

# 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

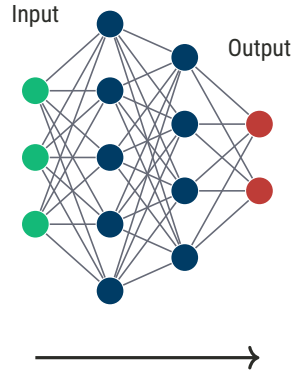
- 1 The Neural Tangent Kernel
- 2 Equivariance and Data Augmentation
- 3 Beyond the strict limit with Feynman diagrams
- 4 Conclusion and Outlook

# Feedforward **Neural Network** (NN)

*alias Multi-layer Perceptron (MLP)*

# Feedforward Neural Network (NN)

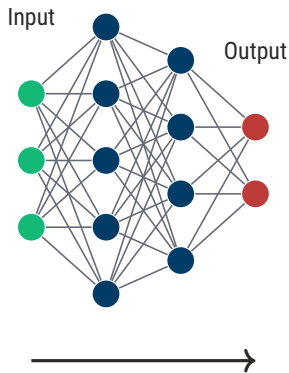
*alias Multi-layer Perceptron (MLP)*



# Feedforward Neural Network (NN)

alias *Multi-layer Perceptron* (MLP)

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



# Feedforward Neural Network (NN)

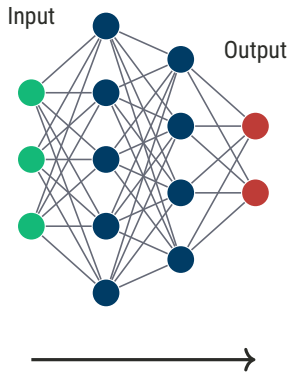
alias *Multi-layer Perceptron* (MLP)

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

Activation function

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

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



# Feedforward Neural Network (NN)

alias *Multi-layer Perceptron* (MLP)

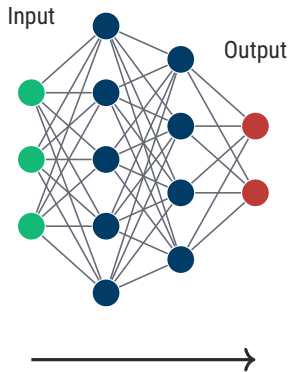
- Is a map  $\mathcal{N}^{(L)} : \mathbb{R}^d \rightarrow \mathbb{R}^{n_L}$ .
- Recursively defined via **layers**  $\mathcal{N}^{(\ell)}$

Activation function

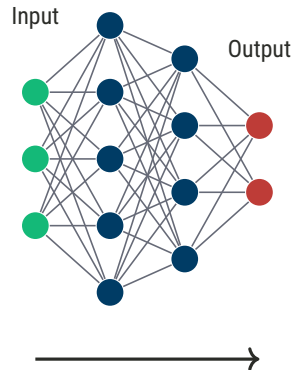
$$\mathcal{N}^{(\ell)}(x) = \sigma \left( \underbrace{\frac{1}{\sqrt{n_{\ell-1}}}}_{\text{weights}} W^{(\ell)} \underbrace{\mathcal{N}^{(\ell-1)}(x)}_{\text{Input}} + \underbrace{b^{(\ell)}}_{\text{biases}} \right),$$

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

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



# Initialization

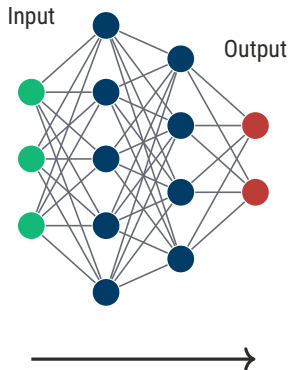




# Initialization

parameters sampled **iid**

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



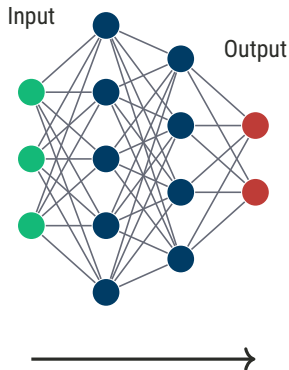
# Initialization

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

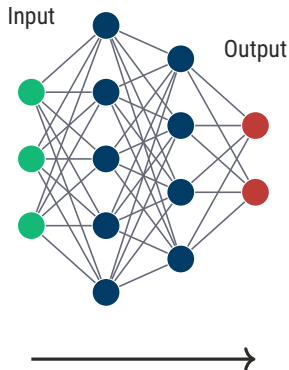
parameters sampled **iid**

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

## 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**

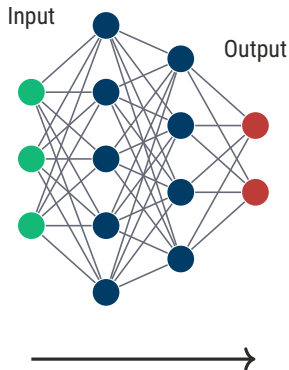
$$W_{ij}^{(\ell)}, b_i^\ell \sim \mathcal{N}(0, 1)$$

## 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**

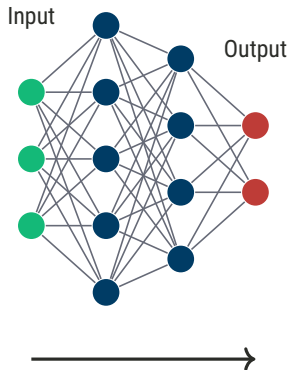
$$W_{ij}^{(\ell)}, b_i^\ell \sim \mathcal{N}(0, 1)$$

## 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**

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

Learning rate

# Training Dynamics

Assume **Gradient Flow**

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

Learning rate

Chain rule  $\rightarrow$

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

# Training Dynamics

Assume **Gradient Flow**

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

Learning rate

Chain rule  $\rightarrow$

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

(empirical) **Neural Tangent Kernel**



# Training Dynamics

Assume **Gradient Flow**

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

Learning rate

$$\Theta_t(x, x') = \sum_{\mu} \frac{\partial \mathcal{N}(x)}{\partial \theta_{\mu}} \left( \frac{\partial \mathcal{N}(x')}{\partial \theta_{\mu}} \right)^{\top}$$

Chain rule  $\rightarrow$

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

(empirical) **Neural Tangent Kernel**

# Training Dynamics

Assume **Gradient Flow**

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

Learning rate

Chain rule  $\rightarrow$

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

(empirical) **Neural Tangent Kernel**

$$\Theta_t(x, x') = \sum_{\mu} \frac{\partial \mathcal{N}(x)}{\partial \theta_{\mu}} \left( \frac{\partial \mathcal{N}(x')}{\partial \theta_{\mu}} \right)^{\top}$$

**Intuition:** Similarity measure of gradients at different inputs

# Training Dynamics

Assume **Gradient Flow**

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

Learning rate

Chain rule  $\rightarrow$

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

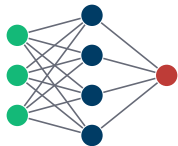
(empirical) **Neural Tangent Kernel**

$$\Theta_t(x, x') = \sum_{\mu} \frac{\partial \mathcal{N}(x)}{\partial \theta_{\mu}} \left( \frac{\partial \mathcal{N}(x')}{\partial \theta_{\mu}} \right)^{\top}$$

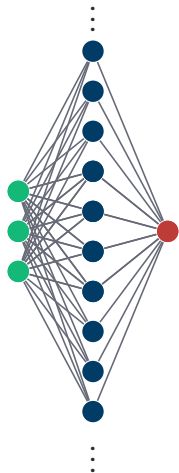
**Intuition:** Similarity measure of gradients at different inputs

$\Theta_t$  is **time-dependent** and **stochastic**.

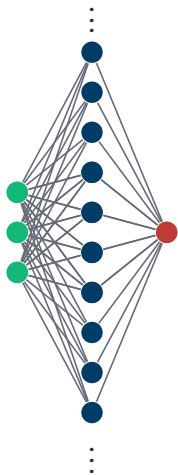
What happens when we make the hidden layers **very wide**?



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



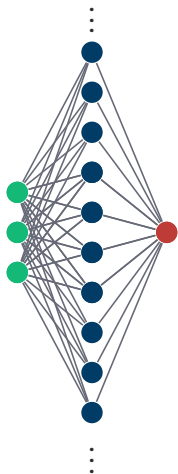
What happens when we make the hidden layers **very wide**?



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

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

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

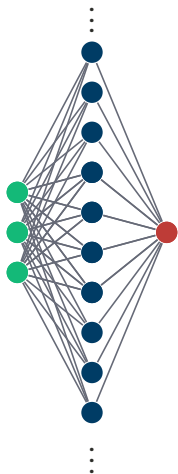


- Obtain a centered **Gaussian process**
- With covariance (**NNGP**) kernel

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

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

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

Due to the *Law of Large Numbers*.

Similarly

## Freezing of the NTK

*(Jacot, Gabriel, and Hongler 2018)*

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



What do we gain?

What do we gain?

is now deterministic and time-independent

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

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**

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  $\mu_t(x) = \mathbb{E}[\mathcal{N}_t(x)]$

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  $\mu_t(x) = \mathbb{E}[\mathcal{N}_t(x)]$

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  $\mu_t(x) = \mathbb{E}[\mathcal{N}_t(x)]$

## Analytic solution

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

Train inputs

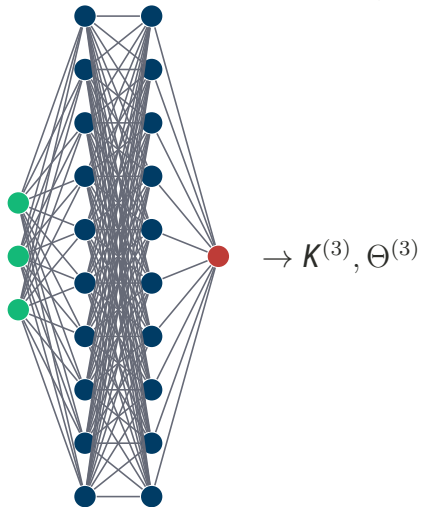
Train labels

How is the NTK computed for a given architecture?

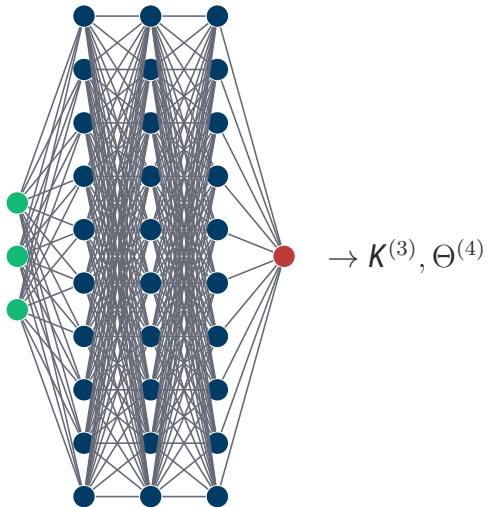
How is the NTK computed for a given architecture? → **Layer by layer**



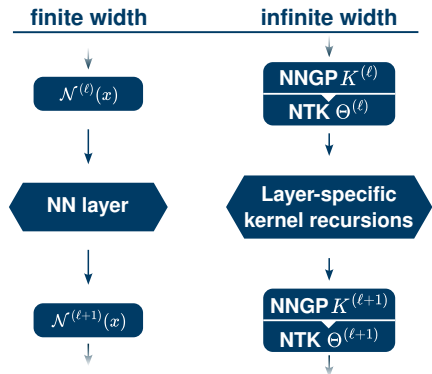
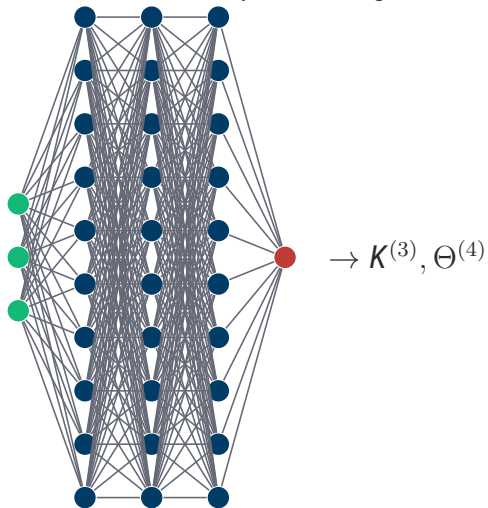
How is the NTK computed for a given architecture? → **Layer by layer**



How is the NTK computed for a given architecture? → **Layer by layer**

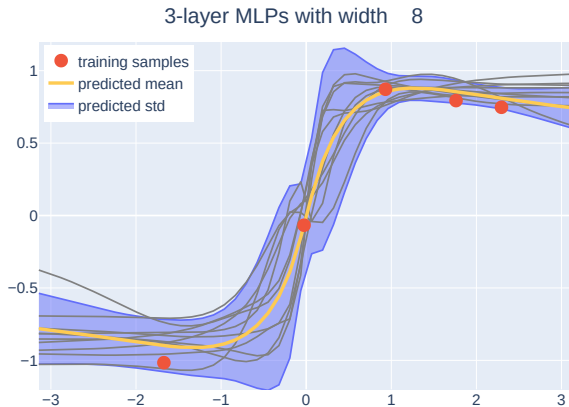


How is the NTK computed for a given architecture? → **Layer by layer**

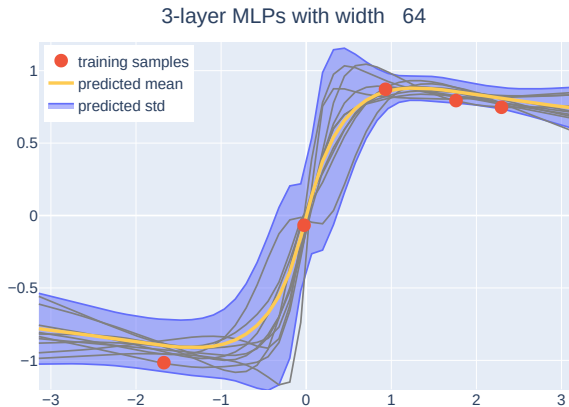


## Toy example: **Learning** $\sin(x)$

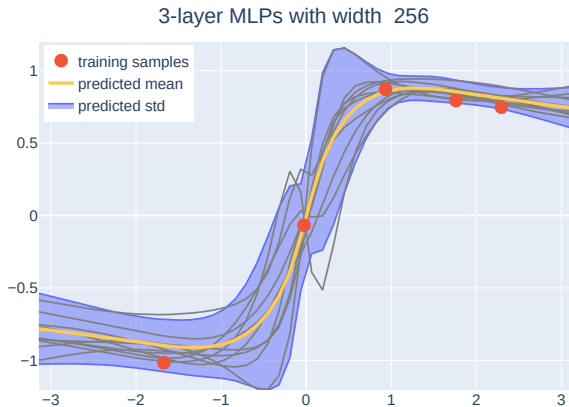
## Toy example: Learning $\sin(x)$



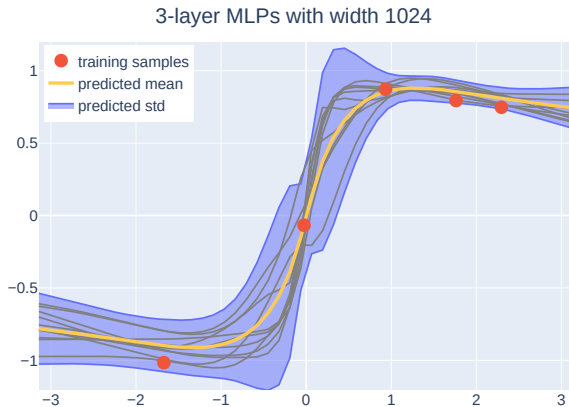
## Toy example: Learning $\sin(x)$



## Toy example: Learning $\sin(x)$

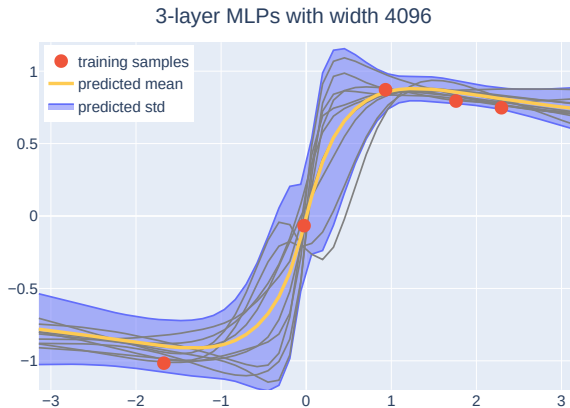


## Toy example: Learning $\sin(x)$





## Toy example: Learning $\sin(x)$



- 1 The Neural Tangent Kernel
- 2 Equivariance and Data Augmentation**
- 3 Beyond the strict limit with Feynman diagrams
- 4 Conclusion and Outlook

---

## Equivariant Neural Tangent Kernels

---

Philipp Misof<sup>1</sup> Pan Kessel<sup>2</sup> Jan E. Gerken<sup>1</sup>

### 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<sup>1</sup> Pan Kessel<sup>2</sup> Jan E. Gerken<sup>1</sup>

### 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

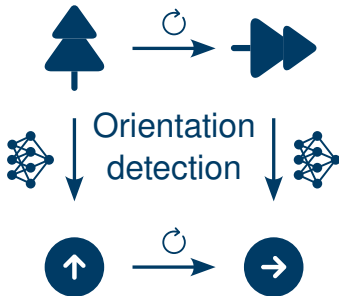


## Symmetries in Machine Learning

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

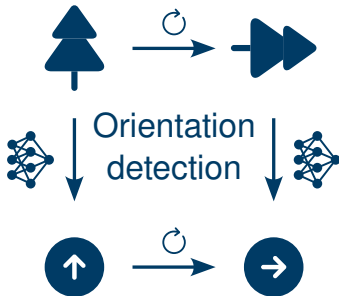
## Symmetries in Machine Learning

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



# Symmetries in Machine Learning

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

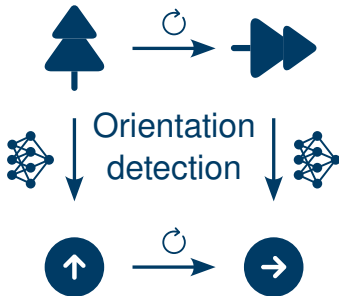


$$\begin{array}{ccc} f & \xrightarrow{\rho_{\text{in}}(g)} & \rho_{\text{in}}(g)(f) \\ \downarrow \mathcal{N} & & \downarrow \mathcal{N} \\ \mathcal{N}(f) & \xrightarrow{\rho_{\text{out}}(g)} & \rho_{\text{out}}(g)[\mathcal{N}(f)] \end{array}$$

$$\forall g \in G$$

# Symmetries in Machine Learning

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



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

$\forall g \in G$

This is called **equivariance**



# Convolutional Neural Networks (CNNs)

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

\*

II

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

# Convolutional Neural Networks (CNNs)

## Classic convolution layer

$$[\mathcal{N}^{(1)}(f)](y) = \frac{1}{\sqrt{|\mathcal{S}_\kappa|}} \int_{\mathbb{R}^d} dx \overset{\substack{\text{filter} \\ \downarrow}}{\kappa}(x - y) \overset{\substack{\text{domain of input signal} \\ \leftarrow}}{f}(x)$$

filter support
filter support

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

\*

II

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

# Convolutional Neural Networks (CNNs)

## Classic convolution layer

$$[\mathcal{N}^{(1)}(f)](y) = \frac{1}{\sqrt{|\mathcal{S}_\kappa|}} \int_{\mathbb{R}^d} dx \overset{\text{filter}}{\kappa}(x - y) \overset{\text{domain of input signal}}{f(x)}$$

filter support
filter
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

\*

II

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](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  $X$ .

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

## Group Convolutional Neural Networks (GCNNs)

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

$$[\mathcal{N}^{(1)}(f)](\textcolor{red}{g}) = \frac{1}{\sqrt{|S_\kappa|}} \int_{\textcolor{red}{X}} d\textcolor{red}{h} \kappa(\textcolor{red}{g}^{-1}\textcolor{red}{h}) [(f)](\textcolor{red}{h})$$


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  $X$ .

$$[\mathcal{N}^{(1)}(f)](\textcolor{red}{g}) = \frac{1}{\sqrt{|S_{\kappa}|}} \int_{\textcolor{red}{X}} d\textcolor{red}{h} \kappa(\textcolor{red}{g}^{-1}\textcolor{red}{h}) [(f)](\textcolor{red}{h})$$

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  $X$ .

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

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

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

**Group pooling**

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



# 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

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

$\uparrow$   
Evaluation point in group space

# The Equivariant NTK

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

$\uparrow$   
Evaluation point in group space

$\infty$ -width limit:

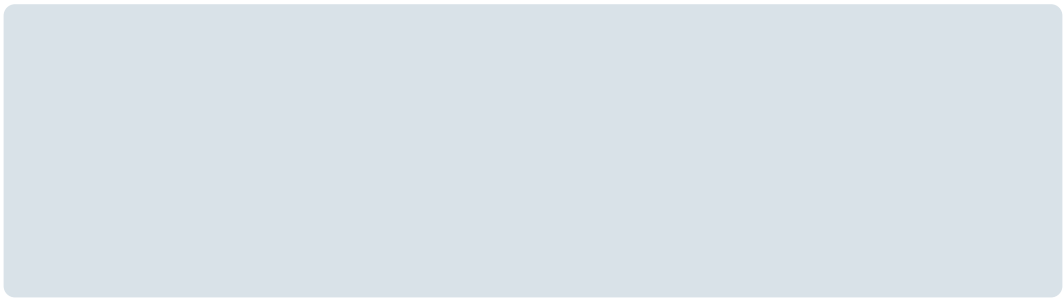
# The Equivariant NTK

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

$\uparrow$   
Evaluation point in group space

$\infty$ -width limit: # channels  $\rightarrow \infty$

## Kernel Recursions of the Group Convolutional Layer



## Kernel Recursions of the Group Convolutional Layer

$$K_{g,g'}^{(\ell+1)}(f, f') = \frac{1}{\text{vol}(S_\kappa)} \int_{S_\kappa} dh K_{gh,g'h}^{(\ell)}(f, f')$$

## Kernel Recursions of the Group Convolutional Layer

$$K_{g,g'}^{(\ell+1)}(f, f') = \frac{1}{\text{vol}(S_\kappa)} \int_{S_\kappa} dh \, K_{gh,g'h}^{(\ell)}(f, f')$$

$$\Theta_{g,g'}^{(\ell+1)}(f, f') = K_{g,g'}^{(\ell+1)}(f, f') + \frac{1}{\text{vol}(S_\kappa)} \int_{S_\kappa} dh \, \Theta_{gh,g'h}^{(\ell)}(f, f')$$



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

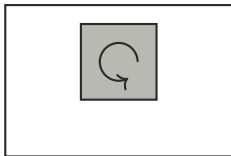
**How to implement this depends on the group  $G$  and the space  $X$ .**

We cover

**Roto-translations in the plane**

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

$$X = \mathbb{Z}^2$$



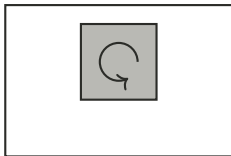
## How to implement this depends on the group $G$ and the space $X$ .

We cover

### Roto-translations in the plane

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

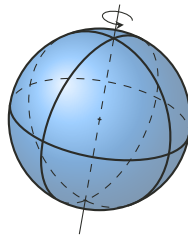
$$X = \mathbb{Z}^2$$



### Rotations on $SO(3)$

$$G = SO(3)$$

$$X = S^2$$



## SO(3) Implementation

$$K_{R,R'}^{(\ell+1)}(f, f') = \frac{1}{8\pi^2} \int_{\text{SO}(3)} dS K_{RS,R'S}^{(\ell)}(f, f')$$

## SO(3) Implementation

$$K_{R,R'}^{(\ell+1)}(f, f') = \frac{1}{8\pi^2} \int_{\text{SO}(3)} dS K_{RS,R'S}^{(\ell)}(f, f')$$

How can we compute this integral efficiently?

## SO(3) Implementation

$$K_{R,R'}^{(\ell+1)}(f, f') = \frac{1}{8\pi^2} \int_{\text{SO}(3)} dS K_{RS,R'S}^{(\ell)}(f, f')$$

How can we compute this integral efficiently?

→ using the **Fourier transform** on compact groups

(Wigner transform on SO(3))

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

## SO(3) Implementation

$$K_{R,R'}^{(\ell+1)}(f, f') = \frac{1}{8\pi^2} \int_{\text{SO}(3)} dS K_{RS,R'S}^{(\ell)}(f, f')$$

How can we compute this integral efficiently?

→ using the **Fourier transform** on compact groups

(Wigner transform on SO(3))

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

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

## SO(3) Implementation

$$K_{R,R'}^{(\ell+1)}(f, f') = \frac{1}{8\pi^2} \int_{\text{SO}(3)} dS K_{RS,R'S}^{(\ell)}(f, f')$$

How can we compute this integral efficiently?

→ using the **Fourier transform** on compact groups

(Wigner transform on SO(3))

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

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



## SO(3) Implementation

$$K_{R,R'}^{(\ell+1)}(f, f') = \frac{1}{8\pi^2} \int_{\text{SO}(3)} dS K_{RS,R'S}^{(\ell)}(f, f')$$

How can we compute this integral efficiently?

→ using the **Fourier transform** on compact groups

(Wigner transform on SO(3))

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

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

# Fourier Recursion

# Fourier Recursion

## Kernel recursion in Fourier space

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

## Fourier Recursion

Kernel recursion in Fourier space

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

**Approximation:** truncate for  $l \geq L$ .

## Fourier Recursion

### Kernel recursion in Fourier space

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

**Approximation:** truncate for  $l \geq L$ .

Straightforward to implement

## Fourier Recursion

### Kernel recursion in Fourier space

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

**Approximation:** truncate for  $l \geq L$ .

Straightforward to implement ... *right?*

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

The screenshot shows the GitHub repository page for 'neural-tangents' by user 'romanngg'. The repository is marked as 'Public archive' and has a notice: 'This repository was archived by the owner on May 6, 2025. It is now read-only.' The repository has 61 watches, 234 forks, and 2.4k stars. The main branch is 'main' with 5 branches and 17 tags. A recent commit by 'romanngg' is shown: 'Fix github action version (missed in previous commit)' from 'c17e770' last year, with 650 commits in total. The file list includes: .github/workflows, docs, examples, neural\_tangents, notebooks, presentation, tests, .readthedocs.yml, CITATION, CONTRIBUTING.md, LICENSE, LICENSE\_SHORT, and README.md. The right sidebar shows the 'About' section with the description 'Fast and Easy Infinite Neural Networks in Python', a link to 'iclr.cc/virtual\_2020/poster\_SkiD9yFP...', and various topic tags like 'kernel', 'neural-networks', 'gradient-descent', 'bayesian-inference', 'gaussian-processes', 'bayesian-networks', 'deep-networks', 'gradient-flow', 'jax', 'infinite-networks', 'training-dynamics', 'neural-tangents', and 'kernel-computation'. Below the topics are links to 'Readme', 'Apache-2.0 license', 'Code of conduct', 'Contributing', 'Security policy', 'Cite this repository', 'Activity', and 'Custom properties'.

google / neural-tangents

Type / to search

<> Code Issues 62 Pull requests 7 Discussions Actions Projects Wiki Security Insights

This repository was archived by the owner on May 6, 2025. It is now read-only.

neural-tangents Public archive

Watch 61 Fork 234 Star 2.4k

main 5 Branches 17 Tags

Go to file Code

romanngg Fix github action version (missed in previous commit) c17e770 · last year 650 Commits

.github/workflows	Fix github action version (missed in previous commit)	last year
docs	Update requirements.txt	2 years ago
examples	No-op refactoring: use the modern jnp/np aliases conventio...	2 years ago
neural_tangents	Avoid deprecated ad.config & ad.source_info_util	last year
notebooks	Updated jax.config import	2 years ago
presentation	Add paper	3 years ago
tests	Remove tests for zeros_like and add_any primitives from ne...	last year
.readthedocs.yml	Update readthedocs python version	2 years ago
CITATION	Deflake tests; update citation file	3 years ago
CONTRIBUTING.md	Project import generated by Copybara.	6 years ago
LICENSE	Refactoring: a long awaited refactor that splits the huge sta...	3 years ago
LICENSE_SHORT	Add paper	3 years ago
README.md	No-op refactoring: use the modern jnp/np aliases conventio...	2 years ago

About

Fast and Easy Infinite Neural Networks in Python

iclr.cc/virtual\_2020/poster\_SkiD9yFP...

kernel neural-networks gradient-descent bayesian-inference gaussian-processes bayesian-networks deep-networks gradient-flow jax infinite-networks training-dynamics neural-tangents kernel-computation

Readme Apache-2.0 license Code of conduct Contributing Security policy Cite this repository Activity Custom properties

2.4k stars 61 watching 234 forks

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

The screenshot shows the GitHub repository page for `s2fft` by `astro-informatics`. The repository is public and has 201 stars, 13 forks, and 10 watchers. The main branch is `main`, with 8 branches and 10 tags. The repository description is "S2FFT: Differentiable and accelerated spherical transforms". The repository includes a README, MIT license, and a Cite this repository link. The repository also has a list of releases, with the latest release being `v1.3.0`.

**Repository Details:**

- Repository: `s2fft` (Public)
- Stars: 201
- Forks: 13
- Watchers: 10
- Branches: 8
- Tags: 10

**Recent Commits:**

Commit	Message	Time
<code>dependabot[bot]</code>	Bump pypa/cibuildwheel from 3.1.3 to 3.1.4 (#323)	8 hours ago
<code>b239a11</code>	Bump pypa/cibuildwheel from 3.1.3 to 3.1.4 (#323)	8 hours ago
<code>main</code>	Update python_requires and test matrix to support Python ...	last month
<code>main</code>	Clean-up unused files (#271)	5 months ago
<code>main</code>	Add headers with attributions / license details	11 months ago
<code>main</code>	Update Torch notebook cell outputs (#301)	4 months ago
<code>main</code>	Update custom_ops.py (#315)	2 weeks ago
<code>main</code>	Factor out torch autograd checks into separate tests and skl...	3 months ago
<code>main</code>	docs: add mdavezac as a contributor for infra (#297)	4 months ago
<code>main</code>	clang format	last year
<code>main</code>	Exclude CUDA functions from test coverage	10 months ago
<code>main</code>	merge with main	11 months ago
<code>main</code>	Add pre-commit config	10 months ago
<code>main</code>	Update CITATION.cff (#270)	6 months ago

**Repository Features:**

- Readme
- MIT license
- Cite this repository
- Activity
- Custom properties
- 201 stars
- 10 watching
- 13 forks
- Report repository

**Releases:**

- `v1.3.0` (Latest)















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

The screenshot shows the GitHub repository page for `astro-informatics / s2fft`. The repository has 13 commits, 201 stars, and 3 forks. The Contributors section is highlighted, showing a grid of 10 contributors with their profile pictures, names, and GitHub handles. Below the contributors, there is a table of recent commits.

**Contributors** ✨

Thanks goes to these wonderful people ([emoji key](#)):

 <a href="#">Matt Price</a> @mattprice	 <a href="#">Jason McEwen</a> @jmcEwen	 <a href="#">Matt Graham</a> @mattgraham	 <a href="#">stefij</a> @stefij	 <a href="#">Devaraj Gopinathan</a> @devarajgopinathan	 <a href="#">Francois Lanusse</a> @francoislanusse	 <a href="#">Ikko Eltoclear</a> @ikko-eltoclear
 <a href="#">Kevin Mulder</a> @kevinmulder	 <a href="#">Philipp Misof</a> @philippmisof	 <a href="#">Ellis Roberts</a> @ellisroberts	 <a href="#">Wassim KABALAN</a> @wassimkabal	 <a href="#">Mayeul d'Avezac</a> @mayeuldavezac		

File	Commit Message	Time Ago
<code>.gitignore</code>	merge with main	11 months ago
<code>pre-commit-config.yaml</code>	Add pre-commit config	10 months ago
<code>CITATION.cff</code>	Update CITATION.cff (#270)	6 months ago

**Releases** 9

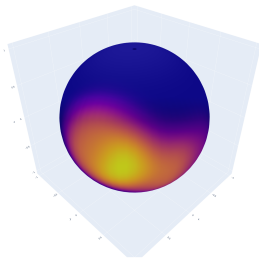
**v1.3.0** (Latest)

## Testing the $SO(3)$ NTK on molecular data (QM9)

## Testing the SO(3) NTK on molecular data (QM9)

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

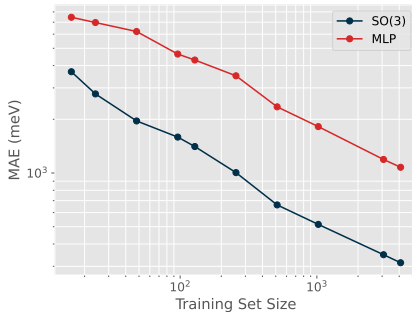
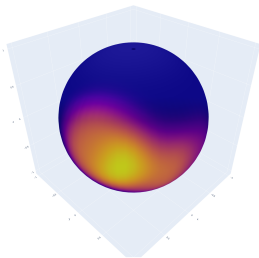
$$f_{i,z,p}(x) = \sum_{j:z_j=z} \frac{z_i z_j}{\|r_{ij}\|^p} e^{-\frac{1}{\beta} \left( \frac{r_{ij}}{\|r_{ij}\|} \cdot x - 1 \right)^2}$$



# Testing the SO(3) NTK on molecular data (QM9)

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

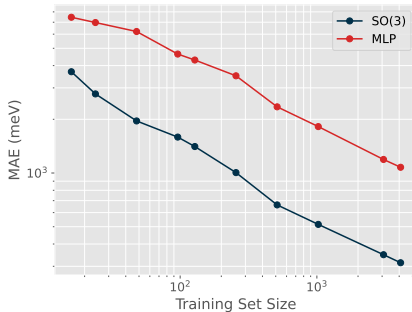
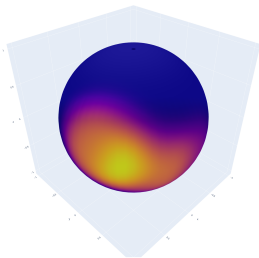
$$f_{i,z,p}(x) = \sum_{j:z_j=z} \frac{z_i z_j}{\|r_{ij}\|^p} e^{-\frac{1}{\beta} \left( \frac{r_{ij}}{\|r_{ij}\|} \cdot x - 1 \right)^2}$$



## Testing the $SO(3)$ NTK on molecular data (QM9)

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

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



Performance boost due to 3d-rotation invariance extends to the  $\infty$ -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  $\Leftrightarrow \tilde{\rho}_{\text{reg}} = \text{id}$

## 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  $\Leftrightarrow \tilde{\rho}_{\text{reg}} = \text{id}$

- $\mu_t^{\text{aug}}$  evolves like a non-augmented NN mean  $\mu_t$  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.

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

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

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

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

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

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

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

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

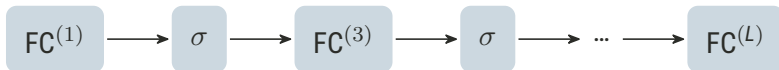


**Expectation** of a data augmented MLP equals the **expectation** of an GCNN at all training times  $t$ .

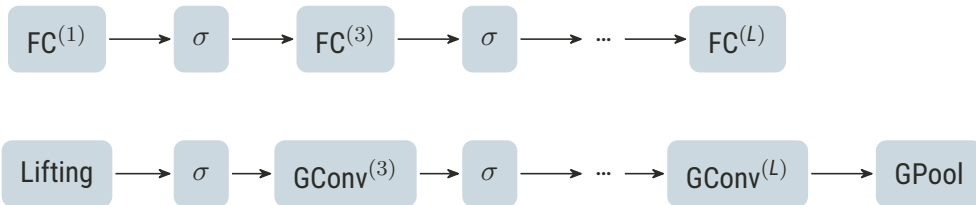
# Architecture correspondence



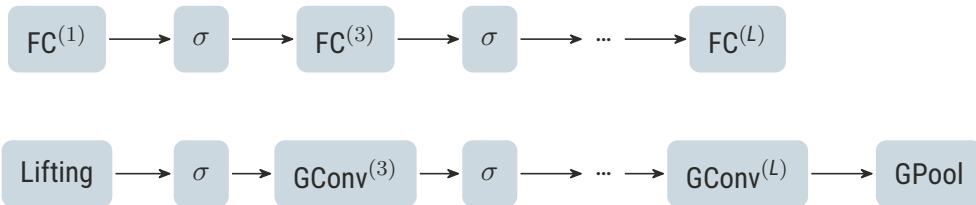
## Architecture correspondence



## 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**

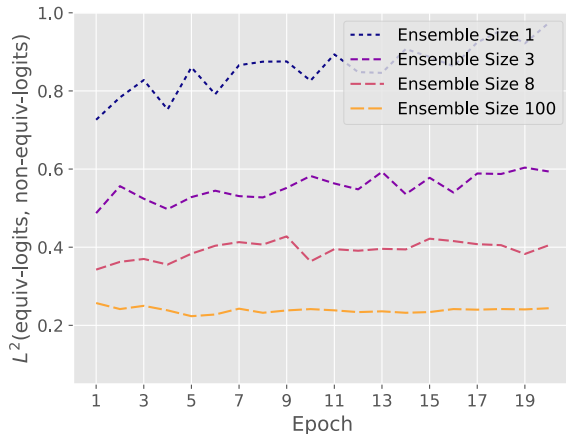
- Data augmented CNN vs  $C_4 \ltimes \mathbb{R}^2$   
GCNN on **MNIST**

## 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



- 1 The Neural Tangent Kernel
- 2 Equivariance and Data Augmentation
- 3 Beyond the strict limit with Feynman diagrams
- 4 Conclusion and Outlook



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

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

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

Taking  $n \rightarrow \infty$  allowed us 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  $n \rightarrow \infty$  allowed us 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  $n \rightarrow \infty$  allowed us 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  $n \rightarrow \infty$  allowed us 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



PAPER

## Neural networks and quantum field theory

James Halverson<sup>✉</sup>, Anindita Maiti<sup>\*</sup> and Keegan Stoner<sup>✉</sup>

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

<sup>\*</sup> Author to whom any correspondence should be addressed.

E-mail: [maiti.a@northeastern.edu](mailto:maiti.a@northeastern.edu)

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 license.  
Any further distribution  
of this work must  
maintain attribution to the author(s) and the copyright owner(s).

**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**

**PAPER**

**Neural networks and quantum field theory**

James Halverson<sup>✉</sup>, Anindita Maiti<sup>\*</sup> and Keegan Stoner<sup>✉</sup>  
Department of Physics, Northeastern University, Boston, MA 02115, United States of America  
<sup>\*</sup> Author to whom any correspondence should be addressed.  
E-mail: [maiti.a@northeastern.edu](mailto:maiti.a@northeastern.edu)

**OPEN ACCESS**

**RECEIVED**  
22 August 2020

**REVISED**  
8 February 2021

**ACCEPTED FOR PUBLICATION**  
8 March 2021

**PUBLISHED**  
27 April 2021

**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

Original Content from this work may be used under the terms of the Creative Commons Attribution 4.0 license. Any further distribution of this work must credit the author(s) and the source.

## The Principles of Deep Learning Theory

An Effective Theory Approach  
to Understanding Neural Networks

DANIEL A. ROBERTS  
*MIT*

SHO YAJIDA  
*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

**PAPER**

**Neural networks and quantum field theory**

James Halverson<sup>✉</sup>, Anindita Maiti<sup>✉</sup> and Keegan Stoner<sup>✉</sup>  
Department of Physics, Northeastern University, Boston, MA 02115, United States of America  
<sup>\*</sup> Author to whom any correspondence should be addressed.  
E-mail: [maiti.a@northeastern.edu](mailto:maiti.a@northeastern.edu)

**OPEN ACCESS**

**RECEIVED**  
22 August 2020

**REVISED**  
8 February 2021

**ACCEPTED FOR PUBLICATION**  
8 March 2021

**PUBLISHED**  
27 April 2021

**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

Original Content from this work may be used under the terms of the [Creative Commons Attribution 4.0 license](#). Any further distribution of this work must maintain attribution to the author(s) and the copyright owner(s).

## The Principles of Deep Learning Theory

An Effective Theory Approach  
to Understanding Neural Networks

DANIEL A. ROBERTS  
*MIT*

SHO YAJIDA  
*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**

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

Decompose it **layer by layer**

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

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

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

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

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

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

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

Instead of a Normal distribution with probability density function

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



$$p(z^{(\ell+1)}|\mathcal{D}) = \int \prod_{i,\alpha} dz_{i;\alpha}^{(\ell)} p(z^{(\ell+1)}|z^{(\ell)}) \color{red}{p(z^{(\ell)}|\mathcal{D})}$$

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

Instead of a Normal distribution with probability density function

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

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

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

$$p(z^{(\ell+1)}|\mathcal{D}) = \int \prod_{i,\alpha} dz_{i;\alpha}^{(\ell)} p(z^{(\ell+1)}|z^{(\ell)}) \textcolor{red}{p}(z^{(\ell)}|\mathcal{D})$$

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

Instead of a Normal distribution with probability density function

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

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

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

# 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  $V^{(\alpha_1 \alpha_2)(\alpha_3 \alpha_4)} = \mathbb{E}^c[z_{\alpha_1}, z_{\alpha_2}, z_{\alpha_3}, z_{\alpha_4}]$  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  $K, \Theta$  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  $K, \Theta$  and  $V_4$  as well as joint cumulants  $A, B, D, F$  of degree 4.

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

# Recursions

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

# Recursions

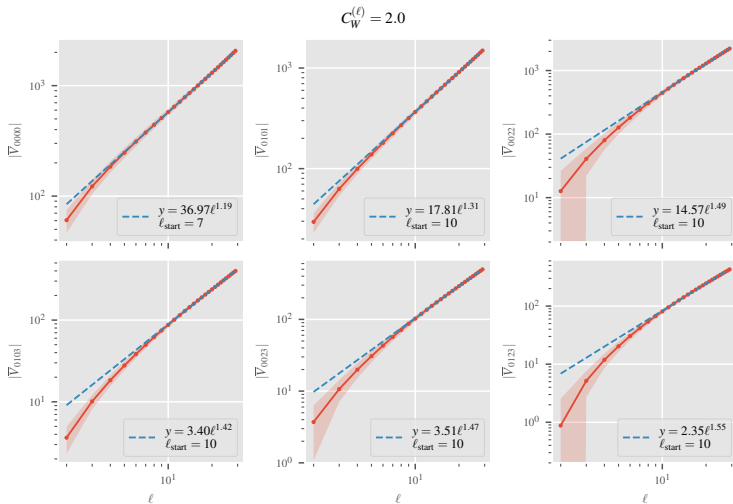
- have been used to find optimal initialization hyperparameters (**Criticality**)
- Explain qualitative **differences between activation functions**

# Recursions

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

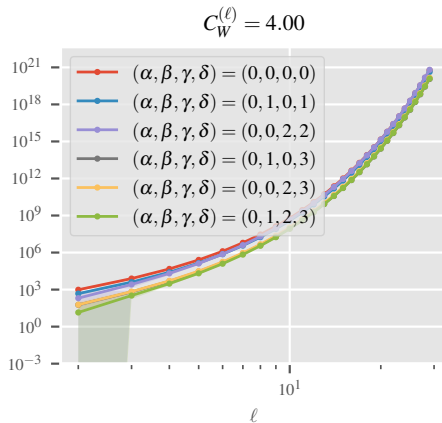
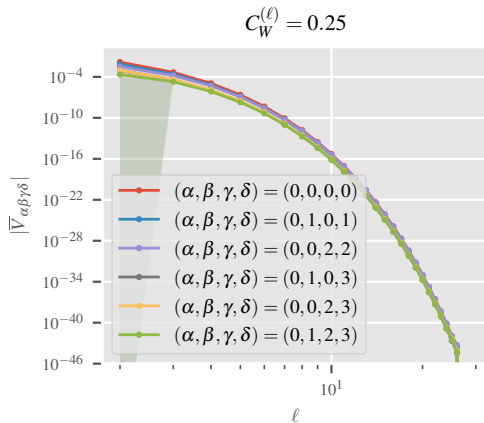
# Empirical $V_4$ evolution at criticality

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<sup>\*a</sup>

Philipp Misof<sup>\*a</sup>

Jan E. Gerken<sup>a</sup>

## 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**

$$\begin{aligned}
 & \text{Diagram with grey circle and external lines } 1, 2, 3, 4 \\
 &= \sum_j \text{Diagram with white circle and external lines } 1, 2, 3, 4 \text{ and labels } \sigma_j, \sigma'_j \\
 &+ \sum_{j_1, j_2} \text{Diagram with two white circles and external lines } 1, 2, 3, 4 \text{ and labels } \sigma_{j_1}, \sigma'_{j_1}, \sigma_{j_2}, \sigma'_{j_2} \\
 &\quad \text{and internal labels } z_{j_1}, z_{j_2} \text{ and } \frac{1}{n_{\ell-1}} F_4^{(\ell)}
 \end{aligned}$$



# Recursions from Feynman diagrams

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

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

**Example:  $F$  recursion**

$$\begin{aligned}
 & \text{Diagram with 4 external legs (1, 2, 3, 4) and a central grey vertex} \\
 &= \sum_j \text{Diagram with 4 external legs (1, 2, 3, 4) and two internal white vertices connected by a dashed line, labeled } \sigma_j \sigma'_j \\
 &+ \sum_{j_1, j_2} \text{Diagram with 4 external legs (1, 2, 3, 4) and three internal vertices (two white, one grey) connected by dashed lines, labeled } \sigma_{j_1} \sigma'_{j_1}, z_{j_1}, z_{j_2}, \sigma_{j_2} \sigma'_{j_2} \\
 &\quad \text{The central dashed line is labeled } \frac{1}{n_{\ell-1}} F_4^{(\ell)}
 \end{aligned}$$

**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  $\Theta^{\{0\}}(\ell)$

The diagram illustrates the recursion for the first order correction  $\Theta^{\{1\}}(\ell)$  to the infinite width NTK  $\Theta^{\{0\}}(\ell)$ . The equation is:

$$\frac{1}{n_\ell} \Theta_{12}^{\{1\}}(\ell+1) = \frac{1}{n_{\ell-1}} \Theta^{(1)}(\ell) + \frac{1}{n_{\ell-1}} K^{\{1\}}(\ell) + \frac{1}{n_{\ell-1}} V_4^{(\ell)} + \frac{1}{n_{\ell-1}} D_4^{(\ell)} + \frac{1}{n_{\ell-1}} F_4^{(\ell)}$$

The diagrams are as follows:

- Left side:** A diagram with two external points labeled 1 and 2, connected by a dashed line. A grey circle is attached to this line. The label below is  $\frac{1}{n_\ell} \Theta_{12}^{\{1\}}(\ell+1)$ .
- First term:** A diagram with two external points labeled 1 and 2, connected by a dashed line. A grey circle is attached to this line, and a dashed blue circle is also attached. The label above is  $\frac{1}{n_{\ell-1}} \Theta^{(1)}(\ell)$ . The label below is  $\frac{1}{n_{\ell-1}} \Theta_{12}^{\{1\}}(\ell)$ .
- Second term:** A diagram with two external points labeled 1 and 2, connected by a dashed line. A grey circle is attached to this line, and a wavy line labeled  $z_j$  is attached. The label above is  $\frac{1}{n_{\ell-1}} K^{\{1\}}(\ell)$ . The label below is  $\frac{1}{n_{\ell-1}} K_{12}^{\{1\}}(\ell)$ .
- Third term:** A diagram with two external points labeled 1 and 2, connected by a dashed line. A grey circle is attached to this line, and a wavy line labeled  $z_j$  is attached. The label above is  $\frac{1}{n_{\ell-1}} V_4^{(\ell)}$ . The label below is  $\frac{1}{n_{\ell-1}} V_{12}^{(\ell)}$ .
- Fourth term:** A diagram with two external points labeled 1 and 2, connected by a dashed line. A grey circle is attached to this line, and a dashed blue circle is also attached. The label above is  $\frac{1}{n_{\ell-1}} D_4^{(\ell)}$ . The label below is  $\frac{1}{n_{\ell-1}} D_{12}^{(\ell)}$ .
- Fifth term:** A diagram with two external points labeled 1 and 2, connected by a dashed line. A grey circle is attached to this line, and a wavy line labeled  $z_j$  is attached. The label above is  $\frac{1}{n_{\ell-1}} F_4^{(\ell)}$ . The label below is  $\frac{1}{n_{\ell-1}} F_{12}^{(\ell)}$ .

## Solving the $V_4$ recursion [WIP]

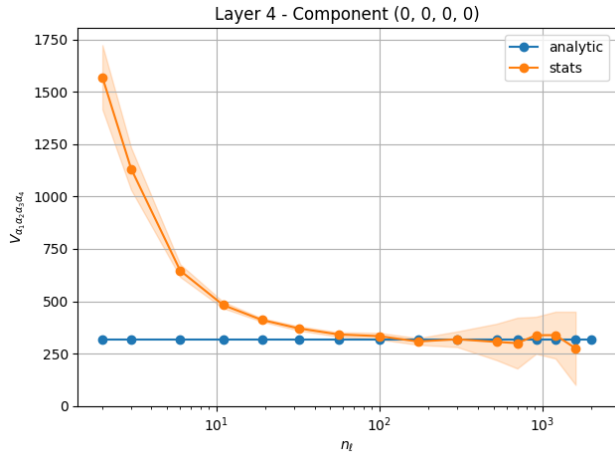
## Solving the $V_4$ recursion [WIP]

$$\begin{aligned}
 \frac{1}{n_\ell} V_{(\alpha_1 \alpha_2)(\alpha_3 \alpha_4)}^{(\ell+1)} &= \frac{1}{n_\ell} \left( \mathcal{C}_W^{(\ell+1)} \right)^2 \left[ \langle \sigma_{\alpha_1} \sigma_{\alpha_2} \sigma_{\alpha_3} \sigma_{\alpha_4} \rangle_{\mathbf{G}^{(\ell)}} - \langle \sigma_{\alpha_1} \sigma_{\alpha_2} \rangle_{\mathbf{G}^{(\ell)}} \langle \sigma_{\alpha_3} \sigma_{\alpha_4} \rangle_{\mathbf{G}^{(\ell)}} \right] \\
 &+ \frac{1}{n_{\ell-1}} \frac{\left( \mathcal{C}_W^{(\ell+1)} \right)^2}{4} \sum_{\beta_1, \dots, \beta_4 \in D} V_{(\ell)}^{(\beta_1 \beta_2)(\beta_3 \beta_4)} \langle \sigma_{\alpha_1} \sigma_{\alpha_2} (\mathbf{z}_{\beta_1} \mathbf{z}_{\beta_2} - \mathbf{g}_{\beta_1 \beta_2}) \rangle_{\mathbf{G}^{(\ell)}} \\
 &\times \langle \sigma_{\alpha_3} \sigma_{\alpha_4} (\mathbf{z}_{\beta_3} \mathbf{z}_{\beta_4} - \mathbf{g}_{\beta_3 \beta_4}) \rangle_{\mathbf{G}^{(\ell)}} + \mathcal{O} \left( \frac{1}{n^2} \right)
 \end{aligned}$$

## Solving the $V_4$ recursion [WIP]

$$\begin{aligned}
 \frac{1}{n_\ell} V_{(\alpha_1 \alpha_2)(\alpha_3 \alpha_4)}^{(\ell+1)} &= \frac{1}{n_\ell} \left( \mathcal{C}_W^{(\ell+1)} \right)^2 \left[ \langle \sigma_{\alpha_1} \sigma_{\alpha_2} \sigma_{\alpha_3} \sigma_{\alpha_4} \rangle_{\mathbf{G}^{(\ell)}} - \langle \sigma_{\alpha_1} \sigma_{\alpha_2} \rangle_{\mathbf{G}^{(\ell)}} \langle \sigma_{\alpha_3} \sigma_{\alpha_4} \rangle_{\mathbf{G}^{(\ell)}} \right] \\
 &+ \frac{1}{n_{\ell-1}} \frac{\left( \mathcal{C}_W^{(\ell+1)} \right)^2}{4} \sum_{\beta_1, \dots, \beta_4 \in D} V_{(\ell)}^{(\beta_1 \beta_2)(\beta_3 \beta_4)} \langle \sigma_{\alpha_1} \sigma_{\alpha_2} (z_{\beta_1} z_{\beta_2} - g_{\beta_1 \beta_2}) \rangle_{\mathbf{G}^{(\ell)}} \\
 &\times \langle \sigma_{\alpha_3} \sigma_{\alpha_4} (z_{\beta_3} z_{\beta_4} - g_{\beta_3 \beta_4}) \rangle_{\mathbf{G}^{(\ell)}} + \mathcal{O} \left( \frac{1}{n^2} \right)
 \end{aligned}$$

## 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  $2d$  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!