^l = W^l x^{l-1} : \mathrm{MatMul}$ | $dx^{l-1} = W^{l,\top} dh^l : \mathrm{MatMul?}$ |

**Problems:**

1. $W$ and $W^\top$ cannot be both input variables since they are dependent;
2. A Netsor program does not allow for multiplying by a transposed A-var.

**A Netsor program cannot express the backward pass!**

$$\text{A } \textsc{Netsor}\top \text{ program} = (\text{a set of input vars}, \text{ a sequence of commands}),$$

where variables are of three different **types:**

1. A-vars: matrices with iid Gaussian entries;
2. G-vars: vectors with *asymptotically* iid Gaussian entries;
3. H-vars: images of G-vars by coordinatewise nonlinearities.

Each command generates a new variable from the previous ones using one of the following **ops:**

1. Trsp: $W : \mathsf{A} \to W^\top : \mathsf{A}$;
2. MatMul: $(W : \mathsf{A},\ x : \mathsf{H}) \to Wx : \mathsf{G}$;
3. LinComb: $(\{x_i : \mathsf{G},\ a_i \in \mathbb{R}\}_{i=1}^k) \to \sum_{i=1}^k a_i x_i : \mathsf{G}$;
4. Nonlin: $(\{x_i : \mathsf{G}\}_{i=1}^k,\ \phi : \mathbb{R}^k \to \mathbb{R}) \to \phi(x_{1:k}) : \mathsf{H}$.

**Can we keep the same symbolic rules for mean and covariance of G-vars?**

$$\mu(g) = \begin{cases} 0 & \text{if } g = Wy; \\ 0? & \text{if } g = W^\top y. \end{cases} \tag{10}$$

$$\Sigma(g, \bar{g}) = \begin{cases} \sigma_W^2 \mathbb{E}_Z \phi(Z) \bar{\phi}(Z) & \text{if } g = W\phi(Z) \text{ and } \bar{g} = W\bar{\phi}(Z); \\ \text{some other rule?} & \text{if } g = W\phi(Z) \text{ and } \bar{g} = W^\top \bar{\phi}(Z); \\ 0 & \text{else.} \end{cases} \tag{11}$$

Here $Z \sim \mathcal{N}(\mu, \Sigma)$ is a set of all previous G-vars.

Let us check that $\mu(WWx) = 0$:

$$\underbrace{\begin{pmatrix} W_{\alpha 1} & W_{\alpha\alpha} & \cdots & W_{\alpha n} \end{pmatrix}}_{\text{all iid} \sim \mathcal{N}(0,\sigma_W^2/n)} \underbrace{\begin{pmatrix} W_{11} & W_{1\alpha} & \cdots & W_{1n} \\ W_{\alpha 1} & W_{\alpha\alpha} & \cdots & W_{\alpha n} \\ \cdots & \cdots & \cdots & \cdots \\ W_{n1} & W_{n\alpha} & \cdots & W_{nn} \end{pmatrix}}_{\text{all iid} \sim \mathcal{N}(0,\sigma_W^2/n)} = \underbrace{\begin{pmatrix} \sum(\text{iid w. mean} = 0) \\ W_{\alpha\alpha}^2 + \sum(\text{iid w. mean} = 0) \\ \cdots \\ \sum(\text{iid w. mean} = 0) \end{pmatrix}}_{\text{all iid except for } \alpha\text{'s term}}^{\top}$$

$$\mu(WWx) = \mathbb{E}\left((WWx)_\alpha\right) = \mathbb{E}\underbrace{\left(\sum_{\beta\neq\alpha}\sum_{\gamma} W_{\alpha\beta}W_{\beta\gamma}x_\gamma\right)}_{\mathbb{E}(\sum \text{iid w. mean}=0)=0} + \mathbb{E}\underbrace{\left(\sum_{\gamma} W_{\alpha\alpha}W_{\alpha\gamma}x_\gamma\right)}_{=\mathbb{E}\, W_{\alpha\alpha}^2 x_\alpha \propto 1/n \to 0}. \qquad (12)$$

**Do we have** $\mu(WW^T x) = 0$?

Let us compute $\mu(W W^T x)$:

$$\underbrace{\begin{pmatrix} W_{\alpha 1} & W_{\alpha\alpha} & \cdots & W_{\alpha n} \end{pmatrix}}_{\text{all iid} \sim \mathcal{N}(0, \sigma_W^2/n)} \underbrace{\begin{pmatrix} W_{11} & W_{\alpha 1} & \cdots & W_{n1} \\ W_{1\alpha} & W_{\alpha\alpha} & \cdots & W_{n\alpha} \\ \cdots & \cdots & \cdots & \cdots \\ W_{1n} & W_{\alpha n} & \cdots & W_{nn} \end{pmatrix}}_{\text{all iid} \sim \mathcal{N}(0, \sigma_W^2/n)} = \underbrace{\begin{pmatrix} \sum(\text{iid w. mean} = 0) \\ \sum_\beta W_{\alpha\beta}^2 \\ \cdots \\ \sum(\text{iid w. mean} = 0) \end{pmatrix}}_{\text{all iid except for } \alpha\text{'s term}}^\top$$

$$\mu(W W^\top x) = \mathbb{E}\left((W W^\top x)_\alpha\right) = \underbrace{\mathbb{E}\left(\sum_\beta \sum_{\gamma \neq \alpha} W_{\alpha\beta} W_{\gamma\beta} x_\gamma\right)}_{\mathbb{E}\left(\sum \text{iid w. mean}=0\right)=0} + \underbrace{\mathbb{E}\left(\sum_\beta W_{\alpha\beta}^2 x_\beta\right)}_{\text{converges to } \sigma_W^2 \mu(x) \neq 0} . \qquad (13)$$

The previous symbolic rules are not applicable for **general** NETSOR⊤ programs,

**but**

they are applicable to NETSOR⊤ programs expressing backpropagation.

**Claim:** the rule

$$\mu(g) = 0 \quad \text{if } g = Wy. \tag{14}$$

works for NETSORT programs expressing backpropagation.

**Evidence:** consider $dx^{l-1} = W^{l,T}(dx^l \odot \phi'(h^l))$. Let $\phi(z) = z^2/2$:

$$\mu(dx^{l-1}) = \mathbb{E}\left(dx_\alpha^{l-1}\right) = \underbrace{\mathbb{E}\left(\sum_\beta W_{\beta\alpha}^l dx_\beta^l \sum_{\gamma\neq\beta} W_{\beta\gamma}^l x_\gamma^{l-1}\right)}_{\mathbb{E}\left(\sum \text{iid w. mean}=0\right)=0} + \underbrace{\mathbb{E}\left(x_\alpha^{l-1}\sum_\beta (W_{\beta\alpha}^l)^2 dx_\beta^l\right)}_{\text{converges to } \mu(x^{l-1})\sigma_W^2\mu(dx^l)}. \tag{15}$$

Hence $\mu(dx^{l-1}) \propto \mu(dx^l)$ which by induction implies $\mu(dx^{l-1}) \propto \mu(dx^L)$.

But $dx^L = \omega^{L+1}$! Hence $\mu(dx^{l-1}) = \mu(\omega^{L+1}) = 0$.

**Proposition ([2])**
*Consider a neural network and a* $\textsc{Netsor}\top$ *program expressing its backward pass.*

*The symbolic rules for* $\mu$ *and* $\Sigma$ *are valid, if*

1. *The output layer has zero mean;*
2. *It is sampled independently from other parameters;*
3. *It is not used anywhere else in the program.*

[2] There is a more general condition called "BP-likeness" which we do not show here.

**Theorem (Netsor$\top$ Master Theorem, [Yang, 2020a]; informal)**
*Let a* $\text{N}\textsc{etsor}\top$ *program* **express a forward or a backward pass** *in a neural network, satisfy the initialization assumption, and let all nonlinearities do not grow too fast. Let $g^{1:M}$ be a set of all* G*-vars in the program. Then, for any well-behaved $\psi : \mathbb{R}^M \to \mathbb{R}$,*

$$\frac{1}{n}\sum_{\alpha=1}^{n} \psi(g_\alpha^1, \ldots, g_\alpha^M) \to \mathbb{E}_{Z\sim\mathcal{N}(\mu,\Sigma)}\psi(Z)$$

*a.s. as $n \to \infty$, where $\mu = \{\mu(g^i)\}_{i=1}^M$ and $\Sigma = \{\Sigma(g^i, g^j)\}_{i,j=1}^M$.*

**The result is (almost) the same as for Netsor programs!**

Back to NTK computation:

$$\Theta(\xi, \bar{\xi}) = \sum_{l=1}^{L+1} \nabla_{\omega^l}^T f(\xi) \nabla_{\omega^l} f(\bar{\xi}) = \sum_{l=1}^{L+1} \left( \frac{dh^{l,\top} d\bar{h}^l}{n_l} \right) \left( \frac{x^{l-1,\top} \bar{x}^{l-1}}{n_{l-1}} \right). \tag{16}$$

Consider the first multiplier:

$$\frac{dh^{l,\top} \bar{d}h^l}{n_l} = \frac{1}{n_l} \sum_{\alpha=1}^{n_l} dx_\alpha^l d\bar{x}_\alpha^l \phi'(h_\alpha^l) \phi'(\bar{h}_\alpha^l) = \underbrace{\frac{1}{n_l} \sum_{\alpha=1}^{n_l} \psi(dx_\alpha^l, d\bar{x}_\alpha^l, h_\alpha^l, \bar{h}_\alpha^l)}_{\text{the limit is given by the Master theorem!}}$$

$$\text{for } \psi(x, y, z, w) = xy\phi'(z)\phi'(w). \tag{17}$$

A BP-like NETSOR$\top$ program

- is able to express the **first forward and backward passes** of a wide class of neural nets (i.e. with shared/structured weights, with BNs etc.);
- reveals their limiting Gaussian process behavior;
- can be applied to initial NTK computation.

Jacot, A., Gabriel, F., and Hongler, C. (2018).
**Neural tangent kernel: Convergence and generalization in neural networks.**
In *Advances in neural information processing systems*, pages 8571–8580.

Matthews, A. G. d. G., Hron, J., Rowland, M., Turner, R. E., and Ghahramani, Z. (2018).
**Gaussian process behaviour in wide deep neural networks.**
In *International Conference on Learning Representations*.

Pennington, J., Schoenholz, S., and Ganguli, S. (2017).
**Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice.**
In *Advances in neural information processing systems*, pages 4785–4795.

Yang, G. (2019).
**Tensor programs i: Wide feedforward or recurrent neural networks of any architecture are gaussian processes.**
*arXiv preprint arXiv:1910.12478*.

Yang, G. (2020a).

**Tensor programs ii: Neural tangent kernel for any architecture.**
*arXiv preprint arXiv:2006.14548*.

Yang, G. (2020b).
**Tensor programs iii: Neural matrix laws.**
*arXiv preprint arXiv:2009.10685*.