Emergent Equivariance in Deep Ensembles

Jan E. Gerken

CHALMERS
UNIVERSITY OF TECHNOLOGY

WASP WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM

Talk at the
One World Seminar Series on the Mathematics of Machine Learning

Based on joint work with
Pan Kessel

Motivation

Symmetries in ML

Many learning problems are symmetric w.r.t. transformations by a symmetry group G

- G acts with some representation $\rho_X : G \to \mathrm{GL}(X)$ on the inputs $x_i \in X$
- G acts with some representation $\rho_Y : G \to \mathrm{GL}(Y)$ on the outputs $y_i \in Y$

Symmetries in ML

Many learning problems are symmetric w.r.t. transformations by a symmetry group G

- G acts with some representation $\rho_X : G \longrightarrow \mathrm{GL}(X)$ on the inputs $x_i \in X$
- G acts with some representation $\rho_Y : G \longrightarrow \mathrm{GL}(Y)$ on the outputs $y_i \in Y$
- In a symmetric learning problem, we have

$$(x, y) \in \mathcal{D} \quad \Rightarrow \quad (\rho_X(g)x, \rho_Y(g)y) \in \mathcal{D} \quad \forall g \in G$$

- Hence, the map $f : x \mapsto y$ satisfies

$$f(\rho_X(g)x) = \rho_Y(g)f(x) \qquad \forall g \in G$$

Symmetries in ML

Many learning problems are symmetric w.r.t. transformations by a symmetry group G

- G acts with some representation $\rho_X : G \to \mathrm{GL}(X)$ on the inputs $x_i \in X$
- G acts with some representation $\rho_Y : G \to \mathrm{GL}(Y)$ on the outputs $y_i \in Y$
- In a symmetric learning problem, we have

$$(x, y) \in \mathcal{D} \quad \Rightarrow \quad (\rho_X(g)x, \rho_Y(g)y) \in \mathcal{D} \quad \forall g \in G$$

- Hence, the map $f : x \mapsto y$ satisfies

$$f(\rho_X(g)x) = \rho_Y(g)f(x) \qquad \forall g \in G$$

- If $\rho_Y \equiv \mathbb{1}$ then we call f ***invariant***, otherwise ***equivariant***

Many learning problems are symmetric w.r.t. transformations by a symmetry group G

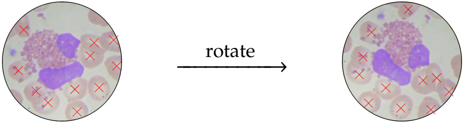
- G acts with some representation $\rho_X : G \to \mathrm{GL}(X)$ on the inputs $x_i \in X$
- G acts with some representation $\rho_Y : G \to \mathrm{GL}(Y)$ on the outputs $y_i \in Y$
- In a symmetric learning problem, we have

$$(x, y) \in \mathcal{D} \quad \Rightarrow \quad (\rho_X(g)x, \rho_Y(g)y) \in \mathcal{D} \quad \forall g \in G$$

- Hence, the map $f : x \mapsto y$ satisfies

$$f(\rho_X(g)x) = \rho_Y(g)f(x) \qquad \forall g \in G$$

- If $\rho_Y \equiv \mathbb{1}$ then we call f ***invariant***, otherwise ***equivariant***



Data augmentation

In ***data augmentation***, we train on an enlarged training dataset:

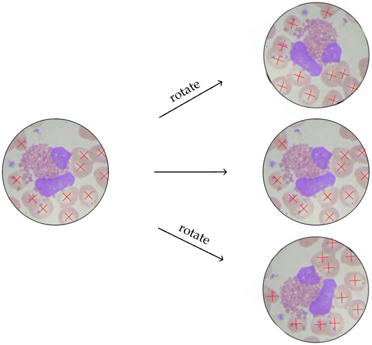
$$\mathcal{D} = \{(x_i, y_i) \mid i = 1, \ldots, N\} \qquad \longrightarrow \qquad \mathcal{D} = \bigcup_{g \in G} \{(\rho_X(g) x_i, \rho_Y(g) y_i) \mid i = 1, \ldots, N\}$$

Data augmentation

In ***data augmentation***, we train on an enlarged training dataset:

$$\mathcal{D} = \{(x_i, y_i) \mid i = 1, \ldots, N\} \qquad \longrightarrow \qquad \mathcal{D} = \bigcup_{g \in G} \{(\rho_X(g)x_i, \rho_Y(g)y_i) \mid i = 1, \ldots, N\}$$

For instance for blood cells:



Data augmentation

Data augmentation is

- Straightforward to implement
- Works with any model architecture

Data augmentation

Data augmentation is

- Straightforward to implement
- Works with any model architecture

However, it also

- Does not yield an exactly equivariant model
- Increases training time to learn equivariance
- Particularly long training needed for continuous symmetry groups

Data augmentation

Data augmentation is

- Straightforward to implement
- Works with any model architecture

However, it also

- Does not yield an exactly equivariant model
- Increases training time to learn equivariance
- Particularly long training needed for continuous symmetry groups

$\Rightarrow$ Good if equivariance is not required exactly or group action complicated.
$\Rightarrow$ Not suitable if symmetry needs to be exact, e.g. science application

Data augmentation

Data augmentation is

- Straightforward to implement
- Works with any model architecture

However, it also

- Does not yield an exactly equivariant model
- Increases training time to learn equivariance
- Particularly long training needed for continuous symmetry groups

$\Rightarrow$ Good if equivariance is not required exactly or group action complicated.
$\Rightarrow$ Not suitable if symmetry needs to be exact, e.g. science application

Goal: Investigate data augmentation theoretically.
What are the symmetry properties of networks trained with augmentation?

Strategy

Consider infinitely wide neural networks

Strategy

Consider infinitely wide neural networks

- In this limit, the mean output μ_t after training time t can be computed analytically

$$\mu_t(\underbrace{x}_{\text{test point}}) = \overbrace{\Theta}^{\text{neural tangent kernel}}(x, \underbrace{X}_{\text{train data}})\Theta(X, X)^{-1}(\mathbb{1} - e^{-\underbrace{\eta}_{\text{learning rate}}\Theta(X,X)t})\underbrace{Y}_{\text{train labels}}$$

Strategy

Consider infinitely wide neural networks

- In this limit, the mean output μ_t after training time t can be computed analytically
- Training data explicit $\Rightarrow$ can argue about training with augmented data

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

augmented data

augmented labels

Strategy

Consider infinitely wide neural networks

- In this limit, the mean output μ_t after training time t can be computed analytically
- Training data explicit $\Rightarrow$ can argue about training with augmented data
- Compute the mean output on a transformed sample

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

augmented data

augmented labels

Strategy

Consider infinitely wide neural networks

- In this limit, the mean output μ_t after training time t can be computed analytically
- Training data explicit $\Rightarrow$ can argue about training with augmented data
- Compute the mean output on a transformed sample

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

augmented data

augmented labels

- The mean prediction corresponds to an ensemble prediction

$$\mu_t(x) = \mathbb{E}_{\theta_0 \sim \text{initializations}}[f_{\theta_t}(x)] = \lim_{n\to\infty} \underbrace{\frac{1}{n} \sum_{\theta_0=\text{init}_1}^{\text{init}_n} f_{\theta_t}(x)}_{\text{mean prediction of deep ensemble}}$$

Background: Neural Tangent Kernels and Wide Neural Networks

Standard parametrization

- We usually parametrize MLPs as

$$z^{(\ell)} = W^{\ell} f^{\ell}(x)\,, \quad f^{(\ell)}(x) = \sigma(z^{(\ell-1)}(x))\,, \quad W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}\,, \quad W_{ij}^{(\ell)} \sim \mathcal{N}\left(0, \frac{1}{n_{\ell-1}}\right)$$

Standard parametrization

- We usually parametrize MLPs as

$$z^{(\ell)} = W^{\ell} f^{\ell}(x)\,, \quad f^{(\ell)}(x) = \sigma(z^{(\ell-1)}(x))\,, \quad W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}\,, \quad W_{ij}^{(\ell)} \sim \mathcal{N}\left(0, \frac{1}{n_{\ell-1}}\right)$$

- In this parametrization, the gradients scale as

$$\frac{\partial f}{\partial W_{ij}^{(\ell)}} \sim \begin{cases} \mathcal{O}\left(\frac{1}{\sqrt{n}}\right) & \text{if} \quad \ell < L \\ \mathcal{O}\left(1\right) & \text{if} \quad \ell = L \end{cases}$$

NTK parametrization

[Jacot et al. 2020]

Standard parametrization:

$$z^{(\ell)} = W^{\ell} f^{\ell}(x)\,, \qquad W_{ij}^{(\ell)} \sim \mathcal{N}\left(0, \frac{1}{n_{\ell-1}}\right)$$

- In ***NTK parametrization***, we instead use

$$z^{(\ell)} = \frac{1}{\sqrt{n_{\ell-1}}} W^{\ell} f^{\ell}(x)\,, \qquad W_{ij}^{(\ell)} \sim \mathcal{N}\left(0, 1\right)$$

NTK parametrization

[Jacot et al. 2020]

Standard parametrization:

$$z^{(\ell)} = W^{\ell} f^{\ell}(x)\,, \qquad W_{ij}^{(\ell)} \sim \mathcal{N}\left(0, \frac{1}{n_{\ell-1}}\right)$$

- In ***NTK parametrization***, we instead use

$$z^{(\ell)} = \frac{1}{\sqrt{n_{\ell-1}}} W^{\ell} f^{\ell}(x)\,, \qquad W_{ij}^{(\ell)} \sim \mathcal{N}(0, 1)$$

- This does not change the output distribution at initialization

NTK parametrization

[Jacot et al. 2020]

Standard parametrization:

$$z^{(\ell)} = W^{\ell} f^{\ell}(x)\,, \qquad W_{ij}^{(\ell)} \sim \mathcal{N}\left(0, \frac{1}{n_{\ell-1}}\right)$$

- In ***NTK parametrization***, we instead use

$$z^{(\ell)} = \frac{1}{\sqrt{n_{\ell-1}}} W^{\ell} f^{\ell}(x)\,, \qquad W_{ij}^{(\ell)} \sim \mathcal{N}\,(0, 1)$$

- This does not change the output distribution at initialization
- However, the gradients now scale as

$$\frac{\partial f}{\partial W_{ij}^{(\ell)}} \sim \begin{cases} \mathcal{O}\left(\frac{1}{\sqrt{n}}\right) & \text{if} \quad \ell = 1 \text{ or } \ell = L \\ \mathcal{O}\left(\frac{1}{n}\right) & \text{if} \quad 1 < \ell < L \end{cases}$$

NTK parametrization

[Jacot et al. 2020]

Standard parametrization:

$$z^{(\ell)} = W^{\ell} f^{\ell}(x)\,, \qquad W_{ij}^{(\ell)} \sim \mathcal{N}\left(0, \frac{1}{n_{\ell-1}}\right)$$

- In ***NTK parametrization***, we instead use

$$z^{(\ell)} = \frac{1}{\sqrt{n_{\ell-1}}} W^{\ell} f^{\ell}(x)\,, \qquad W_{ij}^{(\ell)} \sim \mathcal{N}(0, 1)$$

- This does not change the output distribution at initialization
- However, the gradients now scale as

$$\frac{\partial f}{\partial W_{ij}^{(\ell)}} \sim \begin{cases} \mathcal{O}\left(\frac{1}{\sqrt{n}}\right) & \text{if} \quad \ell = 1 \text{ or } \ell = L \\ \mathcal{O}\left(\frac{1}{n}\right) & \text{if} \quad 1 < \ell < L \end{cases}$$

- In this parametrization, the gradients vanish as $n \to \infty$

NTK parametrization

Standard parametrization:

$$z^{(\ell)} = W^{\ell} f^{\ell}(x)\,, \qquad W_{ij}^{(\ell)} \sim \mathcal{N}\left(0, \frac{1}{n_{\ell-1}}\right)$$

- In ***NTK parametrization***, we instead use

$$z^{(\ell)} = \frac{1}{\sqrt{n_{\ell-1}}} W^{\ell} f^{\ell}(x)\,, \qquad W_{ij}^{(\ell)} \sim \mathcal{N}(0, 1)$$

- This does not change the output distribution at initialization
- However, the gradients now scale as

$$\frac{\partial f}{\partial W_{ij}^{(\ell)}} \sim \begin{cases} \mathcal{O}\left(\frac{1}{\sqrt{n}}\right) & \text{if} \quad \ell = 1 \text{ or } \ell = L \\ \mathcal{O}\left(\frac{1}{n}\right) & \text{if} \quad 1 < \ell < L \end{cases}$$

- In this parametrization, the gradients vanish as $n \to \infty$
- Hence, the dynamics becomes tractable

NTK parametrization

[Jacot et al. 2020]

Standard parametrization:

$$z^{(\ell)} = W^\ell f^\ell(x)\,, \qquad W_{ij}^{(\ell)} \sim \mathcal{N}\left(0, \frac{1}{n_{\ell-1}}\right)$$

- In ***NTK parametrization***, we instead use

$$z^{(\ell)} = \frac{1}{\sqrt{n_{\ell-1}}} W^\ell f^\ell(x)\,, \qquad W_{ij}^{(\ell)} \sim \mathcal{N}(0, 1)$$

- This does not change the output distribution at initialization
- However, the gradients now scale as

$$\frac{\partial f}{\partial W_{ij}^{(\ell)}} \sim \begin{cases} \mathcal{O}\left(\frac{1}{\sqrt{n}}\right) & \text{if} \quad \ell = 1 \text{ or } \ell = L \\ \mathcal{O}\left(\frac{1}{n}\right) & \text{if} \quad 1 < \ell < L \end{cases}$$

- In this parametrization, the gradients vanish as $n \to \infty$
- Hence, the dynamics becomes tractable
- We will assume NTK parametrization in the rest of the talk

Output distribution at initialization

[Neal 1995]

Neural network Gaussian process

When taking the layer width of a neural network to infinity sequentially, the preactivations $z_i^\ell(x)$ at initialization become a Gaussian process with mean zero.

Neural network Gaussian process

When taking the layer width of a neural network to infinity sequentially, the preactivations $z_i^{\ell}(x)$ at initialization become a Gaussian process with mean zero.

- The covariance function $K^{(\ell)}(x, x')\delta_{ij}$ is also known as the ***NNGP kernel***

Output distribution at initialization

[Neal 1995]

Neural network Gaussian process

When taking the layer width of a neural network to infinity sequentially, the preactivations $z_i^\ell(x)$ at initialization become a Gaussian process with mean zero.

- The covariance function $K^{(\ell)}(x, x')\delta_{ij}$ is also known as the ***NNGP kernel***
- The NNGP can be computed recursively layer-by-layer

Intuitively,

- The mean vanishes since

$$\begin{aligned}\mathbb{E}_{W^{(\ell)},\ldots,W^{(1)}}[z^{\ell}(x)] &= \frac{1}{\sqrt{n_{\ell-1}}}\mathbb{E}_{W^{(\ell)},\ldots,W^{(1)}}[W^{(\ell)}\sigma(z^{(\ell-1)}x)] \\ &= \frac{1}{\sqrt{n_{\ell-1}}}\underbrace{\mathbb{E}_{W^{(\ell)}}[W^{(\ell)}]}_{0}\mathbb{E}_{W^{(\ell-1)},\ldots,W^{(1)}}[\sigma(z^{(\ell-1)}x)]\end{aligned}$$

Output distribution at initialization

[Neal 1995]

Intuitively,

- The mean vanishes since

$$
\begin{aligned}
\mathbb{E}_{W^{(\ell)},\dots,W^{(1)}}[z^{\ell}(x)] &= \frac{1}{\sqrt{n_{\ell-1}}}\mathbb{E}_{W^{(\ell)},\dots,W^{(1)}}[W^{(\ell)}\sigma(z^{(\ell-1)}x)] \\
&= \frac{1}{\sqrt{n_{\ell-1}}}\underbrace{\mathbb{E}_{W^{(\ell)}}[W^{(\ell)}]}_{0}\mathbb{E}_{W^{(\ell-1)},\dots,W^{(1)}}[\sigma(z^{(\ell-1)}x)]
\end{aligned}
$$

- The preactivations are Gaussian due to the central limit theorem

$$
z_i^{\ell}(x) = \sqrt{n_{\ell-1}}\underbrace{\frac{1}{n_{\ell-1}}\sum_{j=1}^{n_\ell}}_{\text{mean}}\underbrace{W_{ij}^{(\ell)}\sigma(z_j^{(\ell-1)}(x))}_{\text{i.i.d.}} \sim \mathcal{N}(0, \underbrace{\mathrm{Cov}(z_i^{(\ell)}, z_j^{(\ell)})}_{\text{NNGP}}) \quad \text{as} \quad n_\ell \to \infty
$$

Empirical NTK

- Consider continuous gradient descent

$$\frac{\mathrm{d}\theta_\mu}{\mathrm{d}t} = -\eta \frac{\partial \mathcal{L}(f_\theta, \mathcal{D})}{\partial \theta_\mu}$$

under the loss

$$\mathcal{L}(f_\theta, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} l(f_\theta(x_i), y_i)$$

Empirical NTK

- Consider continuous gradient descent

$$\frac{\mathrm{d}\theta_\mu}{\mathrm{d}t} = -\eta \frac{\partial \mathcal{L}(f_\theta, \mathcal{D})}{\partial \theta_\mu}$$

  under the loss

$$\mathcal{L}(f_\theta, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} l(f_\theta(x_i), y_i)$$

- Then, the network evolves according to

$$\frac{\mathrm{d}f_\theta(x)}{\mathrm{d}t} = -\frac{\eta}{N} \sum_{i=1}^{N} \sum_{\mu} \frac{\partial f(x)}{\partial \theta_\mu} \frac{\partial f(x_i)}{\partial \theta_\mu} \frac{\partial l(f_\theta(x_i), y_i)}{\partial f}$$

Empirical NTK

- Consider continuous gradient descent

$$\frac{\mathrm{d}\theta_\mu}{\mathrm{d}t} = -\eta \frac{\partial \mathcal{L}(f_\theta, \mathcal{D})}{\partial \theta_\mu}$$

  under the loss

$$\mathcal{L}(f_\theta, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} l(f_\theta(x_i), y_i)$$

- Then, the network evolves according to

$$\frac{\mathrm{d}f_\theta(x)}{\mathrm{d}t} = -\frac{\eta}{N} \sum_{i=1}^{N} \sum_{\mu} \frac{\partial f(x)}{\partial \theta_\mu} \frac{\partial f(x_i)}{\partial \theta_\mu} \frac{\partial l(f_\theta(x_i), y_i)}{\partial f}$$

- Hence, the training is driven by the ***empirical neural tangent kernel (NTK)***

$$\Theta^{\theta}_{ij}(x, x') = \sum_{\mu} \frac{\partial f_i(x)}{\partial \theta_\mu} \frac{\partial f_j(x')}{\partial \theta_\mu}$$

NTK at initialization

Deterministic NTK [Jacot et al. 2020]

When taking the layer widths to infinity sequentially, the empirical NTK $\Theta^{\theta}_{ij}(x, x')$ at initialization converges in probability to a deterministic kernel $\Theta(x, x')\delta_{ij}$

NTK at initialization

Deterministic NTK [Jacot et al. 2020]

When taking the layer widths to infinity sequentially, the empirical NTK $\Theta^{\theta}_{ij}(x, x')$ at initialization converges in probability to a deterministic kernel $\Theta(x, x')\delta_{ij}$

- The deterministic kernel is again given in terms of a recursion over layers

NTK at initialization

Deterministic NTK [Jacot et al. 2020]

When taking the layer widths to infinity sequentially, the empirical NTK $\Theta^{\theta}_{ij}(x, x')$ at initialization converges in probability to a deterministic kernel $\Theta(x, x')\delta_{ij}$

- The deterministic kernel is again given in terms of a recursion over layers
- For most common architectures, this recursion can be performed explicitly, e.g. using `neural-tangents` Python package [Novak et al. 2020]

Frozen NTK

Freezing of NTK

For a nonlinearity which is Lipschitz, twice differentiable and has bounded second derivative,

$$\Theta_{ij}^{\theta_t}(x, x') \to \Theta(x, x')\delta_{ij}$$

uniformly in t as the layer widths go to infinity sequentially.

Frozen NTK

[Jacot et al. 2020]

Freezing of NTK

For a nonlinearity which is Lipschitz, twice differentiable and has bounded second derivative,

$$\Theta^{\theta_t}_{ij}(x, x') \to \Theta(x, x')\delta_{ij}$$

uniformly in t as the layer widths go to infinity sequentially.

- Intuitively, this happens because the weight updates vanish in the limit $n \to \infty$

Frozen NTK

[Jacot et al. 2020]

Freezing of NTK

For a nonlinearity which is Lipschitz, twice differentiable and has bounded second derivative,

$$\Theta_{ij}^{\theta_t}(x, x') \to \Theta(x, x')\delta_{ij}$$

uniformly in t as the layer widths go to infinity sequentially.

- Intuitively, this happens because the weight updates vanish in the limit $n \to \infty$
- However, the network still learns because the number of neurons grows, leading to a non-zero collective effect

Gradient descent at inifinite width

[Jacot et al. 2020]

- Consider continuous gradient descent training under the MSE loss

$$\frac{\mathrm{d}f_\theta(x)}{\mathrm{d}t} = -\eta \sum_{i=1}^{N} \Theta^{\theta_t}(x, x_i)(f_\theta(x_i) - y_i)$$

Gradient descent at inifinite width

[Jacot et al. 2020]

- Consider continuous gradient descent training under the MSE loss

$$\frac{\mathrm{d}f_\theta(x)}{\mathrm{d}t} = -\eta \sum_{i=1}^{N} \Theta^{\theta_t}(x, x_i)(f_\theta(x_i) - y_i)$$

- At infinite width, the NTK freezes and is independent of θ_t

$$\frac{\mathrm{d}f_{\theta_t}(x)}{\mathrm{d}t} = -\eta \sum_{i=1}^{N} \Theta(x, x_i)(f_\theta(x_i) - y_i)$$

Gradient descent at inifinite width

[Jacot et al. 2020]

- Consider continuous gradient descent training under the MSE loss

$$\frac{\mathrm{d}f_\theta(x)}{\mathrm{d}t} = -\eta \sum_{i=1}^{N} \Theta^{\theta_t}(x, x_i)(f_\theta(x_i) - y_i)$$

- At infinite width, the NTK freezes and is independent of θ_t

$$\frac{\mathrm{d}f_{\theta_t}(x)}{\mathrm{d}t} = -\eta \sum_{i=1}^{N} \Theta(x, x_i)(f_\theta(x_i) - y_i)$$

- This ODE can be solved analytically, resulting in

$$f_{\theta_t}(x) = \Theta(x, X)\Theta(X, X)^{-1}(e^{-\eta\Theta(X,X)t} - \mathbb{1})(f_{\theta_0}(X) - Y) + f_{\theta_0}(x)$$

where we have introduced

$$X_i = x_i\,, \qquad Y_i = y_i\,, \qquad \Theta(X, X)_{ij} = \Theta(x_i, x_j)$$

Gradient descent at inifinite width

[Jacot et al. 2020]

As a linear combination of the GPs f_{θ_0}, the prediction

$$f_{\theta_t}(x) = \Theta(x, X)\Theta(X, X)^{-1}(e^{-\eta\Theta(X,X)t} - \mathbb{1})(f_{\theta_0}(X) - Y) + f_{\theta_0}(x)$$

is a GP.

Gradient descent at inifinite width

[Jacot et al. 2020]

As a linear combination of the GPs f_{θ_0}, the prediction

$$f_{\theta_t}(x) = \Theta(x, X)\Theta(X, X)^{-1}(e^{-\eta\Theta(X,X)t} - \mathbb{1})(f_{\theta_0}(X) - Y) + f_{\theta_0}(x)$$

is a GP.

- The mean function is given by

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

Gradient descent at inifinite width

[Jacot et al. 2020]

As a linear combination of the GPs f_{θ_0}, the prediction

$$f_{\theta_t}(x) = \Theta(x, X)\Theta(X, X)^{-1}(e^{-\eta\Theta(X,X)t} - \mathbb{1})(f_{\theta_0}(X) - Y) + f_{\theta_0}(x)$$

is a GP.

- The mean function is given by

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

- The covariance function is given by

$$\Sigma_t(x, x') = K(x, x') + \Theta(x, X)\,\Theta^{-1}\,(\mathbb{1} - e^{-\eta\Theta t})\,K\,(\mathbb{1} - e^{-\eta\Theta t})\,\Theta^{-1}\,\Theta(X, x')$$
$$- \left(\Theta(x, X)\,\Theta^{-1}\,(\mathbb{1} - e^{-\eta\Theta t})\,K(X, x') + \text{h.c.}\right)$$

Gradient descent at inifinite width

[Jacot et al. 2020]

As a linear combination of the GPs f_{θ_0}, the prediction

$$f_{\theta_t}(x) = \Theta(x,X)\Theta(X,X)^{-1}(e^{-\eta\Theta(X,X)t} - \mathbb{1})(f_{\theta_0}(X) - Y) + f_{\theta_0}(x)$$

is a GP.

- The mean function is given by

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

- The covariance function is given by

$$\Sigma_t(x,x') = K(x,x') + \Theta(x,X)\,\Theta^{-1}\,(\mathbb{1} - e^{-\eta\Theta t})\,K\,(\mathbb{1} - e^{-\eta\Theta t})\,\Theta^{-1}\,\Theta(X,x')$$
$$- \left(\Theta(x,X)\,\Theta^{-1}\,(\mathbb{1} - e^{-\eta\Theta t})\,K(X,x') + \text{h.c.}\right)$$

- On training samples, at $t \to \infty$, we predict the training labels

$$\lim_{t\to\infty} \mu_t(X) = Y$$

$$\lim_{t\to\infty} \Sigma_t(X,X) = 0$$

Emergent Equivariance for Large-Width Deep Ensembles

Data augmentation

Consider a learning problem which is symmetric w.r.t. some group G

- Assume that we act with representations ρ_X and ρ_Y of G on samples and labels,

$$\rho_X(g)x\,, \qquad \rho_Y(g)y$$

Data augmentation

Consider a learning problem which is symmetric w.r.t. some group G

- Assume that we act with representations ρ_X and ρ_Y of G on samples and labels,

$$\rho_X(g)x\,, \qquad \rho_Y(g)y$$

- We augment a training dataset $\mathcal{T}$ by the group orbits of the training samples, yielding the augmented dataset

$$\mathcal{T}_{\text{aug}} = (X, Y) = \{(\rho_X(g)x, \rho_Y(g)y) \,|\, g \in G, (x, y) \in \mathcal{T}\}$$

Data augmentation

Consider a learning problem which is symmetric w.r.t. some group G

- Assume that we act with representations ρ_X and ρ_Y of G on samples and labels,

$$\rho_X(g)x\,, \qquad \rho_Y(g)y$$

- We augment a training dataset $\mathcal{T}$ by the group orbits of the training samples, yielding the augmented dataset

$$\mathcal{T}_{\text{aug}} = (X, Y) = \{(\rho_X(g)x, \rho_Y(g)y) \mid g \in G, (x, y) \in \mathcal{T}\}$$

- For finite groups, this means that transforming the training samples is equivalent to permuting them

$$\rho_X(g)x_i = x_{\pi_g(i)}\,, \qquad \rho_Y(g)y_i = y_{\pi_g(i)}\,, \qquad \pi \in S_N$$

Data augmentation

Consider a learning problem which is symmetric w.r.t. some group G

- Assume that we act with representations ρ_X and ρ_Y of G on samples and labels,

$$\rho_X(g)x\,, \qquad \rho_Y(g)y$$

- We augment a training dataset $\mathcal{T}$ by the group orbits of the training samples, yielding the augmented dataset

$$\mathcal{T}_{\text{aug}} = (X, Y) = \{(\rho_X(g)x, \rho_Y(g)y) \,|\, g \in G, (x, y) \in \mathcal{T}\}$$

- For finite groups, this means that transforming the training samples is equivalent to permuting them

$$\rho_X(g)x_i = x_{\pi_g(i)}\,, \qquad \rho_Y(g)y_i = y_{\pi_g(i)}\,, \qquad \pi \in S_N$$

- On the training set X, this can be written as a permutation matrix $\Pi(g)$ and hence

$$\Theta(\rho_X(g)X, \rho_X(g)X) = \Pi(g)\Theta(X, X)(\Pi(g))^\top$$

Kernel transformation

The transformation of the inputs induces a transformation of the kernels

$$K(x, x') \to K(\rho_X(g)x, \rho_X(g)x')$$
$$\Theta(x, x') \to \Theta(\rho_X(g)x, \rho_X(g)x')$$

Kernel transformation

The transformation of the inputs induces a transformation of the kernels

$$K(x, x') \to K(\rho_X(g)x, \rho_X(g)x')$$
$$\Theta(x, x') \to \Theta(\rho_X(g)x, \rho_X(g)x')$$

Kernel transformation

The neural tangent kernel Θ as well as the NNGP kernel K transform according to

$$\Theta(\rho_X(g)x, \rho_X(g)x') = \rho_K(g)\Theta(x, x')\rho_K^\top(g)\,,$$
$$K(\rho_X(g)x, \rho_X(g)x') = \rho_K(g)K(x, x')\rho_K^\top(g)\,,$$

for all $g \in G$ and $x, x' \in X$, where ρ_K is a transformation acting on the spatial dimensions of the kernels. If the kernels do not have spatial axes, $\rho_K = \mathbb{1}$.

Sketch of proof

- Prove inductively over layers

Sketch of proof

- Prove inductively over layers
- For nonlinearities, CNN-, fully-connected- and flattening layers

Sketch of proof

- Prove inductively over layers
- For nonlinearities, CNN-, fully-connected- and flattening layers
- Example: Induction step for NNGP of CNN layer
  Assume

$$K_{\ell-1}^{a,a'}(\rho_X(g)x, \rho_X(g)x') = K_{\ell-1}^{\rho^{-1}(g)a, \rho^{-1}(g)a'}(x, x') \,,$$

Sketch of proof

- Prove inductively over layers
- For nonlinearities, CNN-, fully-connected- and flattening layers
- Example: Induction step for NNGP of CNN layer
  Assume
  $$K_{\ell-1}^{a,a'}(\rho_X(g)x, \rho_X(g)x') = K_{\ell-1}^{\rho^{-1}(g)a,\rho^{-1}(g)a'}(x,x'),$$
  then
  $$\begin{aligned} K_{\ell}^{a,a'}(\rho_X(g)x, \rho_X(g)x') &= \sum_{\tilde{a}} K_{\ell-1}^{a+\tilde{a},a'+\tilde{a}}(\rho_X(g)x, \rho_X(g)x') \\ &= \sum_{\tilde{a}} K_{\ell-1}^{\rho^{-1}(g)(a+\tilde{a}),\rho^{-1}(g)(a'+\tilde{a})}(x,x') \\ &= \sum_{\tilde{a}} K_{\ell-1}^{\rho^{-1}(g)a+\tilde{a},\rho^{-1}(g)a'+\tilde{a}}(x,x') \\ &= K_{\ell}^{\rho^{-1}(g)a,\rho^{-1}(g)a'}(x,x') \\ &= (\rho_K(g)K_\ell(x,x')\rho_K^\top(g))^{a,a'} \end{aligned}$$

Permutation shift

Consider an MLP

- Kernel invariance

$$\Theta(\rho_X(g)x, \rho_X(g)x') = \Theta(x, x') \qquad \Rightarrow \qquad \Theta(\rho_X(g)x, x') = \Theta(x, \rho_X^{-1}(g)x')$$

Permutation shift

Consider an MLP

- Kernel invariance

$$\Theta(\rho_X(g)x, \rho_X(g)x') = \Theta(x, x') \qquad \Rightarrow \qquad \Theta(\rho_X(g)x, x') = \Theta(x, \rho_X^{-1}(g)x')$$

- Therefore, for a permutation of training samples associate to g

$$\begin{aligned}
\Pi(g)\Theta(X, X) &= \Theta(\rho_X(g)X, X) \\
&= \Theta(X, \rho_X^{-1}(g)X) \\
&= \Theta(X, X)(\Pi^{-1}(g))^\top \\
&= \Theta(X, X)\Pi(g)
\end{aligned}$$

Permutation shift

Permutation shift

Data augmentation implies that the permutation group action Π commutes with any matrix-valued analytical function F involving the Gram matrices of the NNGP and NTK as well as their inverses:

$$\begin{aligned}&\Pi(g)F(\Theta,\Theta^{-1},K,K^{-1})\\&=\rho_K(g)F(\Theta,\Theta^{-1},K,K^{-1})\Pi(g)\rho_K^\top(g)\,.\end{aligned}$$

Sketch of proof

- Proof permutation shift separately for Θ, Θ^{-1}, K, K^{-1} and all powers of these

Sketch of proof

- Proof permutation shift separately for Θ, Θ^{-1}, K, K^{-1} and all powers of these
- Consider e.g. permutation shift for Θ^{-1}. Then (summation over l implied)

$$\begin{aligned}\Theta(X, \rho_X(g)X)_{il}\,[\Theta(X, \rho_X(g)X)]^{-1}_{lj} &= \delta_{ij}\\ \Theta(X, X)_{i,\pi_g(l)}\,[\Theta(X, \rho_X(g)X)]^{-1}_{lj} &= \delta_{ij}\\ \Theta(X, X)_{il}\,[\Theta(X, \rho_X(g)X)]^{-1}_{\pi_g^{-1}(l),j} &= \delta_{ij}\end{aligned}$$

Sketch of proof

- Proof permutation shift separately for Θ, Θ^{-1}, K, K^{-1} and all powers of these
- Consider e.g. permutation shift for Θ^{-1}. Then (summation over l implied)

$$\begin{aligned}
\Theta(X, \rho_X(g)X)_{il}\,[\Theta(X, \rho_X(g)X)]^{-1}_{lj} &= \delta_{ij}\\
\Theta(X, X)_{i,\pi_g(l)}\,[\Theta(X, \rho_X(g)X)]^{-1}_{lj} &= \delta_{ij}\\
\Theta(X, X)_{il}\,[\Theta(X, \rho_X(g)X)]^{-1}_{\pi_g^{-1}(l),j} &= \delta_{ij}
\end{aligned}$$

- By the uniqueness of the inverse, we have

$$\Theta(X, X)^{-1}_{lj} = [\Theta(X, \rho_X(g)X)]^{-1}_{\pi_g^{-1}(l),j} \quad \Leftrightarrow \quad \Theta(X, X)^{-1}_{\pi_g(l),j} = [\Theta(X, \rho_X(g)X)]^{-1}_{lj}$$

Sketch of proof

- Proof permutation shift separately for Θ, Θ^{-1}, K, K^{-1} and all powers of these
- Consider e.g. permutation shift for Θ^{-1}. Then (summation over l implied)

$$
\begin{aligned}
\Theta(X, \rho_X(g)X)_{il} \, [\Theta(X, \rho_X(g)X)]^{-1}_{lj} &= \delta_{ij} \\
\Theta(X, X)_{i,\,\pi_g(l)} \, [\Theta(X, \rho_X(g)X)]^{-1}_{lj} &= \delta_{ij} \\
\Theta(X, X)_{il} \, [\Theta(X, \rho_X(g)X)]^{-1}_{\pi_g^{-1}(l),\,j} &= \delta_{ij}
\end{aligned}
$$

- By the uniqueness of the inverse, we have

$$
\Theta(X, X)^{-1}_{lj} = [\Theta(X, \rho_X(g)X)]^{-1}_{\pi_g^{-1}(l),\,j} \quad \Leftrightarrow \quad \Theta(X, X)^{-1}_{\pi_g(l),\,j} = [\Theta(X, \rho_X(g)X)]^{-1}_{lj}
$$

- Similarly, one can show that

$$
\Theta(X, X)^{-1}_{i,\pi_g^{-1}(l)} = \left[\Theta(\rho_X^{-1}(g)X, X)\right]^{-1}_{il}
$$

Sketch of proof

- Proof permutation shift separately for Θ, Θ^{-1}, K, K^{-1} and all powers of these
- Consider e.g. permutation shift for Θ^{-1}. Then (summation over l implied)

$$
\begin{aligned}
\Theta(X, \rho_X(g)X)_{il} \left[\Theta(X, \rho_X(g)X)\right]^{-1}_{lj} &= \delta_{ij} \\
\Theta(X, X)_{i, \pi_g(l)} \left[\Theta(X, \rho_X(g)X)\right]^{-1}_{lj} &= \delta_{ij} \\
\Theta(X, X)_{il} \left[\Theta(X, \rho_X(g)X)\right]^{-1}_{\pi_g^{-1}(l), j} &= \delta_{ij}
\end{aligned}
$$

- By the uniqueness of the inverse, we have

$$
\Theta(X, X)^{-1}_{lj} = \left[\Theta(X, \rho_X(g)X)\right]^{-1}_{\pi_g^{-1}(l), j} \quad \Leftrightarrow \quad \Theta(X, X)^{-1}_{\pi_g(l), j} = \left[\Theta(X, \rho_X(g)X)\right]^{-1}_{lj}
$$

- Similarly, one can show that

$$
\Theta(X, X)^{-1}_{i, \pi_g^{-1}(l)} = \left[\Theta(\rho_X^{-1}(g)X, X)\right]^{-1}_{il}
$$

- Therefore

$$
\begin{aligned}
\left[\Theta(X, X)\right]^{-1}_{\pi_g(i), j} = \left[\Theta(X, \rho_X(g)X)\right]^{-1}_{ij} &= \rho_K(g) \left[\Theta(\rho_X^{-1}(g)X, X)\right]^{-1}_{ij} \rho_K^\top(g) \\
&= \rho_K(g) \Theta(X, X)^{-1}_{i, \pi_g^{-1}(j)} \rho_K^\top(g)
\end{aligned}
$$

Invariance of ensemble of MLPs

- Consider an ensemble of MLPs trained with data augmentation towards invariance

Invariance of ensemble of MLPs

- Consider an ensemble of MLPs trained with data augmentation towards invariance
- The mean prediction on a transformed test sample is given by

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

Invariance of ensemble of MLPs

- Consider an ensemble of MLPs trained with data augmentation towards invariance
- The mean prediction on a transformed test sample is given by

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

- By kernel transformation in the MLP case

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

Invariance of ensemble of MLPs

- Consider an ensemble of MLPs trained with data augmentation towards invariance
- The mean prediction on a transformed test sample is given by

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

- By kernel transformation in the MLP case

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

- By permutation shift

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

Invariance of ensemble of MLPs

- Consider an ensemble of MLPs trained with data augmentation towards invariance
- The mean prediction on a transformed test sample is given by

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

- By kernel transformation in the MLP case

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

- By permutation shift

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

- Due to data augmentation, the labels are invariant under group-permutations

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

Emergent equivariance of deep ensembles

Emergent equivariance of deep ensembles

The distribution of large-width ensemble members $f_\theta : X \to Y$ is equivariant with respect to the representations ρ_X and ρ_Y of the group G if data augmentation is applied. In particular, the ensemble prediction

$$\bar{f}_t(x) = \mathbb{E}_{\text{initializations}}[f_\theta(x)]$$

is equivariant,

$$\bar{f}_t(\rho_X(g)\,x) = \rho_Y(g)\,\bar{f}_t(x)\,,$$

for all $g \in G$. This result holds

1. at any training time t,
2. for any element of the input space $x \in X$.

- Prove by showing equivariance of μ_t and Σ_t

Experiments

Ising model

- Consider the 2d Ising model:
  A lattice L with spins $s_i \in \{+1, -1\}$ at each lattice site and energy

$$\mathcal{E} = -\frac{J}{n_{\text{lattice}}} \sum_{i \in L} E(i) \qquad E(i) = \sum_{j \in \mathcal{N}(i)} s_i s_j$$

Ising model

- Consider the 2d Ising model:
  A lattice L with spins $s_i \in \{+1, -1\}$ at each lattice site and energy

$$\mathcal{E} = -\frac{J}{n_{\text{lattice}}} \sum_{i \in L} E(i) \qquad E(i) = \sum_{j \in \mathcal{N}(i)} s_i s_j$$

- Under C_4 (lattice rotations by $90°$), $E(i)$ is equivariant and $\mathcal{E}$ is invariant

Ising model

- Consider the 2d Ising model:
  A lattice L with spins $s_i \in \{+1, -1\}$ at each lattice site and energy

$$\mathcal{E} = -\frac{J}{n_{\text{lattice}}} \sum_{i \in L} E(i) \qquad E(i) = \sum_{j \in \mathcal{N}(i)} s_i s_j$$

- Under C_4 (lattice rotations by 90°), $E(i)$ is equivariant and $\mathcal{E}$ is invariant

- We train a one-hidden-layer MLP on 128 samples, augmented to 512 samples, under the MSE loss on $E(i)$ with full-batch gradient descent

Ising model

- Consider the 2d Ising model:
  A lattice L with spins $s_i \in \{+1, -1\}$ at each lattice site and energy

$$\mathcal{E} = -\frac{J}{n_{\text{lattice}}} \sum_{i \in L} E(i) \qquad E(i) = \sum_{j \in \mathcal{N}(i)} s_i s_j$$

- Under C_4 (lattice rotations by 90°), $E(i)$ is equivariant and $\mathcal{E}$ is invariant
- We train a one-hidden-layer MLP on 128 samples, augmented to 512 samples, under the MSE loss on $E(i)$ with full-batch gradient descent
- For this system, the NTK can be computed exactly (e.g. using the `neural-tangents` package) [Novak et al. 2020]

Ising model

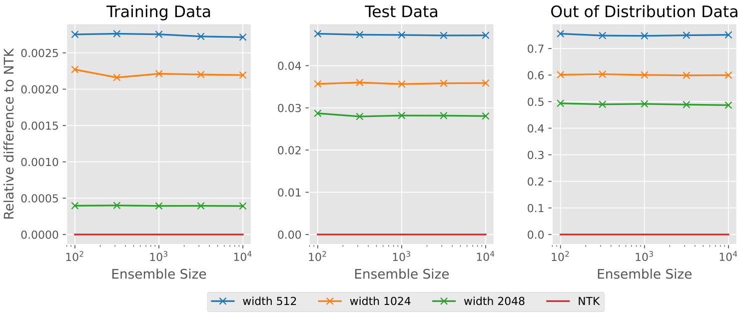
- Consider the 2d Ising model:
  A lattice L with spins $s_i \in \{+1, -1\}$ at each lattice site and energy

$$\mathcal{E} = -\frac{J}{n_{\text{lattice}}} \sum_{i \in L} E(i) \qquad E(i) = \sum_{j \in \mathcal{N}(i)} s_i s_j$$

- Under C_4 (lattice rotations by 90°), $E(i)$ is equivariant and $\mathcal{E}$ is invariant
- We train a one-hidden-layer MLP on 128 samples, augmented to 512 samples, under the MSE loss on $E(i)$ with full-batch gradient descent
- For this system, the NTK can be computed exactly (e.g. using the `neural-tangents` package) [Novak et al. 2020]
- Generate out of distribution data by sampling spins from $\mathcal{N}(0, 1)$

Ising model: convergence to the NTK

- For growing width, the MLP ensemble-predictions converge to the NTK predictions



Ising model: emergent invariance

- Measure ***relative orbit standard deviation***

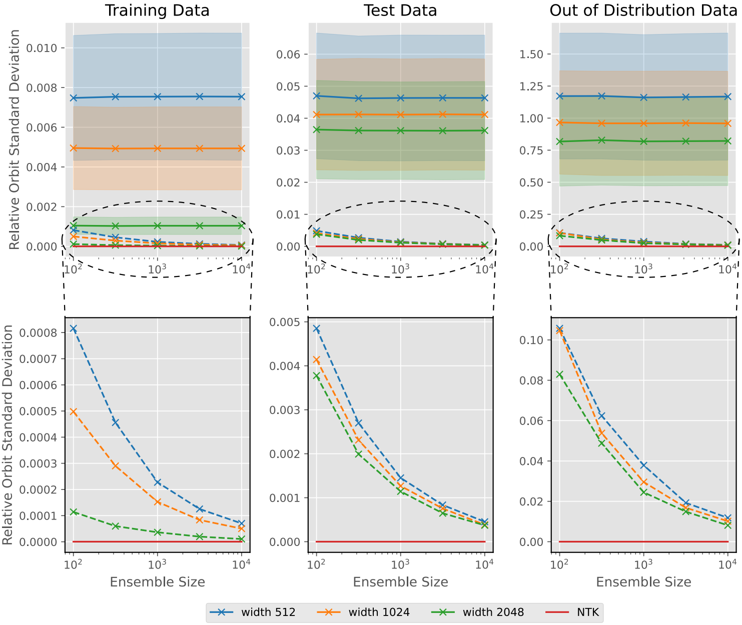
$$\frac{\mathrm{std}_{g \in C_4} \mathcal{E}(\{s_{\rho(g)i}\})}{\mathrm{mean}_{g \in C_4} \mathcal{E}(\{s_{\rho(g)i}\})}$$

Ising model: emergent invariance

- Measure ***relative orbit standard deviation***

$$\frac{\mathrm{std}_{g\in C_4}\mathcal{E}(\{s_{\rho(g)i}\})}{\mathrm{mean}_{g\in C_4}\mathcal{E}(\{s_{\rho(g)i}\})}$$

Ising model: emergent invariance



FashionMNIST: emergent invariance

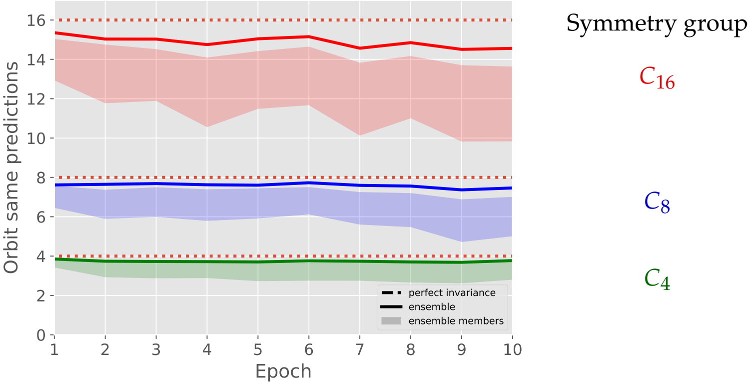
- Train ensembles of CNNs on FashionMNIST augmented by C_k (multiples of $360°/k$) with $k = 4, 8, 16$

FashionMNIST: emergent invariance

- Train ensembles of CNNs on FashionMNIST augmented by C_k (multiples of $360°/k$) with $k = 4, 8, 16$
- Measure invariance using ***orbit same predictions***: number of predictions in the orbit which agree with the prediction on untransformed sample

FashionMNIST: emergent invariance

- Train ensembles of CNNs on FashionMNIST augmented by C_k (multiples of $360°/k$) with $k = 4, 8, 16$
- Measure invariance using ***orbit same predictions***: number of predictions in the orbit which agree with the prediction on untransformed sample
- Throughout training, the ensemble predictions are more invariant than the predictions of the ensemble members, even out of distribution:



FashionMNIST: continuous symmetry

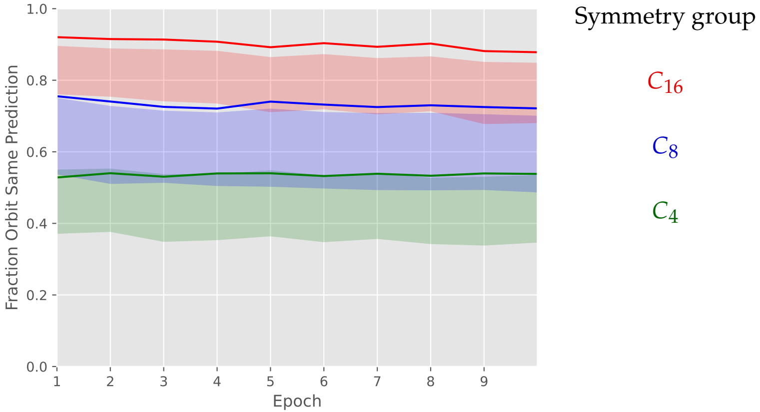
- For continuous groups, the emergent equivariance is only approximate since we cannot augment with the entire group orbit

FashionMNIST: continuous symmetry

- For continuous groups, the emergent equivariance is only approximate since we cannot augment with the entire group orbit
- For augmentation with more samples from the orbit, equivariance should increase

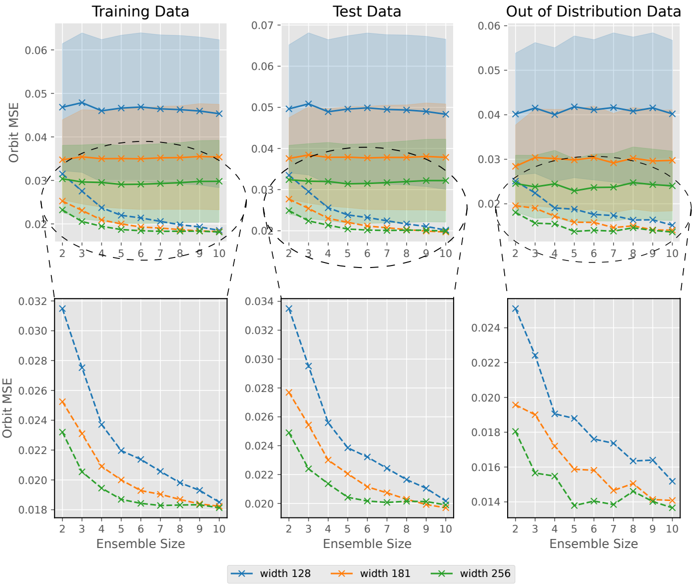
FashionMNIST: continuous symmetry

- For continuous groups, the emergent equivariance is only approximate since we cannot augment with the entire group orbit
- For augmentation with more samples from the orbit, equivariance should increase
- We see this explicitly in the fraction of samples yielding the same prediction when rotations are sampled randomly from SO(2) rotations



Cross products: emergent equivariance

- Train a two-hidden-layer MLP to predict cross-product in $\mathbb{R}^3$



Conclusion

Conclusions

Summary

- Under data augmentation, ensemble predictions become exactly equivariant in the large width limit
- This equivariance holds even out of distribution and at any training time
- We show this by explicitly computing the transformation properties of the neural tangent kernel under data augmentation

Conclusions

Summary

- Under data augmentation, ensemble predictions become exactly equivariant in the large width limit
- This equivariance holds even out of distribution and at any training time
- We show this by explicitly computing the transformation properties of the neural tangent kernel under data augmentation

Application

- If you need an ensemble, consider data augmentation instead of manifestly equivariant models
- If you need data augmentation, consider an ensemble to boost equivariance

Conclusions

Summary

- Under data augmentation, ensemble predictions become exactly equivariant in the large width limit
- This equivariance holds even out of distribution and at any training time
- We show this by explicitly computing the transformation properties of the neural tangent kernel under data augmentation

Application

- If you need an ensemble, consider data augmentation instead of manifestly equivariant models
- If you need data augmentation, consider an ensemble to boost equivariance

Outlook

- Extend proof to general layers
- More detailed investigation of continuous approximation
- Consider finite-width corrections

Emergent Equivariance in Deep Ensembles

arXiv: 2403.03103

Thank you!

Appendix

Standard parametrization

- We usually parametrize MLPs as

$$z^{(\ell)} = W^{\ell} f^{\ell}(x)\,, \quad f^{(\ell)}(x) = \sigma(z^{(\ell-1)}(x))\,, \quad W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_{\ell-1}}\,, \quad W_{ij}^{(\ell)} \sim \mathcal{N}\left(0, \frac{1}{n_{\ell-1}}\right)$$

- For a one-hidden-layer network $\mathbb{R} \longrightarrow \mathbb{R}$

$$f(x) = W^{(2)}\sigma(W^{(1)}x)\,, \qquad W_i^{(1)} \sim \mathcal{N}(0,1)\,, \qquad W_i^{(2)} \sim \mathcal{N}\left(0, \frac{1}{n}\right)$$

- Under gradient descent, the weight updates scale in n as

$$\frac{\partial f}{\partial W_i^{(2)}} = \sigma(W_i^{(1)}x) \in \mathcal{O}(1)\,, \qquad \frac{\partial f}{\partial W_i^{(1)}} = W_i^{(2)}\sigma'(W_i^{(1)}x)x \in \mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$$

- Hence, the updates in the last layer do not decay as $n \longrightarrow \infty$

NTK parametrization

Standard parametrization:

$$z^{(\ell)} = W^\ell f^\ell(x)\,, \qquad W_{ij}^{(\ell)} \sim \mathcal{N}\left(0, \frac{1}{n_{\ell-1}}\right)$$

- In NTK parametrization, we instead use

$$z^{(\ell)} = \frac{1}{\sqrt{n_{\ell-1}}} W^\ell f^\ell(x)\,, \qquad W_{ij}^{(\ell)} \sim \mathcal{N}(0, 1)$$

- This does not change the output distribution at initialization
- However, the gradients now scale as

$$\frac{\partial f}{\partial W_i^{(2)}} = \frac{1}{\sqrt{n}} \sigma(W_i^{(1)} x) \in \mathcal{O}\left(\frac{1}{\sqrt{n}}\right)\,, \qquad \frac{\partial f}{\partial W_i^{(1)}} = \frac{1}{\sqrt{n}} W_i^{(2)} \sigma'(W_i^{(1)} x) x \in \mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$$

- In this parametrization, the gradients vanish as $n \longrightarrow \infty$
- Hence, the dynamics become tractable in this parametrization
- We will assume NTK parametrization in the rest of the talk

Neural network Gaussian process

When taking the layer width of an MLP to infinity sequentially, the preactivations $z_i^\ell(x)$ at initialization become a Gaussian process with mean zero and covariance function $K^{(\ell)}(x, x')\delta_{ij}$ recursively defined by

$$K^{(1)}(x, x') = \frac{1}{n_0} x^\top x'$$

$$\Lambda^{(\ell)}(x, x') = \begin{pmatrix} K^{(\ell)}(x, x) & K^{(\ell)}(x, x') \\ K^{(\ell)}(x', x) & K^{(\ell)}(x', x') \end{pmatrix}$$

$$K^{(\ell)}(x, x') = \mathbb{E}_{(u,v)\sim\mathcal{N}(0,\Lambda^{(\ell-1)}(x,x'))}[\sigma(u)\sigma(v)]$$

$K^{(\ell)}(x, x')$ is also known as the ***NNGP kernel***

NTK at initialization

NTK recursion [Jacot et al. 2020]

When taking the layer widths to infinity sequentially, the empirical NTK $\Theta^{\theta}_{ij}(x, x')$ at initialization converges in probability to a deterministic kernel $\Theta(x, x')\delta_{ij}$ recursively defined by

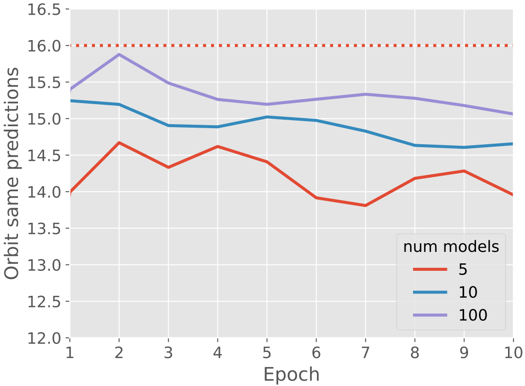
$$\begin{aligned}
\Theta^{(1)}(x, x') &= K^{(1)}(x, x') \\
\Theta^{(\ell+1)}(x, x') &= \Theta^{(\ell)}(x, x')\dot{K}^{(\ell+1)}(x, x') + K^{(\ell+1)}(x, x') \\
\Theta(x, x') &= \Theta^{(L)}(x, x')
\end{aligned}$$

where

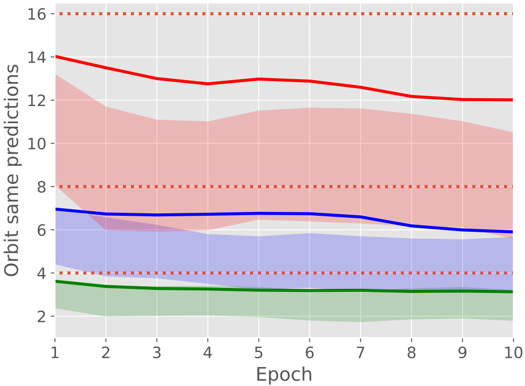
$$\dot{K}^{(\ell)}(x, x') = \mathbb{E}_{(u,v)\sim\mathcal{N}(0,\Lambda^{(\ell-1)}(x,x'))}[\sigma'(u)\sigma'(v)]$$

- For most common architectures, this recursion can be computed explicitly, e.g. using `neural tangents` Python package [Novak et al. 2020]

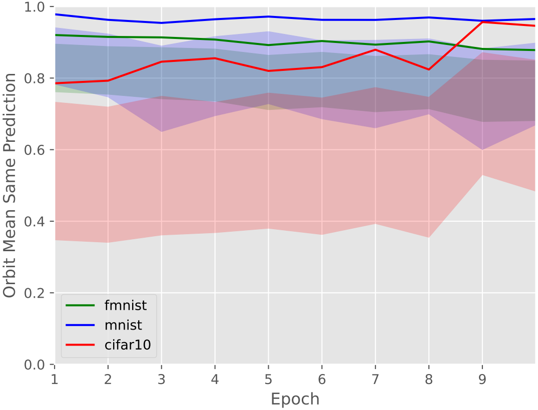
FashionMNIST: ensemble sizes



FashionMNIST: OSP on CIFAR10



FashionMNIST: continuous symmetry OOD



FashionMNIST: OOD equivariance comparison

	DeepEns	E2CNN	Canon
C_4	3.85 ± 0.12	$\mathbf{4 \pm 0.0}$	$\mathbf{4 \pm 0.0}$
C_8	$\mathbf{7.72 \pm 0.34}$	$\mathbf{7.71 \pm 0.21}$	$\mathbf{7.45 \pm 0.14}$
C_{16}	$\mathbf{15.24 \pm 0.69}$	$\mathbf{15.08 \pm 0.34}$	12.41 ± 0.85

- Comparison to equivariant model [Weiler, Cesa 2019] and canonicalization [Kaba et al. 2022]
- Augmented ensembles are competitive with manifestly equivariant methods
- For symmetries larger than C_4, manifestly equivariant models suffer from interpolation effects