Equivariant Neural Tangent Kernels

Connecting data augmentation and equivariant architectures

by Philipp Misof

Department of Mathematical Sciences, Division of Algebra and Geometry

August 19, 2025

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

WASP WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM

Joint work with

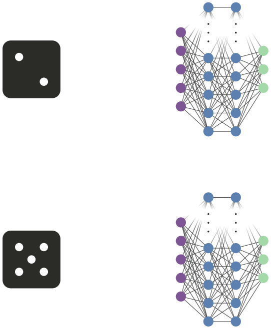
Jan Gerken
(Chalmers)

Pan Kessel
(Prescient Design, Switzerland)

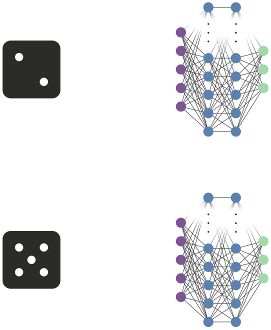
1. The Neural Tangent Kernel
2. NTK of Equivariant NNs
3. Concrete Examples
4. Data Augmentation vs GCNNs

What can we say about the **dynamics of NNs** without specifying their initial parameters?

What can we say about the **dynamics of NNs** without specifying their initial parameters?

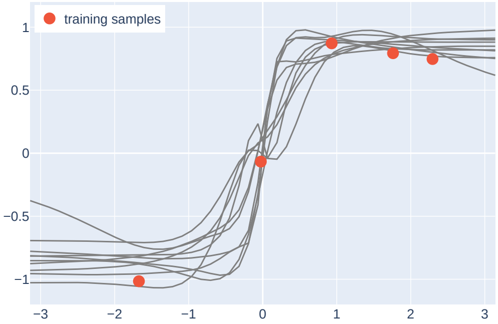


What can we say about the **dynamics of NNs** without specifying their initial parameters?

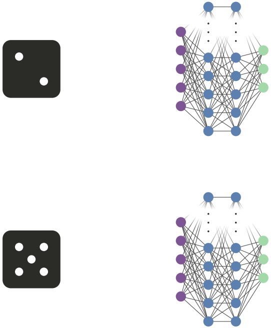


Let's learn $\sin(x)$

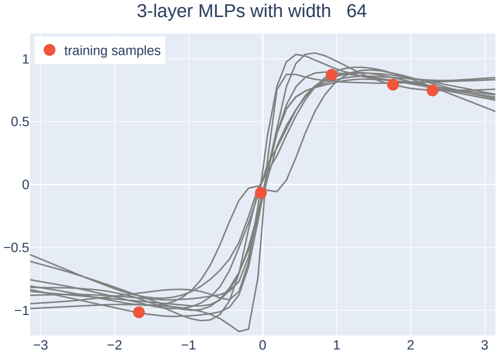
3-layer MLPs with width 8



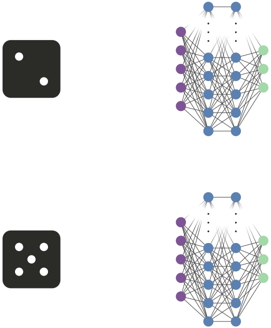
What can we say about the **dynamics of NNs** without specifying their initial parameters?



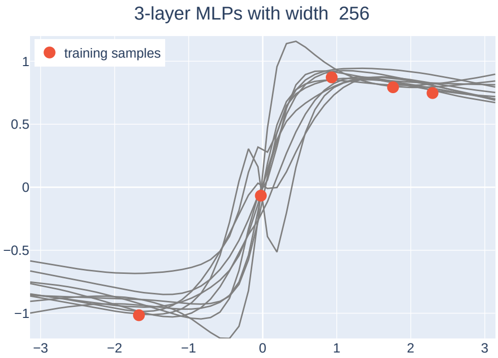
Let's learn $\sin(x)$



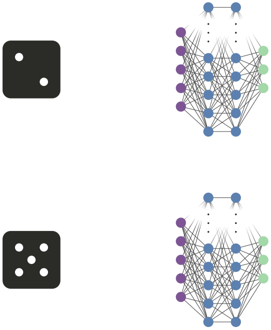
What can we say about the **dynamics of NNs** without specifying their initial parameters?



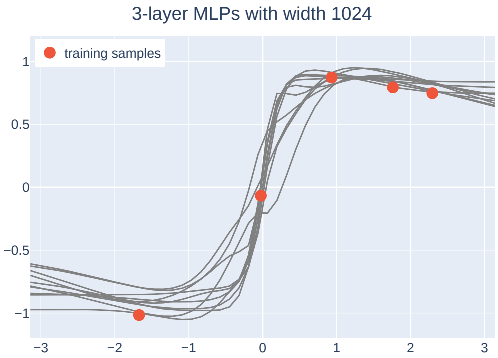
Let’s learn $\sin(x)$



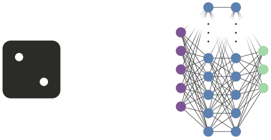
What can we say about the **dynamics of NNs** without specifying their initial parameters?

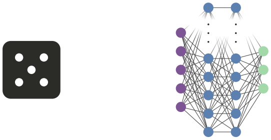


Let’s learn $\sin(x)$

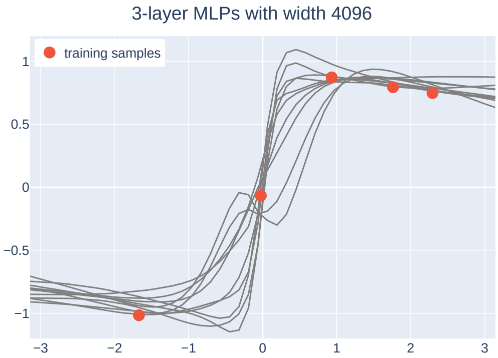


What can we say about the **dynamics of NNs** without specifying their initial parameters?

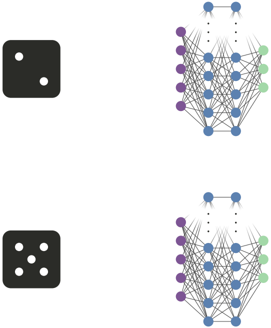




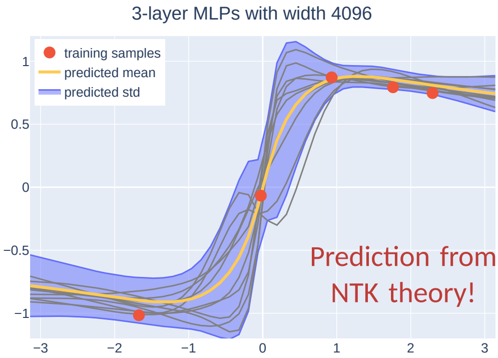
Let's learn $\sin(x)$



What can we say about the **dynamics of NNs** without specifying their initial parameters?



Let’s learn $\sin(x)$



Training dynamics

(Jacot, Gabriel, and Hongler 2018)

Training dynamics

(Jacot, Gabriel, and Hongler 2018)

Assume **Gradient Flow** instead of discrete Gradient Descent

Training dynamics

(Jacot, Gabriel, and Hongler 2018)

Assume **Gradient Flow** instead of discrete Gradient Descent

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

Training dynamics

(Jacot, Gabriel, and Hongler 2018)

Assume **Gradient Flow** instead of discrete Gradient Descent

Neural Network output

Learning rate

Loss

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

Training time

(empirical) Neural Tangent Kernel

Training dynamics

(Jacot, Gabriel, and Hongler 2018)

Assume **Gradient Flow** instead of discrete Gradient Descent

Neural Network output

Learning rate

Loss

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

Training time

(empirical) Neural Tangent Kernel

Training dynamics

(Jacot, Gabriel, and Hongler 2018)

Assume **Gradient Flow** instead of discrete Gradient Descent

Neural Network output

Learning rate

Loss

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

Training time

(empirical) Neural Tangent Kernel

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

So far, nothing is gained

So far, nothing is gained

Θ_t is dependent on θ_t, which is **stochastic** and **time-dependent**

So far, nothing is gained

Θ_t is dependent on θ_t, which is **stochastic** and **time-dependent**

Freezing of the NTK

(Jacot, Gabriel, and Hongler 2018)

So far, nothing is gained

Θ_t is dependent on θ_t, which is **stochastic** and **time-dependent**

Freezing of the NTK

(Jacot, Gabriel, and Hongler 2018)

$$\Theta_t \xrightarrow[\text{layer width}]{\text{increasing}} \Theta = \mathbb{E}[\Theta_t]$$

So far, nothing is gained

Θ_t is dependent on θ_t, which is **stochastic** and **time-dependent**

Freezing of the NTK

(Jacot, Gabriel, and Hongler 2018)

$$\Theta_t \xrightarrow[\text{layer width}]{\text{increasing}} \Theta = \mathbb{E}[\Theta_t] \quad \Rightarrow \quad \text{Simple ODE}$$

So far, nothing is gained

Θ_t is dependent on θ_t, which is **stochastic** and **time-dependent**

Freezing of the NTK

(Jacot, Gabriel, and Hongler 2018)

$$\Theta_t \xrightarrow[\text{layer width}]{\text{increasing}} \Theta = \mathbb{E}[\Theta_t] \quad \Rightarrow \quad \text{Simple ODE}$$

In case of MLE loss, **closed-form solution** of the **mean**

$$\mu(x) = \Theta(x, \mathcal{X})\Theta(\mathcal{X}, \mathcal{X})^{-1}\mathcal{Y} \qquad \text{as } t \to \infty$$

at ∞ width.

So far, nothing is gained

Θ_t is dependent on θ_t, which is **stochastic** and **time-dependent**

Freezing of the NTK

(Jacot, Gabriel, and Hongler 2018)

$$\Theta_t \xrightarrow[\text{layer width}]{\text{increasing}} \Theta = \mathbb{E}[\Theta_t] \quad \Rightarrow \quad \text{Simple ODE}$$

In case of MLE loss, **closed-form solution** of the **mean**

$$\mu(x) = \Theta(x, \underbrace{\mathcal{X}}_{\text{Training inputs}})\Theta(\mathcal{X}, \mathcal{X})^{-1}\underbrace{\mathcal{Y}}_{\text{Training targets}} \qquad \text{as } t \to \infty$$

at ∞ width.

What do we gain?

What do we gain?

- Useful correspondence

$$\textbf{Neural Network} \longleftrightarrow \textbf{Kernel Method}$$

What do we gain?

- Useful correspondence

analytically well-understood!

$$\downarrow$$

Neural Network $\longleftrightarrow$ **Kernel Method**

What do we gain?

- Useful correspondence

analytically well-understood!

$$\downarrow$$

Neural Network $\longleftrightarrow$ **Kernel Method**

- Non-linear **kernel method inspired by** empirical insights from **NNs** *(Arora et al. 2020)*

What do we gain?

- Useful correspondence

$$\begin{array}{ccc} & & \text{analytically well-understood!} \\ & & \downarrow \\ \textbf{Neural Network} & \longleftrightarrow & \textbf{Kernel Method} \end{array}$$

- Non-linear **kernel method inspired by** empirical insights from **NNs** *(Arora et al. 2020)*
- Study **trainability vs generalization** regimes *(Xiao, Pennington, and Schoenholz 2020)*

What do we gain?

- Useful correspondence

$$\begin{array}{ccc} & & \text{analytically well-understood!} \\ & & \downarrow \\ \textbf{Neural Network} & \longleftrightarrow & \textbf{Kernel Method} \end{array}$$

- Non-linear **kernel method inspired by** empirical insights from **NNs** *(Arora et al. 2020)*
- Study **trainability vs generalization** regimes *(Xiao, Pennington, and Schoenholz 2020)*
- **Spectral bias** inside and outside the training set *(Bowman and Montufar 2022)*

What do we gain?

- Useful correspondence

$$\textbf{Neural Network} \longleftrightarrow \underset{\uparrow}{\textbf{Kernel Method}}$$

analytically well-understood!

- Non-linear **kernel method inspired by** empirical insights from **NNs** *(Arora et al. 2020)*
- Study **trainability vs generalization** regimes *(Xiao, Pennington, and Schoenholz 2020)*
- **Spectral bias** inside and outside the training set *(Bowman and Montufar 2022)*
- Tool for **dataset distillation** *(Nguyen, Chen, and Lee 2021)*

How is the NTK computed in practice?

How is the NTK computed in practice?

$\rightarrow$ layer by layer! *(Jacot, Gabriel, and Hongler 2018)*

How is the NTK computed in practice?

$\longrightarrow$ layer by layer! *(Jacot, Gabriel, and Hongler 2018)*

Convenient: *Neural Network Gaussian Process* Kernel (**NNGP**)

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

How is the NTK computed in practice?

$\longrightarrow$ layer by layer! *(Jacot, Gabriel, and Hongler 2018)*

Convenient: *Neural Network Gaussian Process* Kernel (**NNGP**)

$$K^{(\ell)}(x, x') = \mathbb{E}\left[\overbrace{\mathcal{N}^{(\ell)}}^{\ell^{\text{th}}\text{ layer neurons}}(x) \left(\mathcal{N}^{(\ell)}(x')\right)^{\mathsf{T}}\right]$$

How is the NTK computed in practice?

$\longrightarrow$ layer by layer! *(Jacot, Gabriel, and Hongler 2018)*

Convenient: *Neural Network Gaussian Process* Kernel (**NNGP**)

$$K^{(\ell)}(x, x') = \mathbb{E}\left[\overbrace{\mathcal{N}^{(\ell)}}^{\ell^{\text{th}}\text{ layer neurons}}(x)\left(\mathcal{N}^{(\ell)}(x')\right)^{\mathsf{T}}\right]$$

Each NN layer then corresponds to a particular **recursion**

$$\begin{aligned} K^{(\ell+1)}(x, x') &= A^{(\ell)}(K^{(\ell)}(x, x')), \\ \Theta^{(\ell+1)}(x, x') &= B^{(\ell)}(\Theta^{(\ell)}(x, x'), K^{(\ell+1)}(x, x')) \end{aligned}$$

How is the NTK computed in practice?

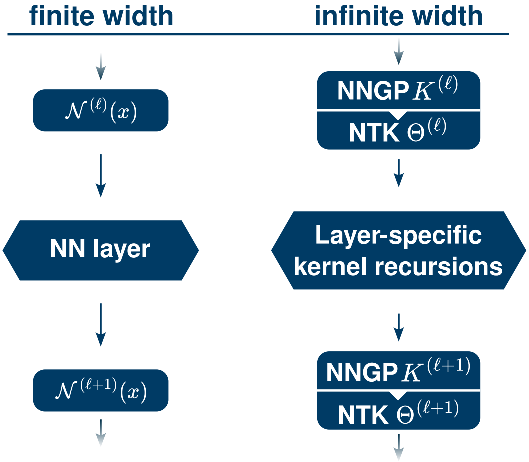
$\longrightarrow$ layer by layer! *(Jacot, Gabriel, and Hongler 2018)*

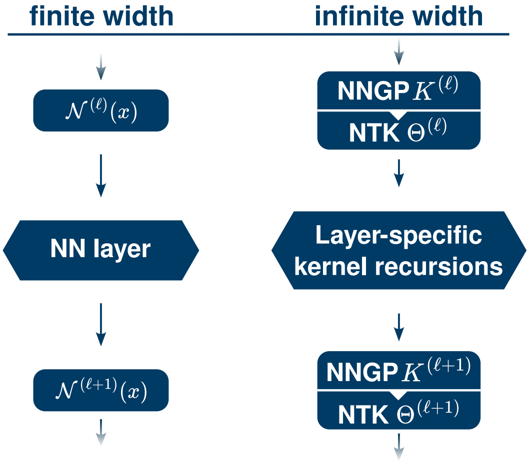
Convenient: *Neural Network Gaussian Process* Kernel (**NNGP**)

$$K^{(\ell)}(x, x') = \mathbb{E}\left[\overbrace{\mathcal{N}^{(\ell)}}^{\ell^{\text{th}}\text{ layer neurons}}(x)\left(\mathcal{N}^{(\ell)}(x')\right)^{\mathsf{T}}\right]$$

Each NN layer then corresponds to a particular **recursion**

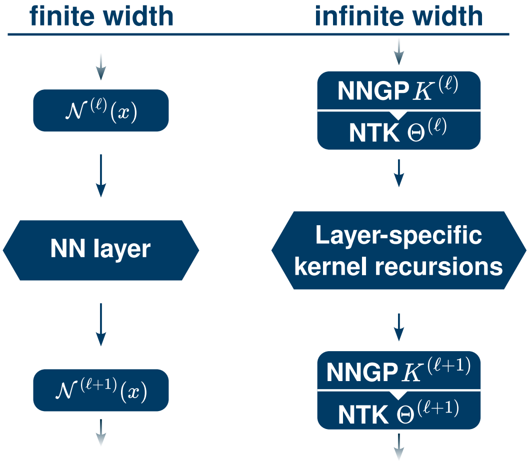
$$\begin{aligned}
K^{(\ell+1)}(x, x') &= \overbrace{A^{(\ell)}}^{\text{layer-specific NNGP map}}(K^{(\ell)}(x, x')), \\
\Theta^{(\ell+1)}(x, x') &= \underbrace{B^{(\ell)}}_{\text{layer-specific NTK map}}(\Theta^{(\ell)}(x, x'), K^{(\ell+1)}(x, x'))
\end{aligned}$$


finite width
infinite width
$\mathcal{N}^{(\ell)}(x)$
NN layer
$\mathcal{N}^{(\ell+1)}(x)$
NNGP $K^{(\ell)}$
NTK $\Theta^{(\ell)}$
Layer-specific kernel recursions
NNGP $K^{(\ell+1)}$
NTK $\Theta^{(\ell+1)}$



$\rightarrow$ Implemented in the `neural-tangents` library

(Novak et al. 2020)



$\rightarrow$ Implemented in the `neural-tangents` library (Novak et al. 2020)

Convention: We treat nonlinearities σ as individual layers

We extend the NTK to equivariant NNs

We extend the NTK to equivariant NNs

We cover

Group Convolution

We extend the NTK to equivariant NNs

We cover

Group Convolution

Group Pooling

We extend the NTK to equivariant NNs

We cover

Group Convolution

Group Pooling

Elementwise Non-linearity

1 The Neural Tangent Kernel

2 NTK of Equivariant NNs

3 Concrete Examples

4 Data Augmentation vs GCNNs

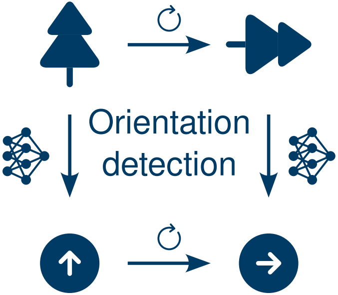
Recap on equivariant NNs

Recap on equivariant NNs

Want to enforce **symmetry** w.r.t a group G acting on the input signal f

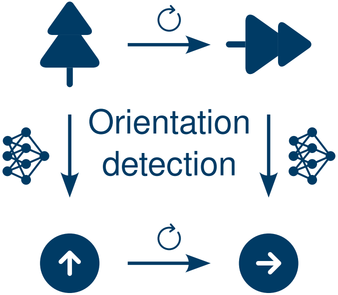
Recap on equivariant NNs

Want to enforce **symmetry** w.r.t a group G acting on the input signal f



Recap on **equivariant NNs**

Want to enforce **symmetry** w.r.t a group G acting on the input signal f



$$\begin{tikzcd} f \arrow[r, "\rho_{\text{in}}(g)"] \arrow[d, "\mathcal{N}"] & \rho_{\text{in}}(g)(f) \arrow[d, "\mathcal{N}"] \\ \mathcal{N}(f) \arrow[r, "\rho_{\text{out}}(g)"] & \rho_{\text{out}}(g)[\mathcal{N}(f)] \end{tikzcd}$$

$$\forall g \in G$$

Recap on Group Convolutional Neural Networks (GCNNs)

Recap on Group Convolutional Neural Networks (GCNNs)

Group convolutional layer

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

Recap on Group Convolutional Neural Networks (GCNNs)

Group convolutional layer

filter

input signal

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

group element

group

\# channels

filter support

Recap on Group Convolutional Neural Networks (GCNNs)

Group pooling

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

Recap on Group Convolutional Neural Networks (GCNNs)

Group pooling

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

Elementwise non-linearity σ

$$[\mathcal{N}^{(\ell+1)}(f)](g) = \sigma\big([\mathcal{N}^{(\ell)}(f)](g)\big), \qquad \sigma : \mathbb{R} \to \mathbb{R} \text{ elementwise}$$

The Equivariant NTK

The Equivariant NTK

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

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

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

Evaluation point in group space

∞-width limit:

The Equivariant NTK

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

Evaluation point in group space

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

Recursion of the group convolutional layer

Recursion of the group convolutional layer

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

Recursion of the group convolutional layer

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

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

Recursion of the group pooling layer

Recursion of the group pooling layer

$$K^{(\ell+1)}(f,f') = \frac{1}{\mathrm{vol}(G)} \int_G \mathrm{d}g\,\mathrm{d}g'\; K^{(\ell)}_{g,g'}(f,f')$$

Recursion of the group pooling layer

$$K^{(\ell+1)}(f,f') = \frac{1}{\mathrm{vol}(G)} \int_G \mathrm{d}g\,\mathrm{d}g'\; K^{(\ell)}_{g,g'}(f,f')$$

$$\Theta^{(\ell+1)}(f,f') = \frac{1}{\mathrm{vol}(G)} \int_G \mathrm{d}g\,\mathrm{d}g'\; \Theta^{(\ell)}_{g,g'}(f,f')$$

Why is that interesting?

Why is that interesting?

- Can compare equivariant architectures with their non-equivariant counterparts **in a parameter-independent setting**

Why is that interesting?

- Can compare equivariant architectures with their non-equivariant counterparts **in a parameter-independent setting**
- Can compare **data augmentation** $\leftrightarrow$ **equivariant architectures**

Why is that interesting?

- Can compare equivariant architectures with their non-equivariant counterparts **in a parameter-independent setting**
- Can compare **data augmentation** $\leftrightarrow$ **equivariant architectures**
  - Ensembles of data-augmented networks are equivariant *(Gerken and Kessel 2024; Nordenfors and Flinth 2024)*

Why is that interesting?

- Can compare equivariant architectures with their non-equivariant counterparts **in a parameter-independent setting**
- Can compare **data augmentation** $\leftrightarrow$ **equivariant architectures**
  - Ensembles of data-augmented networks are equivariant *(Gerken and Kessel 2024; Nordenfors and Flinth 2024)*
  - $\rightarrow$ How does this form of equivariance differ?

Why is that interesting?

- Can compare equivariant architectures with their non-equivariant counterparts **in a parameter-independent setting**
- Can compare **data augmentation** $\leftrightarrow$ **equivariant architectures**
  - Ensembles of data-augmented networks are equivariant *(Gerken and Kessel 2024; Nordenfors and Flinth 2024)*
  - $\rightarrow$ How does this form of equivariance differ?
- Can study **biases** and the **influence of hyperparameters** on equivariant NNs through the NTK *(Xiao, Pennington, and Schoenholz 2020; Bowman and Montufar 2022)*

Why is that interesting?

- Can compare equivariant architectures with their non-equivariant counterparts **in a parameter-independent setting**
- Can compare **data augmentation** $\leftrightarrow$ **equivariant architectures**
  - Ensembles of data-augmented networks are equivariant *(Gerken and Kessel 2024; Nordenfors and Flinth 2024)*
  - $\rightarrow$ How does this form of equivariance differ?
- Can study **biases** and the **influence of hyperparameters** on equivariant NNs through the NTK *(Xiao, Pennington, and Schoenholz 2020; Bowman and Montufar 2022)*
- Obtain **equivariant Kernel methods**

On the practical side

On the practical side

Explicit expressions for

- **roto-translations** $G = C_4 \ltimes \mathbb{R}^2$ in the plane

On the practical side

Explicit expressions for

- **roto-translations** $G = C_4 \ltimes \mathbb{R}^2$ in the plane
- **3d-rotations** $G = \mathrm{SO}(3)$ on the sphere S^2
  via **spherical convolutions**

On the practical side

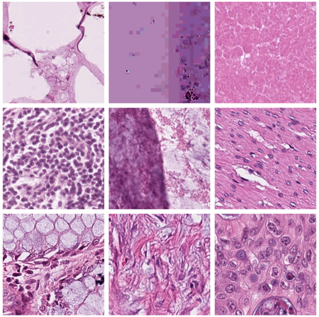
Explicit expressions for

- **roto-translations** $G = C_4 \ltimes \mathbb{R}^2$ in the plane
- **3d-rotations** $G = \mathrm{SO}(3)$ on the sphere S^2 via **spherical convolutions**

Provide **implementations** in the `neural-tangents` library *(Novak et al. 2020)*

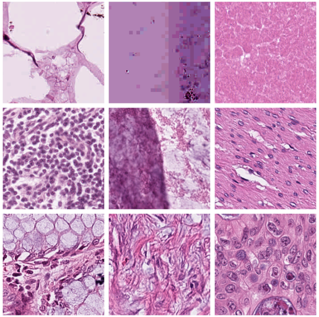
∞-width limit in practice: Histological image classification

∞-width limit in practice: Histological image classification



9 classes of microscopical tissue images *(Kather, Halama, and Marx 2018)*

∞-width limit in practice: Histological image classification



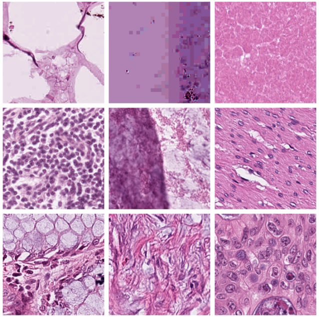
9 classes of microscopical tissue images *(Kather, Halama, and Marx 2018)*

CNN

vs.

$C_4 \ltimes \mathbb{R}^2$ GCNN

∞-width limit in practice: Histological image classification

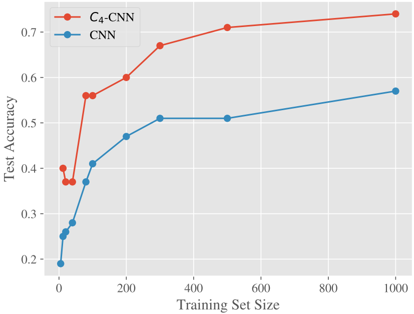


9 classes of microscopical tissue images *(Kather, Halama, and Marx 2018)*

CNN

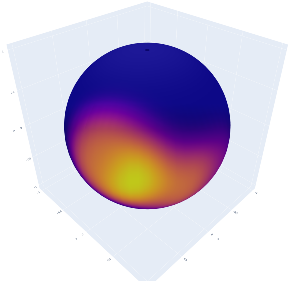
vs.

$C_4 \ltimes \mathbb{R}^2$ **GCNN**



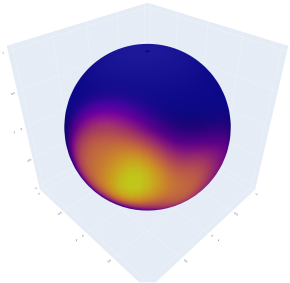
∞-width limit in practice: Molecular Energy Prediction on QM9

∞-width limit in practice: Molecular Energy Prediction on QM9



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

∞-width limit in practice: Molecular Energy Prediction on QM9



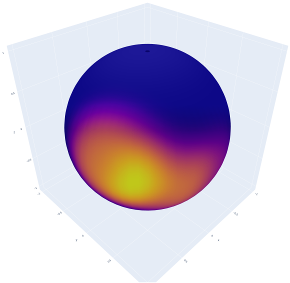
Atoms' environments are represented as signals on the sphere *(Esteves, Slotine, and Makadia 2023)*

MLP

vs.

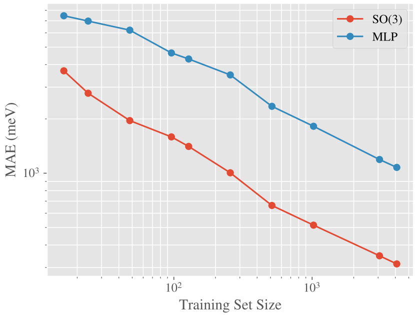
$\mathrm{SO}(3)$ **GCNN**

∞-width limit in practice: Molecular Energy Prediction on QM9

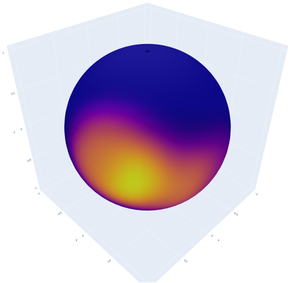


Atoms' environments are represented as signals on the sphere *(Esteves, Slotine, and Makadia 2023)*

MLP
vs.
$\mathrm{SO}(3)$ **GCNN**



∞-width limit in practice: Molecular Energy Prediction on QM9

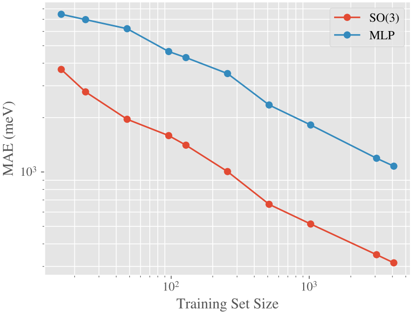


Atoms' environments are represented as signals on the sphere *(Esteves, Slotine, and Makadia 2023)*

MLP

vs.

$\mathrm{SO}(3)$ **GCNN**



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

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

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

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

Can **construct a GCNN** s.t.

$$\Theta^{\mathrm{GC}}(f, f') = \underbrace{\frac{1}{\mathrm{vol}(G)} \int_G \mathrm{d}g\ \Theta^{\mathrm{MLP}}(f, \rho_{\mathrm{reg}}(g) f')}_{\text{Effective MLP kernel under data augmentation}}$$

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

Can **construct a GCNN** s.t.

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

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

➡ **mean** of an ensemble of data augmented MLPs equals the **mean** of an ensemble of GCNNs **at all training times** t.

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

Can **construct a GCNN** s.t.

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

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

➡ **mean** of an ensemble of data augmented MLPs equals the **mean** of an ensemble of GCNNs **at all training times** t.

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

Can **construct a GCNN** s.t.

$$\Theta^{\mathrm{GC}}(f, f') = \underbrace{\frac{1}{\mathrm{vol}(G)} \int_G \mathrm{d}g\, \Theta^{\mathrm{MLP}}(f, \rho_{\mathrm{reg}}(g) f')}_{\text{Effective MLP kernel under data augmentation}}$$

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

➡ **mean** of an ensemble of data augmented MLPs equals the **mean** of an ensemble of GCNNs **at all training times** t.

Similar result for data augmented CNN $\leftrightarrow$ $C_4 \ltimes \mathbb{R}^2$ GCNN

Architecture correspondence

Architecture correspondence

$$\mathrm{FC}^{(1)} \longrightarrow \sigma \longrightarrow \mathrm{FC}^{(3)} \longrightarrow \sigma \longrightarrow \cdots \longrightarrow \mathrm{FC}^{(L)}$$

$$\mathrm{Lifting} \longrightarrow \sigma \longrightarrow \mathrm{GConv}^{(3)} \longrightarrow \sigma \longrightarrow \cdots \longrightarrow \mathrm{GConv}^{(L)} \longrightarrow \mathrm{GPool}$$

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**

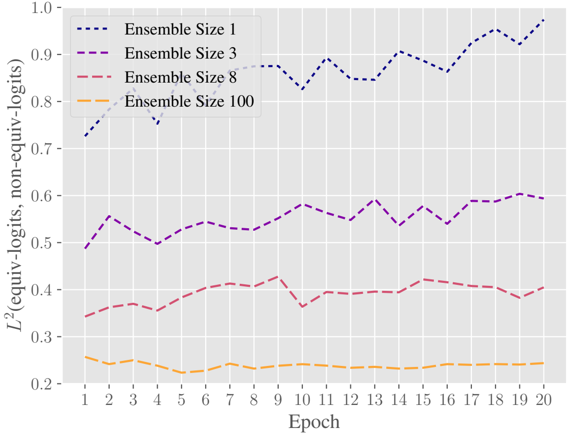
- Data augmented CNN vs $C_4 \ltimes \mathbb{R}^2$ GCNN on **MNIST**

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 mean logits on **out-of-distribution** data

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 mean logits on **out-of-distribution** data



Summing up

Summing up

- **Extended** the Neural Tangent Kernel theory **to equivariant neural networks**

Summing up

- **Extended** the Neural Tangent Kernel theory **to equivariant neural networks**
- Provide explicit expressions for **roto-translations** and **3d-rotations**

Summing up

- **Extended** the Neural Tangent Kernel theory **to equivariant neural networks**
- Provide explicit expressions for **roto-translations** and **3d-rotations**
- Showed how this analytic tool can be used to **analyze the connection between data augmentation and GCNNs**

Summing up

- **Extended** the Neural Tangent Kernel theory **to equivariant neural networks**
- Provide explicit expressions for **roto-translations** and **3d-rotations**
- Showed how this analytic tool can be used to **analyze the connection between data augmentation and GCNNs**

What's next?

Summing up

- **Extended** the Neural Tangent Kernel theory **to equivariant neural networks**
- Provide explicit expressions for **roto-translations** and **3d-rotations**
- Showed how this analytic tool can be used to **analyze the connection between data augmentation and GCNNs**

What's next?

- Relations to Quantum Field Theory *(Banta et al. 2024)*

Summing up

- **Extended** the Neural Tangent Kernel theory **to equivariant neural networks**
- Provide explicit expressions for **roto-translations** and **3d-rotations**
- Showed how this analytic tool can be used to **analyze the connection between data augmentation and GCNNs**

What's next?

- Relations to Quantum Field Theory *(Banta et al. 2024)*
- Trainability and generalization regimes of equivariant NNs *(Xiao, Pennington, and Schoenholz 2020)*

Summing up

- **Extended** the Neural Tangent Kernel theory **to equivariant neural networks**
- Provide explicit expressions for **roto-translations** and **3d-rotations**
- Showed how this analytic tool can be used to **analyze the connection between data augmentation and GCNNs**

What's next?

- Relations to Quantum Field Theory *(Banta et al. 2024)*
- Trainability and generalization regimes of equivariant NNs *(Xiao, Pennington, and Schoenholz 2020)*
- equivariant graph NNs

Do you want to know more?