The Neural Tangent Kernel

Equivariance, Data Augmentation and Corrections from Feynman Diagrams

Philipp Misof

Department of Mathematical Sciences, Division of Algebra and Geometry

August 28, 2025

UNIVERSITY OF GOTHENBURG

CHALMERS UNIVERSITY OF TECHNOLOGY

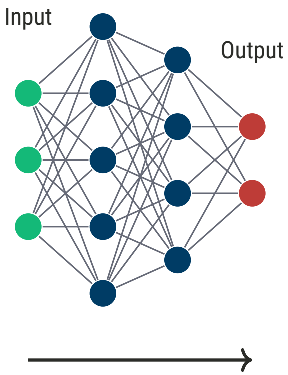
WASP WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM

Feedforward Neural Network (NN)

alias *Multi-layer Perceptron* (MLP)

Feedforward Neural Network (NN)

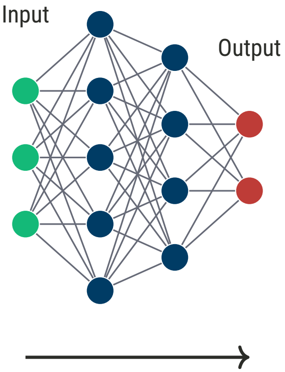
alias *Multi-layer Perceptron* (MLP)



Feedforward Neural Network (NN)

alias *Multi-layer Perceptron* (MLP)

- Is a map $\mathcal{N}^{(L)} : \mathbb{R}^d \to \mathbb{R}^{n_L}$.



Feedforward **Neural Network** (NN)

alias *Multi-layer Perceptron* (MLP)

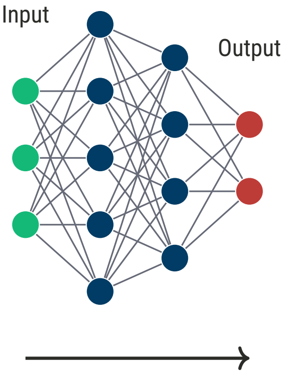
- Is a map $\mathcal{N}^{(L)} : \mathbb{R}^d \to \mathbb{R}^{n_L}$.
- Recursively defined via **layers** $\mathcal{N}^{(\ell)}$

Activation function ↓ Input ↓

$$\mathcal{N}^{(\ell)}(x) = \sigma\left(\frac{1}{\sqrt{n_{\ell-1}}} W^{(\ell)} \mathcal{N}^{(\ell-1)}(x) + b^{(\ell)}\right),$$

weights ↑ biases ↑

for $\ell < L$, $\quad \mathcal{N}^{(L)}(x) = W^{(L)} \mathcal{N}^{(L-1)}(x)$.



Feedforward Neural Network (NN)

alias *Multi-layer Perceptron* (MLP)

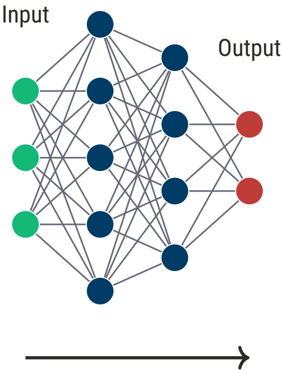
- Is a map $\mathcal{N}^{(L)} : \mathbb{R}^d \to \mathbb{R}^{n_L}$.
- Recursively defined via **layers** $\mathcal{N}^{(\ell)}$

Activation function; Input; weights; biases

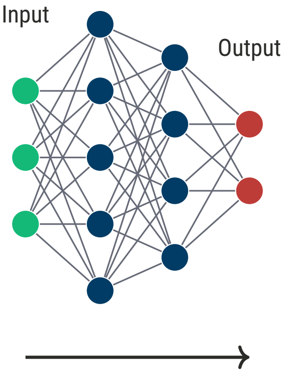
$$\mathcal{N}^{(\ell)}(x) = \sigma\left(\frac{1}{\sqrt{n_{\ell-1}}} W^{(\ell)} \mathcal{N}^{(\ell-1)}(x) + b^{(\ell)}\right),$$

for $\ell < L$, $\quad \mathcal{N}^{(L)}(x) = W^{(L)} \mathcal{N}^{(L-1)}(x)$.

- $\theta_\mu \in \{W_{ij}^{(\ell)}, b_i^{(\ell)}\}_{\ell,i,j}$ are the **parameters**



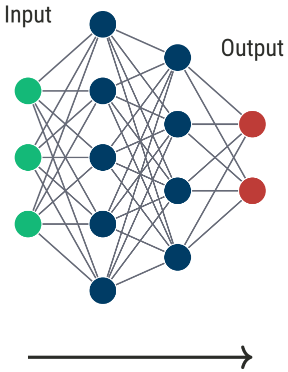
Initialization



Initialization

parameters sampled **iid**

$$W_{ij}^{(\ell)}, b_i^{\ell} \sim \mathcal{N}(0, 1)$$



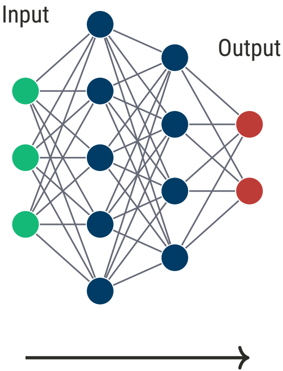
Initialization

parameters sampled **iid**

$$W_{ij}^{(\ell)}, b_i^{\ell} \sim \mathcal{N}(0, 1)$$

Training

- training **data** $\{(x_i, y_i)\}_{i=1}^{n_{\text{train}}}$



Initialization

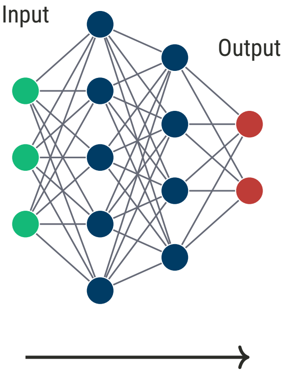
parameters sampled **iid**

$$W_{ij}^{(\ell)}, b_i^{\ell} \sim \mathcal{N}(0, 1)$$

Training

- training **data** $\{(x_i, y_i)\}_{i=1}^{n_{\text{train}}}$
- loss function $L(y, \hat{y})$, **empirical loss**

$$\mathcal{L}(\theta) = \frac{1}{n_{\text{train}}} \sum_i L(y_i, \mathcal{N}_\theta(x_i))$$



Initialization

parameters sampled **iid**

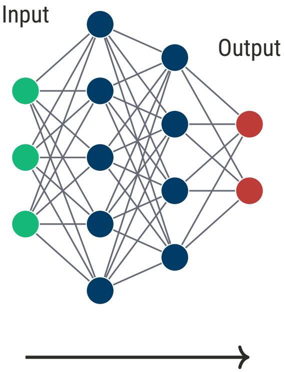
$$W_{ij}^{(\ell)}, b_i^\ell \sim \mathcal{N}(0,1)$$

Training

- training **data** $\{(x_i, y_i)\}_{i=1}^{n_{\text{train}}}$
- loss function $L(y, \hat{y})$, **empirical loss**

$$\mathcal{L}(\theta) = \frac{1}{n_{\text{train}}} \sum_i L(y_i, \mathcal{N}_\theta(x_i))$$

- Training = **Minimizing** the empirical loss



Initialization

parameters sampled **iid**

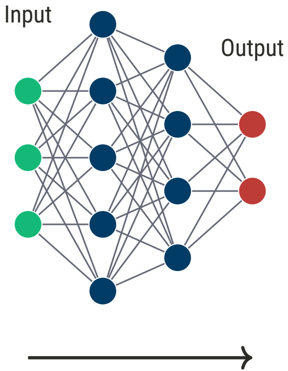
$$W_{ij}^{(\ell)}, b_i^\ell \sim \mathcal{N}(0,1)$$

Training

- training **data** $\{(x_i, y_i)\}_{i=1}^{n_\text{train}}$
- loss function $L(y, \hat{y})$, **empirical loss**

$$\mathcal{L}(\theta) = \frac{1}{n_\text{train}} \sum_i L(y_i, \mathcal{N}_\theta(x_i))$$

- Training = **Minimizing** the empirical loss
- Almost always **Gradient Descent** (GD) based.



Training Dynamics

Assume **Gradient Flow**

Learning rate

$$\frac{\mathrm{d}\theta_\mu(t)}{\mathrm{d}t} = -\eta \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta_\mu}$$

Training Dynamics

Assume **Gradient Flow**

Learning rate

$$\frac{\mathrm{d}\theta_\mu(t)}{\mathrm{d}t} = -\eta \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta_\mu}$$

Chain rule $\rightarrow$

$$\frac{\mathrm{d}\mathcal{N}}{\mathrm{d}t}(x) = -\eta \sum_{i=1}^{n_{\text{train}}} \Theta_t(x, x_i) \frac{\partial \mathcal{L}}{\partial \mathcal{N}(x_i)}$$

Training Dynamics

Assume **Gradient Flow**

Learning rate

$$\frac{\mathrm{d}\theta_\mu(t)}{\mathrm{d}t} = -\eta \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta_\mu}$$

Chain rule $\rightarrow$

$$\frac{\mathrm{d}\mathcal{N}}{\mathrm{d}t}(x) = -\eta \sum_{i=1}^{n_{\text{train}}} \Theta_t(x, x_i) \frac{\partial \mathcal{L}}{\partial \mathcal{N}(x_i)}$$

(empirical) Neural Tangent Kernel

Training Dynamics

Assume **Gradient Flow**

Learning rate

$$\frac{\mathrm{d}\theta_\mu(t)}{\mathrm{d}t} = -\eta \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta_\mu}$$

Chain rule $\to$

$$\frac{\mathrm{d}\mathcal{N}}{\mathrm{d}t}(x) = -\eta \sum_{i=1}^{n_{\text{train}}} \Theta_t(x, x_i) \frac{\partial \mathcal{L}}{\partial \mathcal{N}(x_i)}$$

(empirical) Neural Tangent Kernel

$$\Theta_t(x, x') = \sum_\mu \frac{\partial \mathcal{N}(x)}{\partial \theta_\mu} \left(\frac{\partial \mathcal{N}(x')}{\partial \theta_\mu} \right)^\mathsf{T}$$

Training Dynamics

Assume **Gradient Flow**

Learning rate

$$\frac{\mathrm{d}\theta_\mu(t)}{\mathrm{d}t} = -\eta \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta_\mu}$$

Chain rule $\rightarrow$

$$\frac{\mathrm{d}\mathcal{N}}{\mathrm{d}t}(x) = -\eta \sum_{i=1}^{n_{\text{train}}} \Theta_t(x, x_i) \frac{\partial \mathcal{L}}{\partial \mathcal{N}(x_i)}$$

(empirical) Neural Tangent Kernel

$$\Theta_t(x, x') = \sum_\mu \frac{\partial \mathcal{N}(x)}{\partial \theta_\mu} \left(\frac{\partial \mathcal{N}(x')}{\partial \theta_\mu} \right)^{\mathsf{T}}$$

Intuition: Similarity measure of gradients at different inputs

Training Dynamics

Assume **Gradient Flow**

Learning rate

$$\frac{\mathrm{d}\theta_\mu(t)}{\mathrm{d}t} = -\eta \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta_\mu}$$

Chain rule $\rightarrow$

$$\frac{\mathrm{d}\mathcal{N}}{\mathrm{d}t}(x) = -\eta \sum_{i=1}^{n_{\text{train}}} \Theta_t(x, x_i) \frac{\partial \mathcal{L}}{\partial \mathcal{N}(x_i)}$$

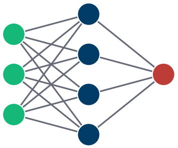
(empirical) Neural Tangent Kernel

$$\Theta_t(x, x') = \sum_\mu \frac{\partial \mathcal{N}(x)}{\partial \theta_\mu} \left(\frac{\partial \mathcal{N}(x')}{\partial \theta_\mu} \right)^{\mathsf{T}}$$

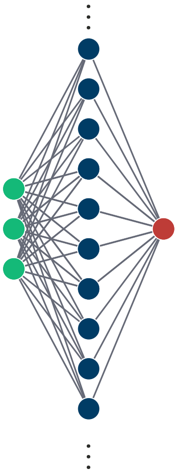
Intuition: Similarity measure of gradients at different inputs

Θ_t is **time-dependent** and **stochastic**.

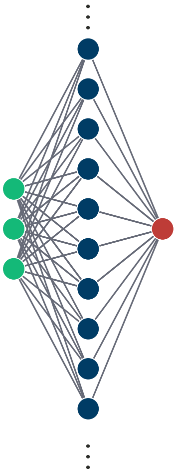
What happens when we make the hidden layers **very wide**?



What happens when we make the hidden layers **very wide**?



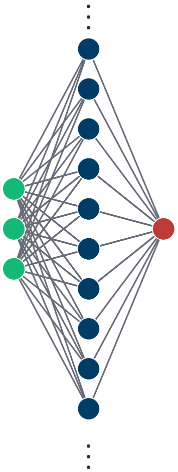
What happens when we make the hidden layers **very wide**?



- Obtain a centered **Gaussian process**
- With covariance (**NNGP**) kernel

$$\mathbb{E}\left[\mathcal{N}(x)\mathcal{N}(x')^\top\right] = K(x,x')\mathbb{I}_{n_L}$$

What happens when we make the hidden layers **very wide**?

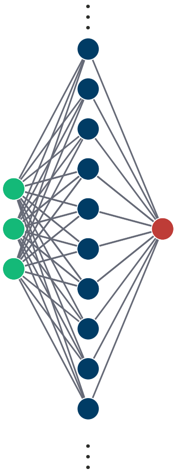


- Obtain a centered **Gaussian process**
- With covariance (**NNGP**) kernel

$$\mathbb{E}\left[\mathcal{N}(x)\mathcal{N}(x')^{\mathsf{T}}\right] = K(x, x')\mathbb{I}_{n_L}$$

Due to the *Law of Large Numbers*.

What happens when we make the hidden layers **very wide**?



- Obtain a centered **Gaussian process**
- With covariance (**NNGP**) kernel

$$\mathbb{E}\left[\mathcal{N}(x)\mathcal{N}(x')^{\mathsf{T}}\right] = K(x,x')\mathbb{I}_{n_L}$$

Due to the *Law of Large Numbers*.

Similarly

Freezing of the NTK

(Jacot, Gabriel, and Hongler 2018)

$$\Theta_t(x,x') \to \mathbb{E}\left[\Theta_t(x,x')\right] = \Theta(x,x')\mathbb{I}_{n_L}$$

What do we gain?

What do we gain?

is now deterministic and time-independent

$$\frac{\mathrm{d}\mathcal{N}}{\mathrm{d}t}(x) = -\eta \sum_{i=1}^{n_{\text{train}}} \Theta(x, x_i) \frac{\partial \mathcal{L}}{\partial \mathcal{N}(x_i)}$$

What do we gain?

is now deterministic and time-independent

$$\frac{\mathrm{d}\mathcal{N}}{\mathrm{d}t}(x) = -\eta \sum_{i=1}^{n_{\text{train}}} \Theta(x, x_i) \frac{\partial \mathcal{L}}{\partial \mathcal{N}(x_i)}$$

- Let's assume **square loss**

What do we gain?

is now deterministic and time-independent

$$\frac{\mathrm{d}\mathcal{N}}{\mathrm{d}t}(x) = -\eta \sum_{i=1}^{n_{\text{train}}} \Theta(x, x_i) \frac{\partial \mathcal{L}}{\partial \mathcal{N}(x_i)}$$

- Let's assume **square loss**
- Then we obtain a **simple ODE** for the NN mean $\mu_t(x) = \mathbb{E}\left[\mathcal{N}_t(x)\right]$

What do we gain?

is now deterministic and time-independent

$$\frac{d\mathcal{N}}{dt}(x) = -\eta \sum_{i=1}^{n_{\text{train}}} \Theta(x, x_i) \frac{\partial \mathcal{L}}{\partial \mathcal{N}(x_i)}$$

- Let's assume **square loss**
- Then we obtain a **simple ODE** for the NN mean $\mu_t(x) = \mathbb{E}\left[\mathcal{N}_t(x)\right]$

What do we gain?

is now deterministic and time-independent

$$\frac{\mathrm{d}\mathcal{N}}{\mathrm{d}t}(x) = -\eta \sum_{i=1}^{n_{\text{train}}} \Theta(x, x_i) \frac{\partial \mathcal{L}}{\partial \mathcal{N}(x_i)}$$

- Let's assume **square loss**
- Then we obtain a **simple ODE** for the NN mean $\mu_t(x) = \mathbb{E}\left[\mathcal{N}_t(x)\right]$

Analytic solution

$$\mu_t(x) = \Theta(x, X)\Theta(X, X)^{-1}(\mathbb{I} - e^{-\eta\Theta(X,X)t})Y$$

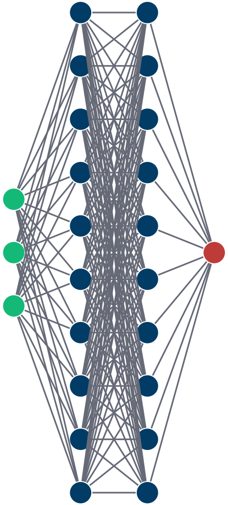
Train inputs

Train labels

How is the NTK computed for a given architecture?

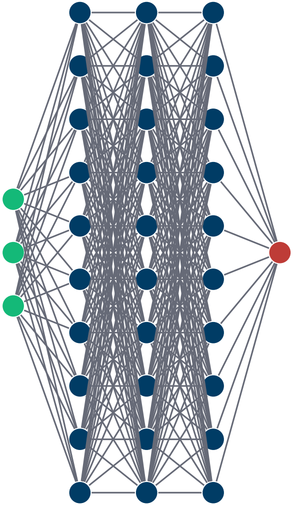
How is the NTK computed for a given architecture? $\rightarrow$ **Layer by layer**

How is the NTK computed for a given architecture? $\rightarrow$ **Layer by layer**



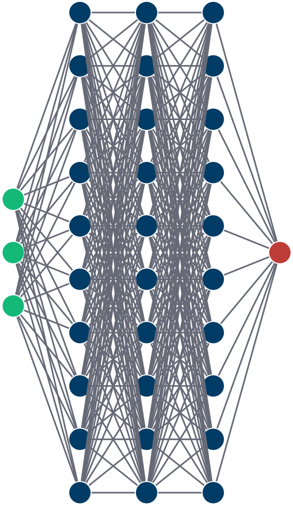
$\rightarrow K^{(3)}, \Theta^{(3)}$

How is the NTK computed for a given architecture? $\rightarrow$ **Layer by layer**

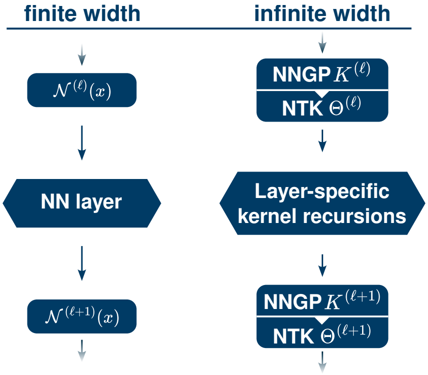


$\rightarrow K^{(3)}, \Theta^{(4)}$

How is the NTK computed for a given architecture? $\rightarrow$ **Layer by layer**

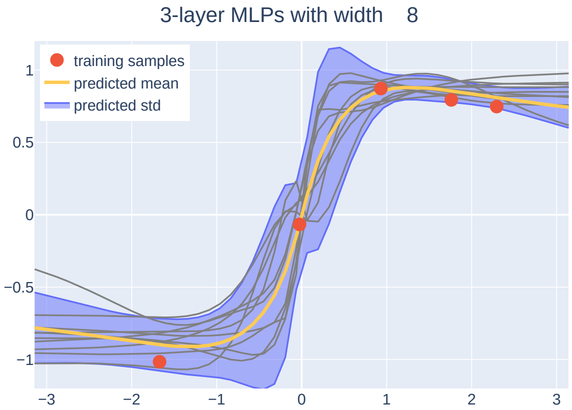


$\rightarrow K^{(3)}, \Theta^{(4)}$

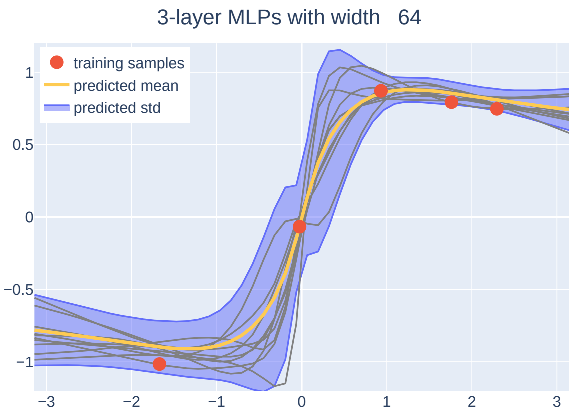


Toy example: Learning $\sin(x)$

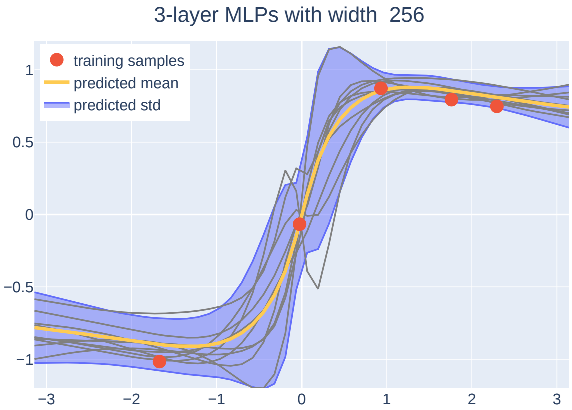
Toy example: Learning $\sin(x)$



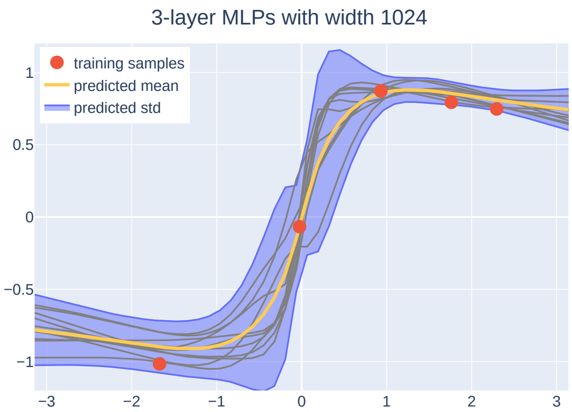
Toy example: Learning $\sin(x)$



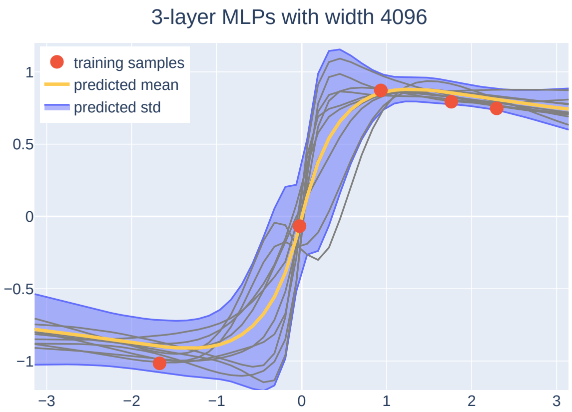
Toy example: Learning $\sin(x)$



Toy example: Learning $\sin(x)$



Toy example: Learning $\sin(x)$



Equivariant Neural Tangent Kernels

Philipp Misof [1] **Pan Kessel** [2] **Jan E. Gerken** [1]

Abstract

Little is known about the training dynamics of equivariant neural networks, in particular how it compares to data augmented training of their non-equivariant counterparts. Recently, neural tangent kernels (NTKs) have emerged as a powerful tool to analytically study the training dynamics of wide neural networks. In this work, we take an important step towards a theoretical understanding of training dynamics of equivariant models by deriving neural tangent kernels for a broad class of equivariant architectures based on group convolutions. As a demonstration of the capabilities of our framework, we show an interesting relationship between data augmentation and group convolutional networks. Specifically, we prove that they share the same expected pre-

Schütt et al., 2021; Unke et al., 2021). Other application areas include particle physics (Bogatskiy et al., 2020), cosmology (Perraudin et al., 2019) and even fairness in large language models (Basu et al., 2023).

Recently, there has been a number of works which avoid equivariant architectures but rely on data augmentation to approximately learn equivariance, most notably AlphaFold3 (Abramson et al., 2024). This has the potential advantage that non-equivariant architectures may offer better training dynamics, for example favorable scaling capabilities. There has been a vigorous debate on this subject with some empirical works claiming superiority of equivariant architectures (Gerken et al., 2022; Brehmer et al., 2024) while others suggest the opposite (Wang et al., 2024; Abramson et al., 2024). One challenging aspect to conclusively settle the matter is that there is no good theoretical understanding of how the equivariant and the purely augmentation-based

Equivariant Neural Tangent Kernels

Philipp Misof [1] **Pan Kessel** [2] **Jan E. Gerken** [1]

Abstract

Little is known about the training dynamics of equivariant neural networks, in particular how it compares to data augmented training of their non-equivariant counterparts. Recently, neural tangent kernels (NTKs) have emerged as a powerful tool to analytically study the training dynamics of wide neural networks. In this work, we take an important step towards a theoretical understanding of training dynamics of equivariant models by deriving neural tangent kernels for a broad class of equivariant architectures based on group convolutions. As a demonstration of the capabilities of our framework, we show an interesting relationship between data augmentation and group convolutional networks. Specifically, we prove that they share the same expected pre-

Schütt et al., 2021; Unke et al., 2021). Other application areas include particle physics (Bogatskiy et al., 2020), cosmology (Perraudin et al., 2019) and even fairness in large language models (Basu et al., 2023).

Recently, there has been a number of works which avoid equivariant architectures but rely on data augmentation to approximately learn equivariance, most notably AlphaFold3 (Abramson et al., 2024). This has the potential advantage that non-equivariant architectures may offer better training dynamics, for example favorable scaling capabilities. There has been a vigorous debate on this subject with some empirical works claiming superiority of equivariant architectures (Gerken et al., 2022; Brehmer et al., 2024) while others suggest the opposite (Wang et al., 2024; Abramson et al., 2024). One challenging aspect to conclusively settle the matter is that there is no good theoretical understanding of how the equivariant and the purely augmentation-based

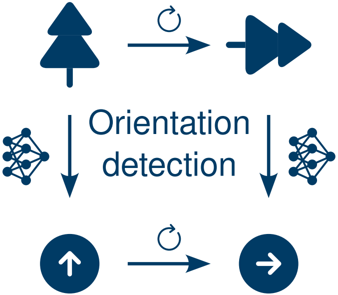
Presented at the **ICML 2025** in Vancouver

Symmetries in Machine Learning

Want to enforce symmetry w.r.t a group G acting on the input signal $f : X \to \mathbb{R}^d$

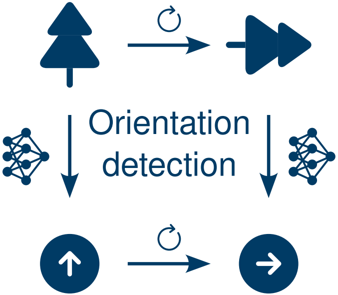
Symmetries in Machine Learning

Want to enforce symmetry w.r.t a group G acting on the input signal $f : X \to \mathbb{R}^d$



Symmetries in Machine Learning

Want to enforce symmetry w.r.t a group G acting on the input signal $f : X \to \mathbb{R}^d$

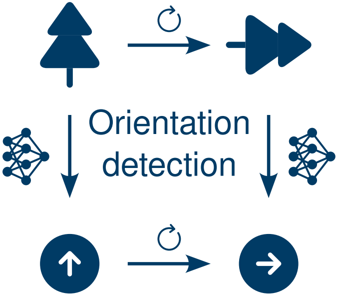


$$\begin{array}{ccc} f & \xrightarrow{\rho_{\text{in}}(g)} & \rho_{\text{in}}(g)(f) \\ \downarrow \mathcal{N} & & \downarrow \mathcal{N} \\ \mathcal{N}(f) & \xrightarrow{\rho_{\text{out}}(g)} & \rho_{\text{out}}(g)[\mathcal{N}(f)] \end{array}$$

$$\forall g \in G$$

Symmetries in Machine Learning

Want to enforce symmetry w.r.t a group G acting on the input signal $f : X \to \mathbb{R}^d$

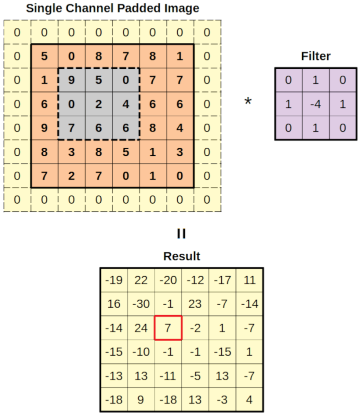


$$\begin{array}{ccc} f & \xrightarrow{\rho_{\text{in}}(g)} & \rho_{\text{in}}(g)(f) \\ \downarrow \mathcal{N} & & \downarrow \mathcal{N} \\ \mathcal{N}(f) & \xrightarrow{\rho_{\text{out}}(g)} & \rho_{\text{out}}(g)[\mathcal{N}(f)] \end{array}$$

$$\forall g \in G$$

This is called **equivariance**

Convolutional Neural Networks (CNNs)



(https://en.wikipedia.org/wiki/Convolutional_neural_network)

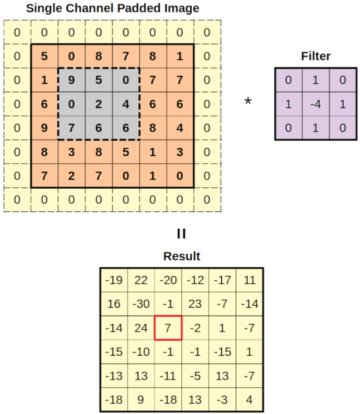
Convolutional Neural Networks (CNNs)

Classic convolution layer

filter

$$[\mathcal{N}^{(1)}(f)](y) = \frac{1}{\sqrt{|S_\kappa|}} \int_{\mathbb{R}^d} \mathrm{d}x \, \kappa(x-y) f(x)$$

filter support → domain of input signal



(https://en.wikipedia.org/wiki/Convolutional_neural_network)

Convolutional Neural Networks (CNNs)

Classic convolution layer

$$[\mathcal{N}^{(1)}(f)](y) = \frac{1}{\sqrt{|S_\kappa|}} \int_{\mathbb{R}^d} \mathrm{d}x \, \kappa(x-y) f(x)$$

filter

filter support

domain of input signal

Equivariant w.r.t. translation group $G = \mathbb{R}^d$

Single Channel Padded Image

0	0	0	0	0	0	0	0
0	5	0	8	7	8	1	0
0	1	9	5	0	7	7	0
0	6	0	2	4	6	6	0
0	9	7	6	6	8	4	0
0	8	3	8	5	1	3	0
0	7	2	7	0	1	0	0
0	0	0	0	0	0	0	0

*

Filter

0	1	0
1	-4	1
0	1	0

=

Result

-19	22	-20	-12	-17	11
16	-30	-1	23	-7	-14
-14	24	7	-2	1	-7
-15	-10	-1	-1	-15	1
-13	13	-11	-5	13	-7
-18	9	-18	13	-3	4

(https://en.wikipedia.org/wiki/Convolutional_neural_network)

Group Convolutional Neural Networks (GCNNs)

Group Convolutional Neural Networks (GCNNs)

Generalization of a CNN to other groups G acting on a homogeneous space X.

$$[\mathcal{N}^{(1)}(f)](y) = \frac{1}{\sqrt{|S_\kappa|}} \int_{\mathbb{R}^d} \mathrm{d}x \, \kappa(x - y) f(x)$$

Group Convolutional Neural Networks (GCNNs)

Generalization of a CNN to other groups G acting on a homogeneous space X.

$$[\mathcal{N}^{(1)}(f)](g) = \frac{1}{\sqrt{|S_\kappa|}} \int_X \mathrm{d}h \, \kappa(g^{-1}h)[(f)](h)$$

group element

Remark: Subtle difference between the first (*lifting*) layer and subsequent layers

Group Convolutional Neural Networks (GCNNs)

Generalization of a CNN to other groups G acting on a homogeneous space X.

$$[\mathcal{N}^{(1)}(f)](g) = \frac{1}{\sqrt{|S_\kappa|}} \int_X \mathrm{d}h \, \kappa(g^{-1}h)[(f)](h)$$

group element

Remark: Subtle difference between the first (*lifting*) layer and subsequent layers

Equivariant w.r.t. the **regular representation**

Group Convolutional Neural Networks (GCNNs)

Generalization of a CNN to other groups G acting on a homogeneous space X.

$$[\mathcal{N}^{(1)}(f)](g) = \frac{1}{\sqrt{|S_\kappa|}} \int_X \mathrm{d}h \, \kappa(g^{-1}h)[(f)](h)$$

group element

Remark: Subtle difference between the first (*lifting*) layer and subsequent layers

Equivariant w.r.t. the **regular representation**

Group pooling

$$\mathcal{N}^{(\ell+1)}(f) = \frac{1}{\mathrm{vol}(G)} \int_G \mathrm{d}g \, [\mathcal{N}^{(\ell)}(f)](g)$$

The Equivariant NTK

The Equivariant NTK

$$\mathbb{E}\left[\sum_{\mu} \frac{\partial[\mathcal{N}^{(\ell)}(f)](g)}{\partial\theta_\mu}\left(\frac{\partial[\mathcal{N}^{(\ell)}(f')](g')}{\partial\theta_\mu}\right)^{\top}\right]$$

The Equivariant NTK

$$\Theta^{(\ell)}_{g,g'}(f, f') = \mathbb{E}\left[\sum_{\mu} \frac{\partial[\mathcal{N}^{(\ell)}(f)](g)}{\partial\theta_\mu}\left(\frac{\partial[\mathcal{N}^{(\ell)}(f')](g')}{\partial\theta_\mu}\right)^{\mathsf{T}}\right]$$

↑ Evaluation point in group space

The Equivariant NTK

$$\Theta^{(\ell)}_{g,g'}(f, f') = \mathbb{E}\left[\sum_{\mu} \frac{\partial[\mathcal{N}^{(\ell)}(f)](g)}{\partial\theta_\mu} \left(\frac{\partial[\mathcal{N}^{(\ell)}(f')](g')}{\partial\theta_\mu}\right)^\top\right]$$

↑ Evaluation point in group space

∞-width limit:

The Equivariant NTK

$$\Theta^{(\ell)}_{g,g'}(f, f') = \mathbb{E}\left[\sum_{\mu} \frac{\partial[\mathcal{N}^{(\ell)}(f)](g)}{\partial\theta_\mu}\left(\frac{\partial[\mathcal{N}^{(\ell)}(f')](g')}{\partial\theta_\mu}\right)^{\intercal}\right]$$

↑ Evaluation point in group space

∞**-width limit:** # channels $\to \infty$

Kernel Recursions of the Group Convolutional Layer

Kernel Recursions of the Group Convolutional Layer

$$K^{(\ell+1)}_{g,g'}(f,f') = \frac{1}{\mathrm{vol}(S_\kappa)} \int_{S_\kappa} \mathrm{d}h \; K^{(\ell)}_{gh,g'h}(f,f')$$

Kernel Recursions of the Group Convolutional Layer

$$K^{(\ell+1)}_{g,g'}(f,f') = \frac{1}{\mathrm{vol}(S_\kappa)} \int_{S_\kappa} \mathrm{d}h\, K^{(\ell)}_{gh,g'h}(f,f')$$

$$\Theta^{(\ell+1)}_{g,g'}(f,f') = K^{(\ell+1)}_{g,g'}(f,f') + \frac{1}{\mathrm{vol}(S_\kappa)} \int_{S_\kappa} \mathrm{d}h\, \Theta^{(\ell)}_{gh,g'h}(f,f')$$

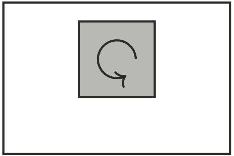
How to implement this depends on the group G and the space X.

How to implement this depends on the group G and the space X.

We cover

Roto-translations in the plane

$$G = C^4 \ltimes \mathbb{Z}^2$$
$$X = \mathbb{Z}^2$$



How to implement this depends on the group G and the space X.

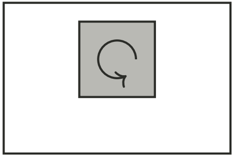
We cover

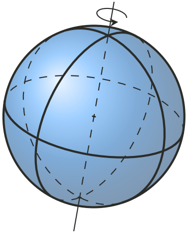
Roto-translations in the plane

$$G = C^4 \ltimes \mathbb{Z}^2$$
$$X = \mathbb{Z}^2$$

Rotations on $\mathrm{SO}(3)$

$$G = \mathrm{SO}(3)$$
$$X = S^2$$





SO(3) Implementation

$$K^{(\ell+1)}_{R,R'}(f,f') = \frac{1}{8\pi^2}\int_{\mathrm{SO}(3)} \mathrm{d}S\, K^{(\ell)}_{RS,R'S}(f,f')$$

$\mathsf{SO}(3)$ Implementation

$$K^{(\ell+1)}_{R,R'}(f,f') = \frac{1}{8\pi^2}\int_{\mathsf{SO}(3)} \mathrm{d}S\, K^{(\ell)}_{RS,R'S}(f,f')$$

How can we compute this integral efficiently?

SO(3) **Implementation**

$$K^{(\ell+1)}_{R,R'}(f,f') = \frac{1}{8\pi^2}\int_{\mathrm{SO}(3)} \mathrm{d}S\, K^{(\ell)}_{RS,R'S}(f,f')$$

How can we compute this integral efficiently?

$\rightarrow$ using the **Fourier transform** on compact groups (*Wigner transform* on SO(3))

$$\left[\widehat{K^{(\ell)}(f,f')}\right]^{l,l'}_{mn,m'n'} = \int \mathrm{d}R \int \mathrm{d}R'\, K^{(\ell)}_{R,R'}(f,f')\mathcal{D}^{l}_{mn}(R)\mathcal{D}^{l'}_{m'n'}(R')$$

SO(3) Implementation

$$K^{(\ell+1)}_{R,R'}(f,f') = \frac{1}{8\pi^2}\int_{\mathrm{SO}(3)} \mathrm{d}S\, K^{(\ell)}_{RS,R'S}(f,f')$$

How can we compute this integral efficiently?

$\rightarrow$ using the **Fourier transform** on compact groups (*Wigner transform* on SO(3))

$$\left[\widehat{K^{(\ell)}(f,f')}\right]^{l,l'}_{mn,m'n'} = \int \mathrm{d}R \int \mathrm{d}R'\, K^{(\ell)}_{R,R'}(f,f')\mathcal{D}^{l}_{mn}(R)\mathcal{D}^{l'}_{m'n'}(R')$$

- $\mathcal{D}^{l}_{mn}$ are the **Wigner D-matrices**, irreps of SO(3)

SO(3) Implementation

$$K^{(\ell+1)}_{R,R'}(f,f') = \frac{1}{8\pi^2}\int_{\mathrm{SO}(3)} \mathrm{d}S\, K^{(\ell)}_{RS,R'S}(f,f')$$

How can we compute this integral efficiently?

$\rightarrow$ using the **Fourier transform** on compact groups (*Wigner transform* on SO(3))

$$\left[\widehat{K^{(\ell)}(f,f')}\right]^{l,l'}_{mn,m'n'} = \int \mathrm{d}R \int \mathrm{d}R'\, K^{(\ell)}_{R,R'}(f,f')\mathcal{D}^{l}_{mn}(R)\mathcal{D}^{l'}_{m'n'}(R')$$

- $\mathcal{D}^{l}_{mn}$ are the **Wigner D-matrices**, irreps of SO(3)
- $l \in \mathbb{N}_0, m, n \in \{-l,\dots,l\}$

$\mathsf{SO}(3)$ Implementation

$$K^{(\ell+1)}_{R,R'}(f,f') = \frac{1}{8\pi^2}\int_{\mathsf{SO}(3)} \mathrm{d}S\, K^{(\ell)}_{RS,R'S}(f,f')$$

How can we compute this integral efficiently?

$\rightarrow$ using the **Fourier transform** on compact groups (*Wigner transform* on $\mathsf{SO}(3)$)

$$\left[\widehat{K^{(\ell)}(f,f')}\right]^{l,l'}_{mn,m'n'} = \int \mathrm{d}R \int \mathrm{d}R'\, K^{(\ell)}_{R,R'}(f,f')\mathcal{D}^{l}_{mn}(R)\mathcal{D}^{l'}_{m'n'}(R')$$

- $\mathcal{D}^{l}_{mn}$ are the **Wigner D-matrices**, irreps of $\mathsf{SO}(3)$
- $l \in \mathbb{N}_0, m, n \in \{-l, \dots, l\}$
- for the first layer $\ell = 1$, **spherical harmonics** Y^m_l are used instead of Wigner D-matrices

Fourier Recursion

Fourier Recursion

Kernel recursion in Fourier space

$$[\widehat{K^{(\ell+1)}(f, f')}]^{l,l'}_{mn,m'n'} = \frac{1}{2l+1}\delta_{ll'}\delta_{n,-n'}\sum_{p=-l}^{l}(-1)^{n-p}[\widehat{K^{\ell}(f, f')}]^{l,l'}_{mp,m'(-p)}$$

Fourier Recursion

Kernel recursion in Fourier space

$$[\widehat{K^{(\ell+1)}(f, f')}]^{l,l'}_{mn,m'n'} = \frac{1}{2l+1} \delta_{ll'} \delta_{n,-n'} \sum_{p=-l}^{l} (-1)^{n-p} [\widehat{K^{\ell}(f, f')}]^{l,l'}_{mp,m'(-p)}$$

Approximation: truncate for $l \geq L$.

Fourier Recursion

Kernel recursion in Fourier space

$$[\widehat{K^{(\ell+1)}(f,f')}]^{l,l'}_{mn,m'n'} = \frac{1}{2l+1}\delta_{ll'}\delta_{n,-n'}\sum_{p=-l}^{l}(-1)^{n-p}[\widehat{K^{\ell}(f,f')}]^{l,l'}_{mp,m'(-p)}$$

Approximation: truncate for $l \geq L$.

Straightforward to implement

Fourier Recursion

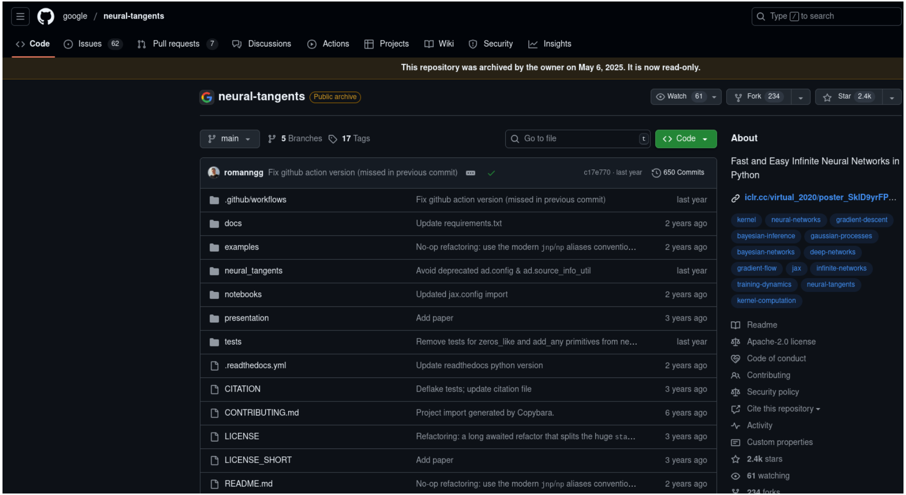
Kernel recursion in Fourier space

$$[\widehat{K^{(\ell+1)}(f,f')}]^{l,l'}_{mn,m'n'} = \frac{1}{2l+1}\delta_{ll'}\delta_{n,-n'}\sum_{p=-l}^{l}(-1)^{n-p}[\widehat{K^{\ell}(f,f')}]^{l,l'}_{mp,m'(-p)}$$

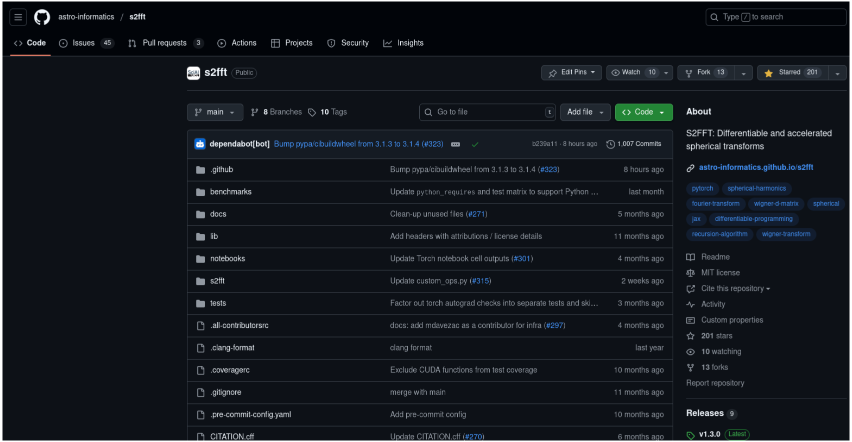
Approximation: truncate for $l \geq L$.

Straightforward to implement . . . *right?*

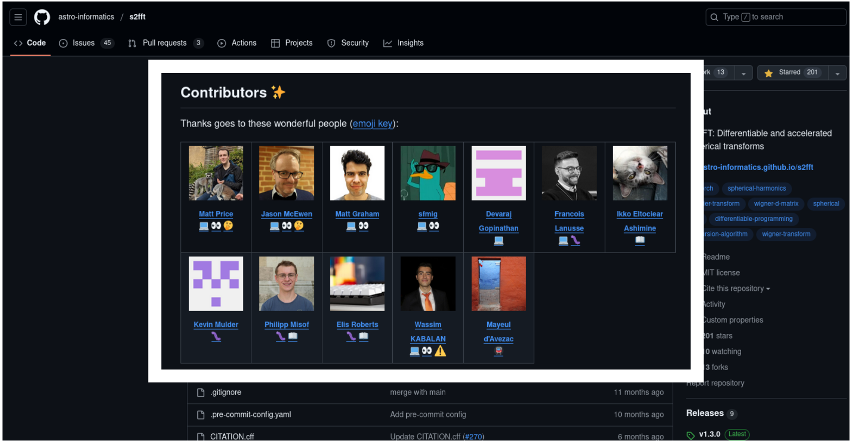
Goal: Integrate it in the `neural-tangents` library (written in `JAX`).



Fortunately, **Fast Fourier Transforms** (FFT) on $\mathrm{SO}(3)$ and S^2 provided by `s2fft`.



Fortunately, **Fast Fourier Transforms** (FFT) on $\mathrm{SO}(3)$ and S^2 provided by `s2fft`.

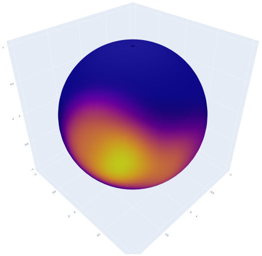


Testing the $\mathrm{SO}(3)$ NTK on molecular data (QM9)

Testing the $\mathrm{SO}(3)$ NTK on molecular data (QM9)

Atoms' environments are represented as signals on the sphere *(Esteves, Slotine, and Makadia 2023)*

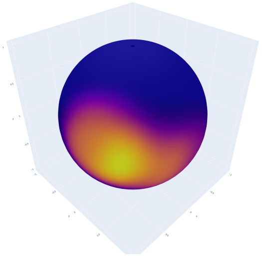
$$f_{i,z,p}(x) = \sum_{j:z_j=z} \frac{z_i z}{||r_{ij}||^p} e^{-\frac{1}{\beta}\left(\frac{r_{ij}}{||r_{ij}||}\cdot x - 1\right)^2}$$

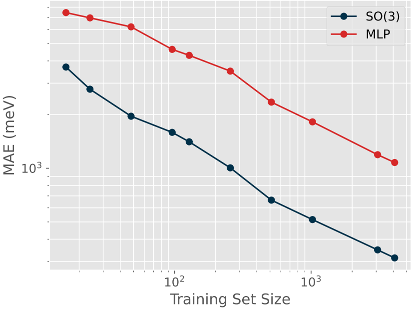


Testing the $\mathrm{SO}(3)$ NTK on molecular data (QM9)

Atoms' environments are represented as signals on the sphere *(Esteves, Slotine, and Makadia 2023)*

$$f_{i,z,p}(x) = \sum_{j:z_j=z} \frac{z_j z}{\|r_{ij}\|^p} e^{-\frac{1}{\beta}\left(\frac{r_{ij}}{\|r_{ij}\|}\cdot x - 1\right)^2}$$

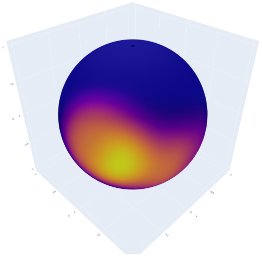


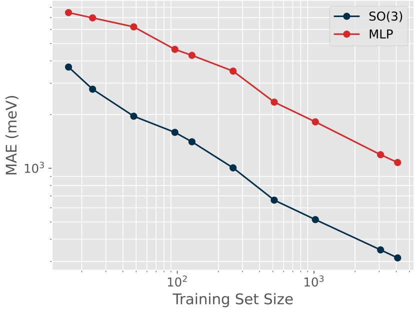


Testing the $\mathrm{SO}(3)$ NTK on molecular data (QM9)

Atoms' environments are represented as signals on the sphere *(Esteves, Slotine, and Makadia 2023)*

$$f_{i,z,p}(x) = \sum_{j:z_j=z} \frac{z_j z}{\|r_{ij}\|^p} e^{-\frac{1}{\beta}\left(\frac{r_{ij}}{\|r_{ij}\|}\cdot x - 1\right)^2}$$





➡ Performance boost due to 3d-rotation invariance extends to the ∞-width limit

Often, equivariance is not enforced but learned approximately through **data augmentation**

Often, equivariance is not enforced but learned approximately through **data augmentation**

Can we **compare the two approaches** theoretically?

NTK under data augmentation

NTK under data augmentation

- Full data augmentation

$$\mathcal{D}^{\text{aug}} = \bigcup_{i=1}^{n_{\text{train}}} \bigcup_{g \in G} \{(\rho_{\text{reg}}(g) f_i, \tilde{\rho}_{\text{reg}}(g) y_i)\},$$

NTK under data augmentation

- Full data augmentation

$$\mathcal{D}^{\text{aug}} = \bigcup_{i=1}^{n_{\text{train}}} \bigcup_{g \in G} \{(\rho_{\text{reg}}(g) f_i, \tilde{\rho}_{\text{reg}}(g) y_i)\},$$

- Invariance $\quad \leftrightarrow \quad \tilde{\rho}_{\text{reg}} = \text{id}$

NTK under **data augmentation**

- Full data augmentation

$$\mathcal{D}^{\text{aug}} = \bigcup_{i=1}^{n_{\text{train}}} \bigcup_{g \in G} \{(\rho_{\text{reg}}(g) f_i, \tilde{\rho}_{\text{reg}}(g) y_i)\},$$

- Invariance $\quad \leftrightarrow \quad \tilde{\rho}_{\text{reg}} = \text{id}$
- μ_t^{aug} evolves like a non-augmented NN mean μ_t with NTK

$$\Theta(f, f') = \frac{1}{|G|} \sum_{g \in G} \Theta^{\text{aug}}(f, \rho_{\text{reg}}(g) f')$$

Data Augmentation $\leftrightarrow$ Group Convolutional (GC) NNs

Data Augmentation $\leftrightarrow$ Group Convolutional (GC) NNs

For a given MLP, we can **construct a GCNN** s.t.

$$\Theta^{\mathrm{GC}}(f, f') = \frac{1}{\mathrm{vol}(G)} \int_G \mathrm{d}g\; \Theta^{\mathrm{MLP}}(f, \rho_{\mathrm{reg}}(g) f')$$

Data Augmentation $\leftrightarrow$ Group Convolutional (GC) NNs

For a given MLP, we can **construct a GCNN** s.t.

$$\Theta^{\mathrm{GC}}(f, f') = \underbrace{\frac{1}{\mathrm{vol}(G)} \int_G \mathrm{d}g \; \Theta^{\mathrm{MLP}}(f, \rho_{\mathrm{reg}}(g) f')}_{\substack{\text{same effective kernel} \\ \text{resulting from data augmentation}}}$$

Data Augmentation $\leftrightarrow$ Group Convolutional (GC) NNs

For a given MLP, we can **construct a GCNN** s.t.

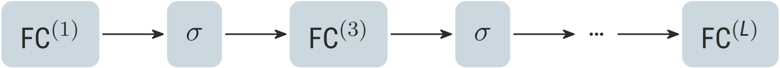
$$\Theta^{\text{GC}}(f, f') = \underbrace{\frac{1}{\text{vol}(G)} \int_G \text{d}g \; \Theta^{\text{MLP}}(f, \rho_{\text{reg}}(g) f')}_{\substack{\text{same effective kernel} \\ \text{resulting from data augmentation}}}$$

At ∞-width and quadratic $\mathcal{L}$:

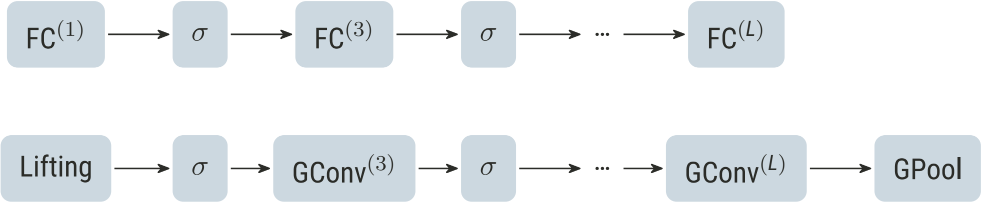
➡ **Expectation** of a data augmented MLP equals the **expectation** of an GCNN **at all training times** t.

Architecture correspondence

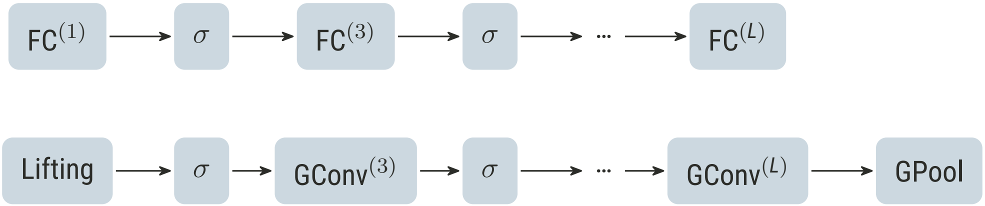
Architecture correspondence



Architecture correspondence



Architecture correspondence



All group convolutions with global filter support $S_\kappa^\ell = G$ or $S_\kappa^1 = X$ for the lifting layer.

Data Augmentation vs. Group Convolutions at finite width

Data Augmentation vs. Group Convolutions at finite width

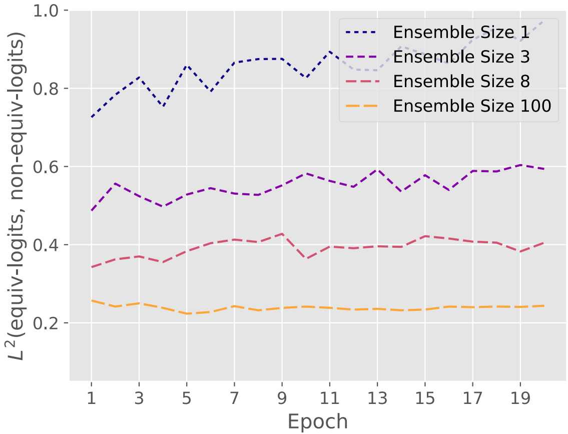
- Data augmented CNN vs $C_4 \ltimes \mathbb{R}^2$ GCNN on **MNIST**

Data Augmentation vs. Group Convolutions at finite width

- Data augmented CNN vs $C_4 \ltimes \mathbb{R}^2$ GCNN on **MNIST**
- Compare L_2-difference of averaged outputs

Data Augmentation vs. Group Convolutions at finite width

- Data augmented CNN vs $C_4 \ltimes \mathbb{R}^2$ GCNN on **MNIST**
- Compare L_2-difference of averaged outputs



Taking $n \to \infty$ allowed us to to derive exact analytic results for the training dynamics of NNs.

Taking $n \to \infty$ allowed us to to derive exact analytic results for the training dynamics of NNs.

But it also **introduces artifacts** compared to finite width NNs:

Taking $n \to \infty$ allowed us to to derive exact analytic results for the training dynamics of NNs.

But it also **introduces artifacts** compared to finite width NNs:

- Kernel methods and Gaussian processes in general underperform compared to NNs

Taking $n \to \infty$ allowed us to to derive exact analytic results for the training dynamics of NNs.

But it also **introduces artifacts** compared to finite width NNs:

- Kernel methods and Gaussian processes in general underperform compared to NNs
- The empirical NTK changes during training

Taking $n \to \infty$ allowed us to to derive exact analytic results for the training dynamics of NNs.

But it also **introduces artifacts** compared to finite width NNs:

- Kernel methods and Gaussian processes in general underperform compared to NNs
- The empirical NTK changes during training
- **No feature learning:**

$$\Delta \mathcal{N}^{(\ell)}(x) = \mathcal{O}\left(\frac{1}{n}\right) \quad \text{for } \ell < L$$

Taking $n \to \infty$ allowed us to to derive exact analytic results for the training dynamics of NNs.

But it also **introduces artifacts** compared to finite width NNs:

- Kernel methods and Gaussian processes in general underperform compared to NNs
- The empirical NTK changes during training
- **No feature learning:**

$$\Delta \mathcal{N}^{(\ell)}(x) = \mathcal{O}\left(\frac{1}{n}\right) \quad \text{for } \ell < L$$

Idea: Derive finite-width corrections with techniques from **quantum field theory** (QFT)

Idea: Derive finite-width corrections with techniques from **quantum field theory** (QFT)

IOP Publishing *Mach. Learn.: Sci. Technol.* **2** (2021) 035002 https://doi.org/10.1088/2632-2153/abeca3

MACHINE LEARNING Science and Technology

CrossMark

OPEN ACCESS

RECEIVED
22 August 2020

REVISED
8 February 2021

ACCEPTED FOR PUBLICATION
8 March 2021

PUBLISHED
27 April 2021

PAPER

Neural networks and quantum field theory

James Halverson, Anindita Maiti* and Keegan Stoner

Department of Physics, Northeastern University, Boston, MA 02115, United States of America

* Author to whom any correspondence should be addressed.

E-mail: maiti.a@northeastern.edu

Keywords: Wilsonian RG flow, infinite width NNGP, finite width NN, Gaussian processes, quantum field theory

Abstract

We propose a theoretical understanding of neural networks in terms of Wilsonian effective field theory. The correspondence relies on the fact that many asymptotic neural networks are drawn from Gaussian processes (GPs), the analog of non-interacting field theories. Moving away from the asymptotic limit yields a non-Gaussian process (NGP) and corresponds to turning on particle interactions, allowing for the computation of correlation functions of neural network outputs with

Idea: Derive finite-width corrections with techniques from **quantum field theory** (QFT)

IOP Publishing   *Mach. Learn.: Sci. Technol.* **2** (2021) 035002   https://doi.org/10.1088/2632-2153/abeca3

MACHINE LEARNING
Science and Technology

CrossMark

OPEN ACCESS

RECEIVED
22 August 2020

REVISED
8 February 2021

ACCEPTED FOR PUBLICATION
8 March 2021

PUBLISHED
27 April 2021

PAPER

Neural networks and quantum field theory

James Halverson, Anindita Maiti* and Keegan Stoner

Department of Physics, Northeastern University, Boston, MA 02115, United States of America
* Author to whom any correspondence should be addressed.

E-mail: maiti.a@northeastern.edu

Keywords: Wilsonian RG flow, infinite width NNGP, finite width NN, Gaussian processes, quantum field theory

Abstract

We propose a theoretical understanding of neural networks in terms of Wilsonian effective field theory. The correspondence relies on the fact that many asymptotic neural networks are drawn from Gaussian processes (GPs), the analog of non-interacting field theories. Moving away from the asymptotic limit yields a non-Gaussian process (NGP) and corresponds to turning on particle interactions, allowing for the computation of correlation functions of neural network outputs with

The Principles of Deep Learning Theory

An Effective Theory Approach
to Understanding Neural Networks

DANIEL A. ROBERTS
MIT

SHO YAIDA
Meta AI

based on research in collaboration with

BORIS HANIN
Princeton University

Idea: Derive finite-width corrections with techniques from **quantum field theory** (QFT)

IOP Publishing *Mach. Learn.: Sci. Technol.* **2** (2021) 035002 https://doi.org/10.1088/2632-2153/abeca3

MACHINE LEARNING
Science and Technology

CrossMark

OPEN ACCESS

RECEIVED
22 August 2020

REVISED
8 February 2021

ACCEPTED FOR PUBLICATION
8 March 2021

PUBLISHED
27 April 2021

Original Content from this work may be used under the terms of the Creative Commons Attribution 4.0 licence.

Any further distribution of this work must

PAPER

Neural networks and quantum field theory

James Halverson, Anindita Maiti* and Keegan Stoner
Department of Physics, Northeastern University, Boston, MA 02115, United States of America
* Author to whom any correspondence should be addressed.

E-mail: maiti.a@northeastern.edu

Keywords: Wilsonian RG flow, infinite width NNGP, finite width NN, Gaussian processes, quantum field theory

Abstract
We propose a theoretical understanding of neural networks in terms of Wilsonian effective field theory. The correspondence relies on the fact that many asymptotic neural networks are drawn from Gaussian processes (GPs), the analog of non-interacting field theories. Moving away from the asymptotic limit yields a non-Gaussian process (NGP) and corresponds to turning on particle interactions, allowing for the computation of correlation functions of neural network outputs with

The Principles of Deep Learning Theory

An Effective Theory Approach
to Understanding Neural Networks

DANIEL A. ROBERTS
MIT

SHO YAIDA
Meta AI

based on research in collaboration with

BORIS HANIN
Princeton University

The Setup

We now focus on **preactivations**

$$z_i^{(\ell)}(x) = \frac{1}{\sqrt{n_{\ell-1}}} \sum_{j=1}^{n_{\ell-1}} W_{ij}^{(\ell)} \underbrace{\sigma(z_j^{(\ell-1)}(x))}_{\mathcal{N}^{(\ell-1)} \text{ before}} + b_i^{(\ell)}$$

The Setup

We now focus on **preactivations**

$$z_i^{(\ell)}(x) = \frac{1}{\sqrt{n_{\ell-1}}} \sum_{j=1}^{n_{\ell-1}} W_{ij}^{(\ell)} \underbrace{\sigma(z_j^{(\ell-1)}(x))}_{\mathcal{N}^{(\ell-1)} \text{ before}} + b_i^{(\ell)}$$

We are now interested in the **distribution** of the output **at initialization**

$$p\left(z^{(L)} | \mathcal{D}\right)$$

The Setup

We now focus on **preactivations**

$$z_i^{(\ell)}(x) = \frac{1}{\sqrt{n_{\ell-1}}} \sum_{j=1}^{n_{\ell-1}} W_{ij}^{(\ell)} \underbrace{\sigma(z_j^{(\ell-1)}(x))}_{\mathcal{N}^{(\ell-1)} \text{ before}} + b_i^{(\ell)}$$

We are now interested in the **distribution** of the output **at initialization**

$$p\left(z^{(L)} | \mathcal{D}\right)$$

Decompose it **layer by layer**

Notation: $z_{i;\alpha}^{(\ell)} = z_i^{(\ell)}(x_\alpha)$

$$p\left(z^{(\ell+1)} | \mathcal{D}\right) = \int \prod_{i,\alpha} dz_{i;\alpha}^{(\ell)} \underbrace{p(z^{(\ell+1)} | z^{(\ell)})}_{\text{Normal dist.}} p(z^{(\ell)} | \mathcal{D})$$

$$p\left(z^{(\ell+1)}|\mathcal{D}\right) = \int \prod_{i,\alpha} \mathrm{d}z_{i;\alpha}^{(\ell)}\, p(z^{(\ell+1)}|z^{(\ell)})p(z^{(\ell)}|\mathcal{D})$$

$$p\left(z^{(\ell+1)}|\mathcal{D}\right) = \int \prod_{i,\alpha} \mathrm{d}z_{i;\alpha}^{(\ell)}\, p(z^{(\ell+1)}|z^{(\ell)})p(z^{(\ell)}|\mathcal{D})$$

We know that $p(z^{(\ell)}|\mathcal{D})$ **becomes a Normal distribution** as $n \to \infty$.

$$p\left(z^{(\ell+1)}|\mathcal{D}\right) = \int \prod_{i,\alpha} \mathrm{d}z_{i;\alpha}^{(\ell)}\, p(z^{(\ell+1)}|z^{(\ell)})p(z^{(\ell)}|\mathcal{D})$$

We know that $p(z^{(\ell)}|\mathcal{D})$ **becomes a Normal distribution** as $n \to \infty$.

Instead of a Normal distribution with probability density function

$$p(z) = \frac{1}{Z}\exp\left(-\frac{1}{2}z^{\mathsf{T}}K^{-1}z\right) = e^{-S[z]}$$

$$p\left(z^{(\ell+1)}|\mathcal{D}\right) = \int \prod_{i,\alpha} \mathrm{d}z_{i;\alpha}^{(\ell)}\, p(z^{(\ell+1)}|z^{(\ell)})p(z^{(\ell)}|\mathcal{D})$$

We know that $p(z^{(\ell)}|\mathcal{D})$ **becomes a Normal distribution** as $n \to \infty$.

Instead of a Normal distribution with probability density function

$$p(z) = \frac{1}{Z}\exp\left(-\frac{1}{2}z^{\mathsf{T}}K^{-1}z\right) = e^{-S[z]}$$

make an **ansatz** for the *action* (Roberts, Yaida, and Hanin 2022)

$$S[z] = \frac{1}{2}\sum_{\alpha_1,\alpha_2 \in D} g^{\alpha_1\alpha_2}\sum_{i=1}^{n} z_{i;\alpha_1}z_{i;\alpha_2}$$

$$-\frac{1}{2}\sum_{\alpha_1,\ldots,\alpha_4 \in D} v^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}\sum_{i_1,i_2=1}^{n} z_{i_1;\alpha_1}z_{i_1;\alpha_2}z_{i_2;\alpha_3}z_{i_2;\alpha_4}$$

$$p\left(z^{(\ell+1)}|\mathcal{D}\right) = \int \prod_{i,\alpha} \mathrm{d}z_{i;\alpha}^{(\ell)}\, p(z^{(\ell+1)}|z^{(\ell)})p(z^{(\ell)}|\mathcal{D})$$

We know that $p(z^{(\ell)}|\mathcal{D})$ **becomes a Normal distribution** as $n \to \infty$.

Instead of a Normal distribution with probability density function

$$p(z) = \frac{1}{Z}\exp\left(-\frac{1}{2}z^{\mathsf{T}}K^{-1}z\right) = e^{-S[z]}$$

make an **ansatz** for the *action* (Roberts, Yaida, and Hanin 2022)

$$S[z] = \frac{1}{2}\sum_{\alpha_1,\alpha_2\in D} g^{\alpha_1\alpha_2}\sum_{i=1}^{n} z_{i;\alpha_1}z_{i;\alpha_2}$$
$$-\frac{1}{2}\sum_{\alpha_1,\ldots,\alpha_4\in D} v^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)}\sum_{i_1,i_2=1}^{n} z_{i_1;\alpha_1}z_{i_1;\alpha_2}z_{i_2;\alpha_3}z_{i_2;\alpha_4}$$

Characterizing the Distribution Through its Cumulants

Characterizing the Distribution Through its Cumulants

Using the ansatz, one can compare coefficients with cumulants to first order in $1/n$:

Characterizing the Distribution Through its Cumulants

Using the ansatz, one can compare coefficients with cumulants to first order in $1/n$:

$$g^{\alpha_1\alpha_2} = K^{\alpha_1\alpha_2} + \mathcal{O}\left(\frac{1}{n}\right)$$

$$v^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} = \frac{1}{n} V^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} + \mathcal{O}\left(\frac{1}{n^2}\right)$$

Characterizing the Distribution Through its Cumulants

Using the ansatz, one can compare coefficients with cumulants to first order in $1/n$:

$$g^{\alpha_1\alpha_2} = K^{\alpha_1\alpha_2} + \mathcal{O}\left(\frac{1}{n}\right)$$

$$v^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} = \frac{1}{n} V^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} + \mathcal{O}\left(\frac{1}{n^2}\right)$$

where $V^{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} = \mathbb{E}^{\mathrm{c}}[z_{\alpha_1}, z_{\alpha_2}, z_{\alpha_3}, z_{\alpha_4}]$ is the **4th cumulant**.

Statistics Described by Recursion System

- Analysis can be extended to joint distribution

$$p(z^{(\ell)}, \Theta^{(\ell)} | \mathcal{D})$$

Statistics Described by Recursion System

- Analysis can be extended to joint distribution

$$p(z^{(\ell)}, \Theta^{(\ell)} | \mathcal{D})$$

- At order $\mathcal{O}(\frac{1}{n})$, fully characterized by a **closed system of recursions** containing K, Θ and V_4 as well as joint cumulants A, B, D, F of degree 4.

Statistics Described by Recursion System

- Analysis can be extended to joint distribution

$$p(z^{(\ell)}, \Theta^{(\ell)} | \mathcal{D})$$

- At order $\mathcal{O}(\frac{1}{n})$, fully characterized by a **closed system of recursions** containing K, Θ and V_4 as well as joint cumulants A, B, D, F of degree 4.

$$K^{(\ell)}, V_4^{(\ell)}, A^{(\ell)}, B^{(\ell)}, D^{(\ell)}, F^{(\ell)}$$
$$\longrightarrow K^{(\ell+1)}, \Theta^{(\ell)}, V_4^{(\ell+1)}, A^{(\ell+1)}, B^{(\ell+1)}, D^{(\ell+1)}, F^{(\ell+1)} + \mathcal{O}\left(\frac{1}{n^2}\right)$$

Recursions

- have been used to find optimal initialization hyperparameters (**Criticality**)

Recursions

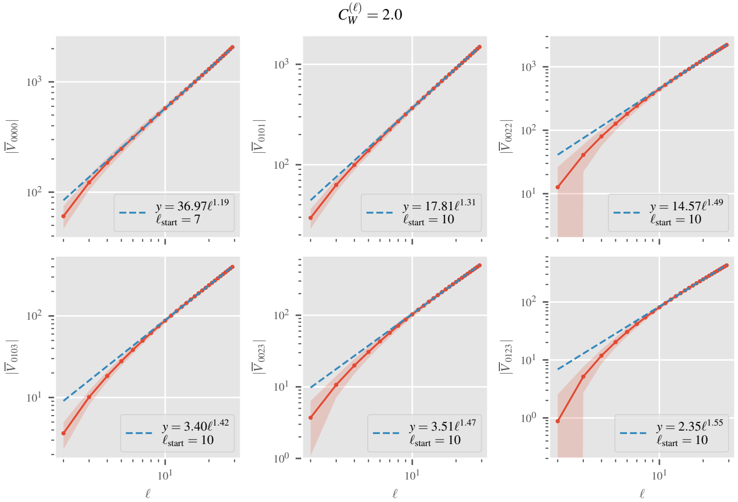
- have been used to find optimal initialization hyperparameters (**Criticality**)
- Explain qualitative **differences between activation functions**

Recursions

- have been used to find optimal initialization hyperparameters (**Criticality**)
- Explain qualitative **differences between activation functions**
- Explain **Exploding and Vanishing Gradients**

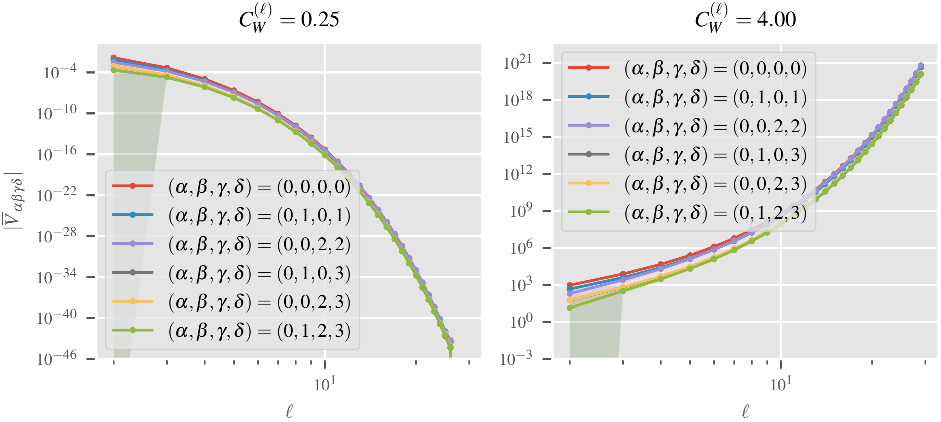
Empirical V_4 evolution at criticality

for a ReLU network



Empirical V_4 evolution away from criticality

for a ReLU network



Recursions seem very useful.

Recursions seem very useful.

But: They are tedious to derive

Recursions seem very useful.

But: They are tedious to derive

In QFT

This is done with **Feynman diagrams**

Recursions seem very useful.

But: They are tedious to derive

In QFT

This is done with **Feynman diagrams**

- **Diagrammatic representation** of algebraic expressions

Recursions seem very useful.

But: They are tedious to derive

In QFT

This is done with **Feynman diagrams**

- **Diagrammatic representation** of algebraic expressions
- Careful rules specify what diagrams are allowed

Recursions seem very useful.

But: They are tedious to derive

In QFT

This is done with **Feynman diagrams**

- **Diagrammatic representation** of algebraic expressions
- Careful rules specify what diagrams are allowed
- Each diagram corresponds to a term at a certain order

Finite-Width Neural Tangent Kernels from Feynman Diagrams

Max Guillen*[a] Philipp Misof *[a] Jan E. Gerken[a]

Abstract

Neural tangent kernels (NTKs) are a powerful tool for analyzing deep, non-linear neural networks. In the infinite-width limit, NTKs can easily be computed for most common architectures, yielding full analytic control over the training dynamics. However, at infinite width, important properties of training such as NTK evolution or feature learning are absent. Nevertheless, finite width effects can be included by computing corrections to the Gaussian statistics at infinite width. We introduce Feynman diagrams for computing finite-width corrections to NTK statistics. These dramatically simplify the

Recursions from Feynman diagrams

Already existed for preactivation recursions *(Banta et al. 2024)*

Recursions from Feynman diagrams

Already existed for preactivation recursions *(Banta et al. 2024)*

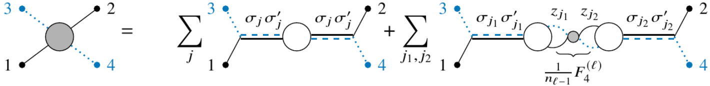
We **extend this to joint preactivation-NTK statistics**

Recursions from Feynman diagrams

Already existed for preactivation recursions *(Banta et al. 2024)*

We **extend this to joint preactivation-NTK statistics**

Example: F **recursion**

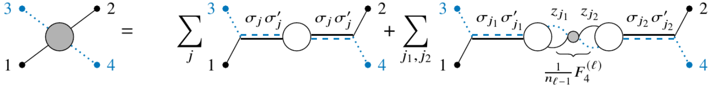


Recursions from Feynman diagrams

Already existed for preactivation recursions *(Banta et al. 2024)*

We **extend this to joint preactivation-NTK statistics**

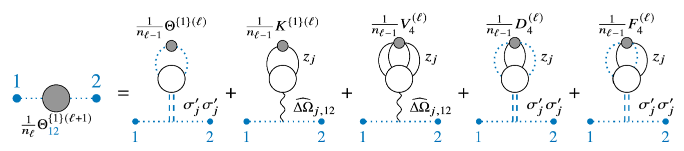
Example: F **recursion**



Generalization to higher orders follows the same principles.

Recursions from Feynman diagrams

New Recursion: First order correction $\Theta^{\{1\}(\ell)}$ to the infinite width NTK $\Theta^{\{0\}(\ell)}$



Solving the V_4 recursion [WIP]

Solving the V_4 recursion [WIP]

$$\frac{1}{n_\ell} V^{(\ell+1)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} = \frac{1}{n_\ell}\left(C_W^{(\ell+1)}\right)^2 \left[\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\sigma_{\alpha_3}\sigma_{\alpha_4}\rangle_{G^{(\ell)}} - \langle \sigma_{\alpha_1}\sigma_{\alpha_2}\rangle_{G^{(\ell)}} \langle \sigma_{\alpha_3}\sigma_{\alpha_4}\rangle_{G^{(\ell)}}\right]$$

$$+ \frac{1}{n_{\ell-1}} \frac{\left(C_W^{(\ell+1)}\right)^2}{4} \sum_{\beta_1,\dots,\beta_4 \in D} V^{(\beta_1\beta_2)(\beta_3\beta_4)}_{(\ell)} \langle \sigma_{\alpha_1}\sigma_{\alpha_2}(z_{\beta_1}z_{\beta_2} - g_{\beta_1\beta_2})\rangle_{G^{(\ell)}}$$

$$\times \langle \sigma_{\alpha_3}\sigma_{\alpha_4}(z_{\beta_3}z_{\beta_4} - g_{\beta_3\beta_4})\rangle_{G^{(\ell)}} + \mathcal{O}\left(\frac{1}{n^2}\right)$$

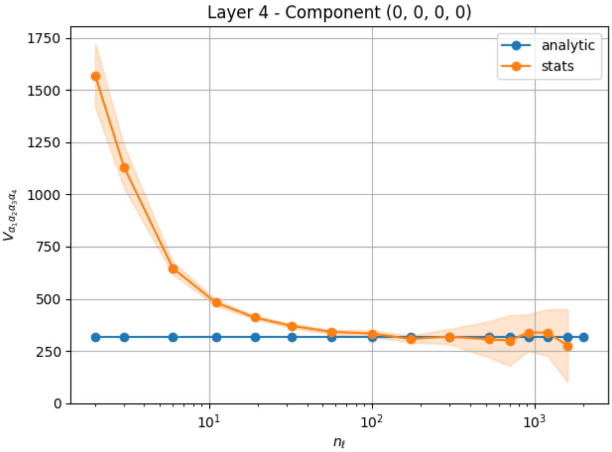
Solving the V_4 recursion [WIP]

$$\frac{1}{n_\ell} V^{(\ell+1)}_{(\alpha_1\alpha_2)(\alpha_3\alpha_4)} = \frac{1}{n_\ell}\left(C_W^{(\ell+1)}\right)^2 \left[\langle \sigma_{\alpha_1}\sigma_{\alpha_2}\sigma_{\alpha_3}\sigma_{\alpha_4}\rangle_{G^{(\ell)}} - \langle \sigma_{\alpha_1}\sigma_{\alpha_2}\rangle_{G^{(\ell)}} \langle \sigma_{\alpha_3}\sigma_{\alpha_4}\rangle_{G^{(\ell)}}\right]$$

$$+ \frac{1}{n_{\ell-1}} \frac{\left(C_W^{(\ell+1)}\right)^2}{4} \sum_{\beta_1,\ldots,\beta_4 \in D} V^{(\beta_1\beta_2)(\beta_3\beta_4)}_{(\ell)} \langle \sigma_{\alpha_1}\sigma_{\alpha_2}(z_{\beta_1}z_{\beta_2} - g_{\beta_1\beta_2})\rangle_{G^{(\ell)}}$$

$$\times \langle \sigma_{\alpha_3}\sigma_{\alpha_4}(z_{\beta_3}z_{\beta_4} - g_{\beta_3\beta_4})\rangle_{G^{(\ell)}} + \mathcal{O}\left(\frac{1}{n^2}\right)$$

Solving the V_4 recursion [WIP]



Combining symbolic and numeric computations

- Most integrals can be reduced to $2d$ Gaussian integrals using **integration by parts** (IBP).

Combining symbolic and numeric computations

- Most integrals can be reduced to $2d$ Gaussian integrals using **integration by parts** (IBP).
- **Numerically cheaper**

Combining symbolic and numeric computations

- Most integrals can be reduced to $2d$ Gaussian integrals using **integration by parts** (IBP).
- **Numerically cheaper**

Combining symbolic and numeric computations

- Most integrals can be reduced to 2*d* Gaussian integrals using **integration by parts** (IBP).
- **Numerically cheaper**, but **number of terms explodes** fast.
- **Solution:** Do IBP symbolically and create numeric functions from that.

```
GaussExpec(sig(z[a1])*sig(z[a2]))*K[b1, b3]*K[b2, b4] +GaussExpec(sig(z[a1])*sig(z[a2]))*K[b1,
b4]*K[b2, b3] +GaussExpec(Derivative(sig(z[a1]), z[a1])*Derivative(sig(z[a2]), z[a2]))*K[b1,
a1]*K[b2, b3]*K[b4, a2] +GaussExpec(Derivative(sig(z[a1]), z[a1])*Derivative(sig(z[a2]),
z[a2]))*K[b1, a1]*K[b2, b4]*K[b3, a2] +GaussExpec(Derivative(sig(z[a1]), z[a1])*Derivative(sig(z[a2]),
z[a2]))*K[b1, a2]*K[b2, b3]*K[b4, a1] +GaussExpec(Derivative(sig(z[a1]), z[a1])*Derivative(sig(z[a2]),
z[a2]))*K[b1, a2]*K[b2, b4]*K[b3, a1] +GaussExpec(Derivative(sig(z[a1]), z[a1])*Derivative(sig(z[a2]),
z[a2]))*K[b1, b3]*K[b2, a1]*K[b4, a2] +GaussExpec(Derivative(sig(z[a1]), z[a1])*Derivative(sig(z[a2]),
z[a2]))*K[b1, b3]*K[b2, a2]*K[b4, a1] +GaussExpec(Derivative(sig(z[a1]), z[a1])*Derivative(sig(z[a2]),
z[a2]))*K[b1, b4]*K[b2, a1]*K[b3, a2] +GaussExpec(Derivative(sig(z[a1]), z[a1])*Derivative(sig(z[a2]),
z[a2]))*K[b1, b4]*K[b2, a2]*K[b3, a1] +GaussExpec(sig(z[a2])*Derivative(sig(z[a1]), (z[a1],
2)))*K[b1, a1]*K[b2, b3]*K[b4, a1] +GaussExpec(sig(z[a2])*Derivative(sig(z[a1]), (z[a1], 2)))*K[b1,
a1]*K[b2, b4]*K[b3, a1] +GaussExpec(sig(z[a2])*Derivative(sig(z[a1]), (z[a1], 2)))*K[b1, b3]*K[b2,
a1]*K[b4, a1]
...
```

Summary

Summary

- NTK is a **valuable tool** to study NNs analytically

Summary

- NTK is a **valuable tool** to study NNs analytically
- We extended it to **equivariant NNs**

Summary

- NTK is a **valuable tool** to study NNs analytically
- We extended it to **equivariant NNs**
- Can be used to study **equivariance vs data augmentation**

Summary

- NTK is a **valuable tool** to study NNs analytically
- We extended it to **equivariant NNs**
- Can be used to study **equivariance vs data augmentation**
- Introduced **diagrammatic framework** for finite width NTK statistics

Summary

- NTK is a **valuable tool** to study NNs analytically
- We extended it to **equivariant NNs**
- Can be used to study **equivariance vs data augmentation**
- Introduced **diagrammatic framework** for finite width NTK statistics
- We implement solutions to the governing recursions

What’s next?

What's next?

- Finite width corrections for **orthogonal weights**

What's next?

- Finite width corrections for **orthogonal weights**
- Connection to other limits, e.g. the **mean field limit**

What's next?

- Finite width corrections for **orthogonal weights**
- Connection to other limits, e.g. the **mean field limit**

But first

What's next?

- Finite width corrections for **orthogonal weights**
- Connection to other limits, e.g. the **mean field limit**

But first

- **Internship** in Switzerland at **Genentech (Roche)** with Pan Kessel
- 10 months
- About **generative models** for protein design

Genentech

Thank you for the last two years!