Bayesian Neural Model Selection for Symmetry Learning

Tycho van der Ouderaa

@ JSM 2024 in Portland, US

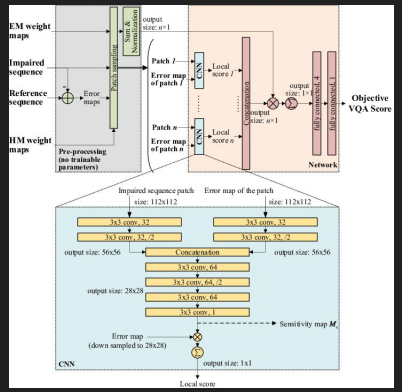
I. What is Bayesian model selection?

II. How to scale to deep neural networks

III. Learning inductive bias and symmetries

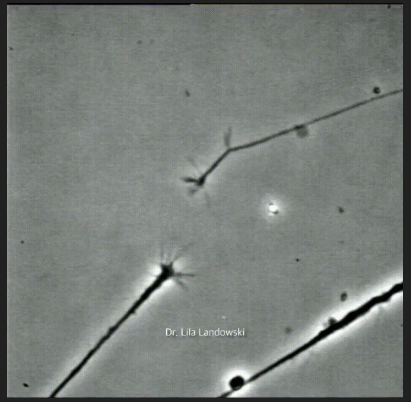
Future vision of ML

Architectures determined by
trial-and-error



Bayesian **model selection**

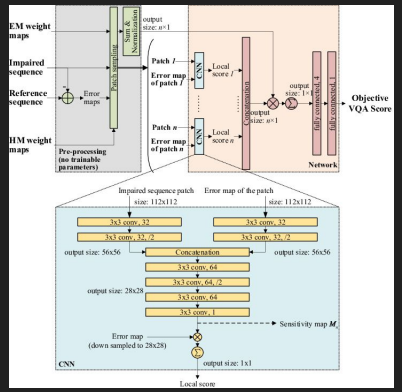
Morphable architectures
automatic inductive bias



References:
Left: *Bridge the Gap Between VQA and Human Behavior on Omnidirectional Video: A Large-Scale Dataset and a Deep Learning Model*, Chen Li, Mai Xu, Xinzhe Du, Zulin Wang
Right video:"Neurons connecting to one another in a Petri dish - growth cones." by *Dr Lila Landowski*

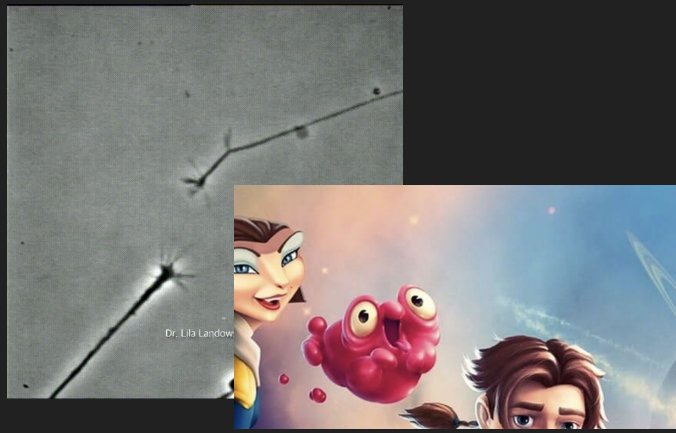
Future vision of ML

Architectures determined by
trial-and-error



Bayesian **model selection**

Morphable architectures
automatic inductive bias



References:
Left: *Bridge the Gap Between VQA and Human Behavior on Omnidirectional Video: A Large-Scale Dataset and a Deep Learning Model*, Chen Li, Mai Xu, Xinzhe Du, Zulin Wang
Right video:"Neurons connecting to one another in a Petri dish - growth cones." by *Dr Lila Landowski*
Right picture: Wells, R. (Producer), Clements, R., & Musker, J. (Directors). (2002). Treasure Planet [Film]. Walt Disney Pictures.

Fitting a neural network

$$\mathrm{NN}_{\boldsymbol{\theta}} : \mathbb{R}^M \to \mathbb{R}^N$$

Regular training (maximum likelihood / MAP)

Likelihood: $p(\mathcal{D} \mid \boldsymbol{\theta})$

Prior: $p(\boldsymbol{\theta})$

Regular training (maximum likelihood / MAP)

Likelihood: $p(\mathcal{D} \mid \boldsymbol{\theta})$

Prior: $p(\boldsymbol{\theta})$

Posterior: $p(\boldsymbol{\theta} \mid \mathcal{D}) = \dfrac{p(\mathcal{D} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}$

Regular training (maximum likelihood / MAP)

Likelihood: $p(\mathcal{D} \mid \boldsymbol{\theta})$

Prior: $p(\boldsymbol{\theta})$

Posterior: $p(\boldsymbol{\theta} \mid \mathcal{D}) = \dfrac{p(\mathcal{D} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})}$

Optimal $\boldsymbol{\theta}_* = \operatorname{argmax}_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathcal{D})$

$$= \operatorname{argmax}_{\boldsymbol{\theta}} \log \frac{p(\mathcal{D} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{\cancel{p(\mathcal{D})}} = \log p(\mathcal{D} \mid \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$$

$$= \operatorname{argmin}_{\boldsymbol{\theta}} \underbrace{-\log p(\mathcal{D} \mid \boldsymbol{\theta})}_{\text{negative log likelihood}} \underbrace{-\log p(\boldsymbol{\theta})}_{\text{regularizer}}$$

Regular loss

Supervised training (maximum likelihood / MAP)

Likelihood: $p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) = \prod_{n=1}^{N} \mathcal{N}(y_n \mid \mathrm{NN}_{\boldsymbol{\theta}}(x_n), \sigma^2 \mathbf{I})$

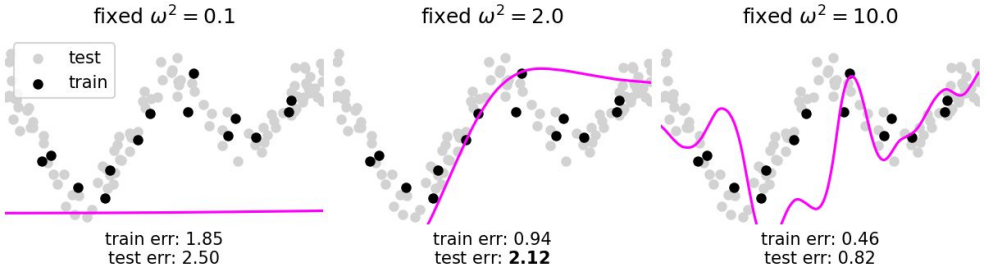
Prior: $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta} \mid \mathbf{0}, \sigma^2_{\text{prior}} \mathbf{I})$

Posterior: $p(\boldsymbol{\theta} \mid \boldsymbol{x}, \boldsymbol{y})$

Optimal $\boldsymbol{\theta}_* = \mathrm{argmin}_{\boldsymbol{\theta}} \underbrace{\frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - \mathrm{NN}_{\boldsymbol{\theta}}(x_n))^2}_{\text{mean squared error}} + \underbrace{\frac{1}{2\sigma^2_{\text{prior}}} ||\boldsymbol{\theta}||_2^2}_{\text{weight-decay}}$

Bayesian model selection

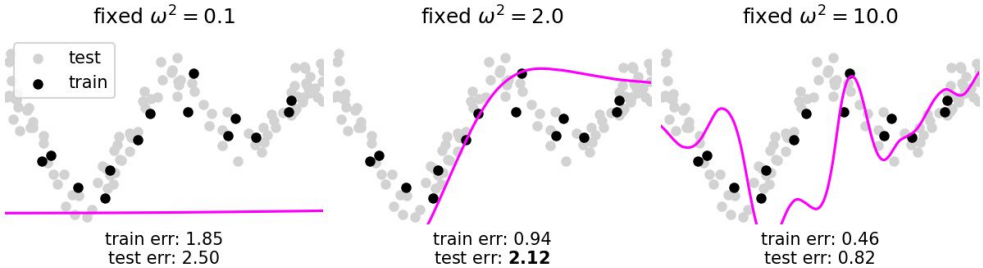
Regular training (maximum likelihood)



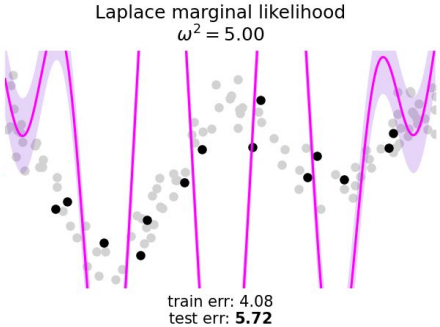
$$\mathrm{NN}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathbf{W}_3 \mathrm{relu}(\mathbf{W}_2 \cos(\omega^2 \mathbf{W}_1 \boldsymbol{x}))$$

Bayesian model selection

Regular training (maximum likelihood)



Bayesian model selection (marginal likelihood)



Uses gradients.
No expensive cross-validation

$$\mathrm{NN}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathbf{W}_3 \mathrm{relu}(\mathbf{W}_2 \cos(\omega^2 \mathbf{W}_1 \boldsymbol{x}))$$

Learn how to generalize

Bayesian model selection (marginal likelihood)

$$p(\boldsymbol{\theta}, \boldsymbol{\eta} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{\eta}) p(\boldsymbol{\eta})}{p(\mathcal{D} \mid \boldsymbol{\eta})} = \underbrace{\frac{p(\mathcal{D} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{\eta})}{p(\mathcal{D} \mid \boldsymbol{\eta})}}_{\text{usual posterior}} \underbrace{\frac{p(\mathcal{D} \mid \boldsymbol{\eta}) p(\boldsymbol{\eta})}{p(\mathcal{D})}}_{\text{hyper posterior}}$$

Bayesian model selection (marginal likelihood)

$$p(\boldsymbol{\theta}, \boldsymbol{\eta} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{\eta}) p(\boldsymbol{\eta})}{p(\mathcal{D} \mid \boldsymbol{\eta})} = \underbrace{\frac{p(\mathcal{D} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{\eta})}{p(\mathcal{D} \mid \boldsymbol{\eta})}}_{\text{usual posterior}} \underbrace{\frac{p(\mathcal{D} \mid \boldsymbol{\eta}) p(\boldsymbol{\eta})}{p(\mathcal{D})}}_{\text{hyper posterior}}$$

Marginal likelihood $p(\mathcal{D} \mid \boldsymbol{\eta}) = \int_{\boldsymbol{\theta}} p(\mathcal{D} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{\eta}) \mathrm{d}\boldsymbol{\theta}$

Bayesian model selection (marginal likelihood)

$$p(\boldsymbol{\theta}, \boldsymbol{\eta} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{\eta}) p(\boldsymbol{\eta})}{p(\mathcal{D} \mid \boldsymbol{\eta})} = \underbrace{\frac{p(\mathcal{D} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{\eta})}{p(\mathcal{D} \mid \boldsymbol{\eta})}}_{\text{usual posterior}} \underbrace{\frac{p(\mathcal{D} \mid \boldsymbol{\eta}) p(\boldsymbol{\eta})}{p(\mathcal{D})}}_{\text{hyper posterior}}$$

Marginal likelihood $p(\mathcal{D} \mid \boldsymbol{\eta}) = \int_{\boldsymbol{\theta}} p(\mathcal{D} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{\eta}) \mathrm{d}\boldsymbol{\theta}$

Optimise $\boldsymbol{\eta}_* = \operatorname{argmax}_{\boldsymbol{\eta}} \log p(\mathcal{D} \mid \boldsymbol{\eta})$ [Type-II ML / Empirical Bayes]

Allows optimization of architecture/inductive bias with gradients!

How to integrate out neural network parameters?

Gaussian approximation

Approximate posterior with a Gaussian

$$p(\boldsymbol{\theta} \mid \mathcal{D}, \boldsymbol{\eta}) \approx \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

How to find $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$?

Approximate Bayesian inference

Laplace approximation

$$q(\boldsymbol{\theta}) \approx \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\theta}_{\mathrm{MAP}}, \mathbf{H}^{-1})$$

Variational Inference

$$q(\boldsymbol{\theta}) \approx \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Approximate Bayesian inference

Laplace approximation

$$q(\boldsymbol{\theta}) \approx \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\theta}_{\mathrm{MAP}}, \mathbf{H}^{-1})$$

(local)

Pretrained network + Hessian

Variational Inference

$$q(\boldsymbol{\theta}) \approx \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

(global)

Optimize ELBO objective

Only correlate rows and columns

Independence between layers

$$\mathbb{E}[\mathbf{H}] = \bigoplus_{l=1}^{L} \mathbf{H}_l$$

Independence between inputs and outputs

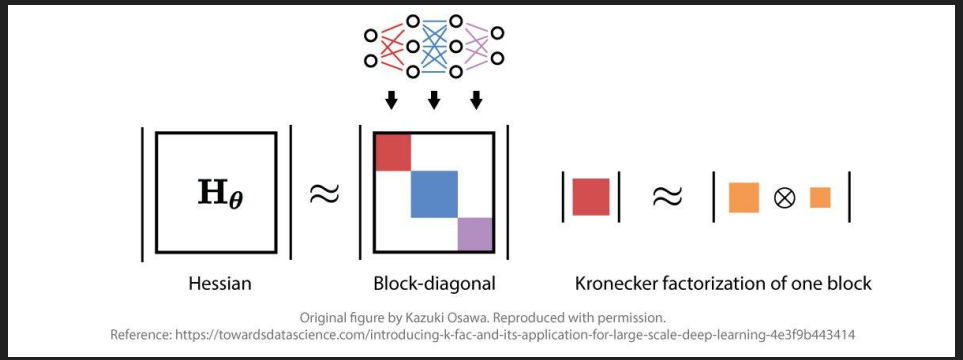
$$\mathbb{E}[\mathbf{H}] = \mathbb{E}[\mathbf{g}_l \mathbf{g}_l^T \otimes \mathbf{x}_l \mathbf{x}_l^T] \approx \mathbb{E}[\mathbf{g}_l \mathbf{g}_l^T] \otimes \mathbb{E}[\mathbf{x}_l \mathbf{x}_l^T]$$

$$\mathbb{E}[\mathbf{g}_l \mathbf{g}_l^T] \approx \frac{1}{N} \sum_{n=1}^{N} \mathbf{g}_l^{(n)} (\mathbf{g}_l^{(n)})^T = \mathbf{S}_l$$

$$\mathbb{E}[\mathbf{x}_l \mathbf{x}_l^T] \approx \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_l^{(n)} (\mathbf{x}_l^{(n)})^T = \mathbf{A}_l$$

KFAC Hessian approximation:

$$\mathbb{E}[\mathbf{H}] = \bigoplus_{l=1}^{L} (\mathbf{S}_l \otimes \mathbf{A}_l)$$



Approximate Bayesian inference

Laplace approximation

$$q(\boldsymbol{\theta}) \approx \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\theta}_{\mathrm{MAP}}, \mathbf{H}^{-1})$$

$$\mathbf{H} = \bigoplus_{l=1}^{L} (\mathbf{S}_l \otimes \mathbf{A}_l)$$ [1, 2]

Variational Inference

$$q(\boldsymbol{\theta}) \approx \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$\boldsymbol{\Sigma}^{-1} = \bigoplus_{l=1}^{L} (\mathbf{S}_l \otimes \mathbf{A}_l)$$ [3]

[1] **Optimizing Neural Networks with Kronecker-factored Approximate Curvature**
James Martens and Roger Grosse
In ICML 2016
[2] **Scalable Marginal Likelihood Estimation for Model Selection in Deep Learning**
Alexander Immer,Matthias Bauer, Vincent Fortuin, Gunnar Ratsch, Mohammad Emtiyaz Khan
In ICML 2021
[3] **Structured and efficient variational deep learning with matrix gaussian posteriors**
Christos Louizos and Max Welling
In ICML 2016

Approximate Bayesian inference

Laplace approximation

$$q(\boldsymbol{\theta}) \approx \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\theta}_{\mathrm{MAP}}, \mathbf{H}^{-1})$$

$$\mathbf{H} = \bigoplus_{l=1}^{L} (\mathbf{S}_l \otimes \mathbf{A}_l)$$

Variational Inference

$$q(\boldsymbol{\theta}) \approx \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$\boldsymbol{\Sigma}^{-1} = \bigoplus_{l=1}^{L} (\mathbf{S}_l \otimes \mathbf{A}_l)$$

[4] **Stochastic Marginal Likelihood Gradients using Neural Tangent Kernels**
Alexander Immer, Tycho F.A. van der Ouderaa, Mark van der Wilk, Gunnar Ratsch, Bernhard Schölkopf
In ICML 2023

[5] **Variational Inference Failures Under Model Symmetries: Permutation Invariant Posteriors for Bayesian Neural Networks**
Yoav Gelberg, Tycho F. A. van der Ouderaa, Mark van der Wilk, Yarin Gal
In ICML 2024, GRaM workshop (best paper)

Tricks of the trade:

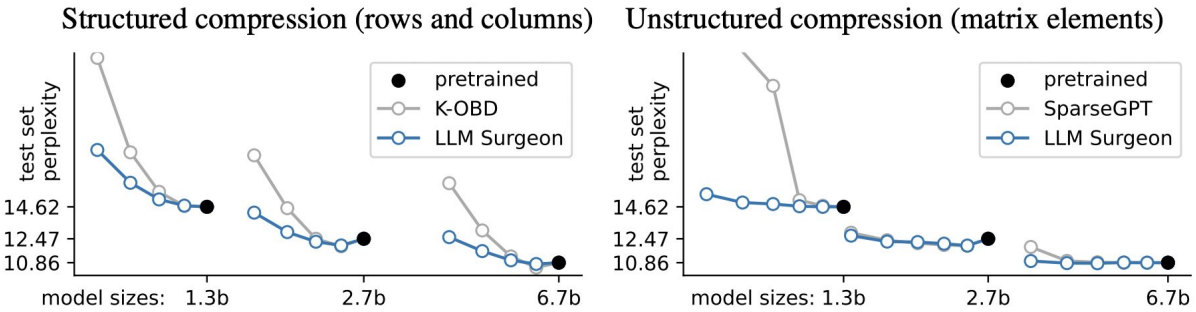
- Easy inverse / determinant (Kronecker identities in Laplace [1,2], Cholesky parameterization in VI as App. D in [13])
- Use mini-batches (easy in VI, requires care in Laplace [4])
- Account for weight-symmetries (with correction in KL term [5])
- Optimize prior variance and output variance empirically ([2], closed-form VI updates in App. D of [13])

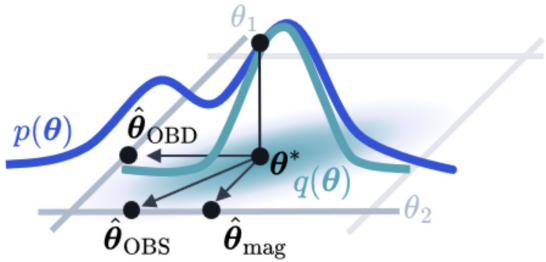
Laplace approximations at scale

Laplace of networks with billions of parameters

Laplace can be applied to modern architectures [6]

State-of-the-art in structured LLM pruning [7]:



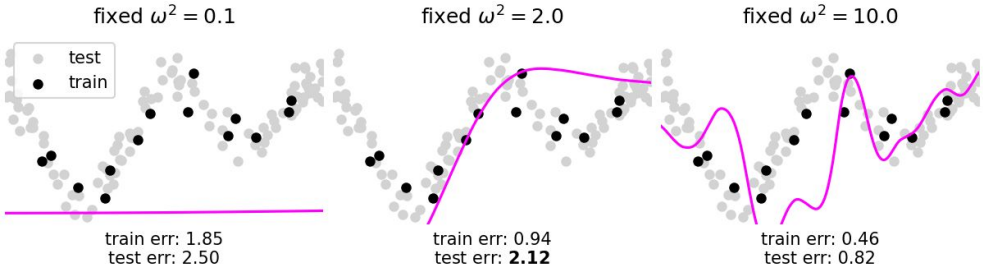


[6] **Kronecker-Factored Approximate Curvature for Modern Neural Network Architectures**
Runa Eschenhagen, Alexander Immer, Richard E Turner, Frank Schneider, Philipp Hennig
In NeurIPS 2023

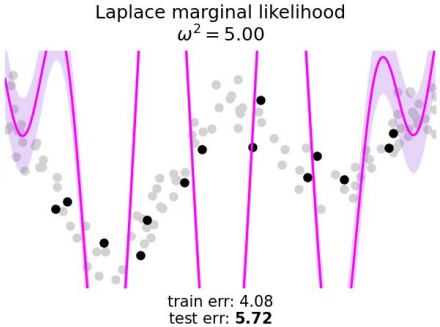
[7] **The LLM Surgeon**
Tycho F.A. van der Ouderaa, Markus Nagel, Mart van Baalen, Yuki M. Asano, Tijmen Blankevoort
In ICLR 2024

Bayesian model selection

Regular training (maximum likelihood)



Bayesian model selection (marginal likelihood)

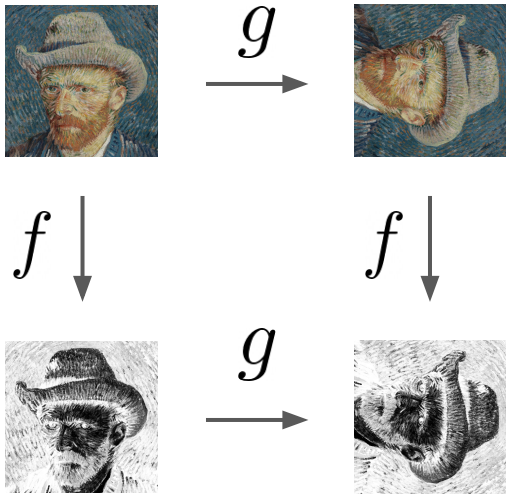


$$\mathrm{NN}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathbf{W}_3 \mathrm{relu}(\mathbf{W}_2 \cos(\omega^2 \mathbf{W}_1 \boldsymbol{x}))$$

Inductive bias in deep learning

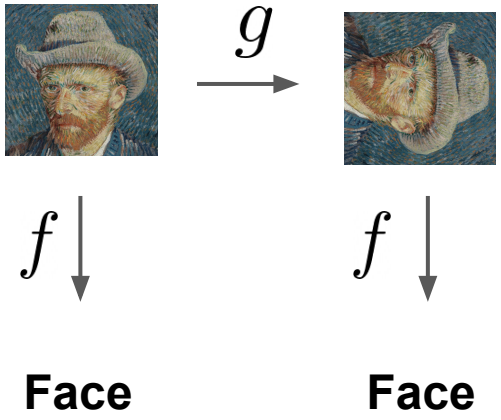
Equivariance

$$f(g \cdot x) = g \cdot f(x)$$



Invariance

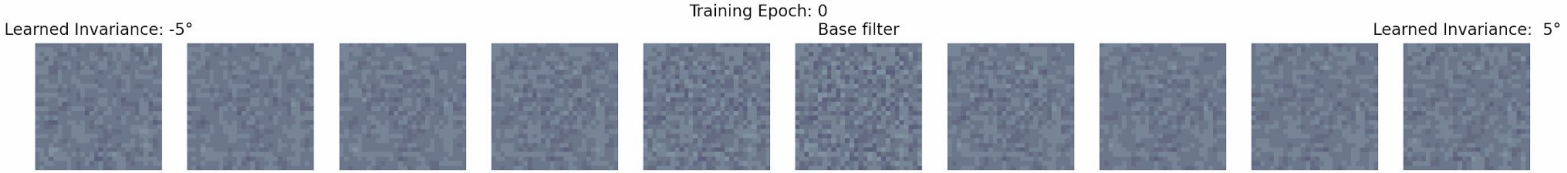
$$f(g \cdot x) = f(x)$$



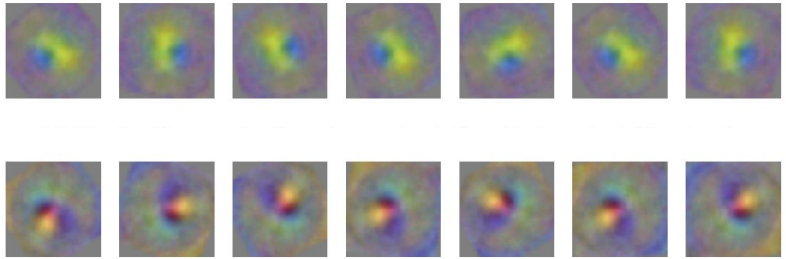
Example image: *Vincent van Gogh, Self-Portrait with Grey Felt Hat, 1887*
Van Gogh Museum Amsterdam

Can we learn symmetries?

Learned filter bank on rotated MNIST:



Learned filter bank on rotated CIFAR-10:



Learning Invariant Weights in Neural Networks

Tycho F.A. van der Ouderaa[1] Mark van der Wilk[1]

[1]Imperial College London, UK

arXiv:2202.12439v1 [stat.ML] 25 Feb 2022

Abstract

Assumptions about invariances or symmetries in data can significantly increase the predictive power of statistical models. Many commonly used models in machine learning are constraint to respect certain symmetries in the data, such as translation equivariance in convolutional neural networks, and incorporation of new symmetry types is actively being studied. Yet, efforts to learn such invariances from the data itself remains an open research problem. It has been shown that marginal likelihood offers a principled way to learn invariances in Gaussian Processes. We propose a weight-space equivalent to this approach, by minimizing a lower bound on the marginal likelihood to learn invariances in neural networks resulting in naturally higher performing models.

1 INTRODUCTION

Intuitively, invariances allow models to extrapolate (or "generalize") beyond training data (see Figure 1 for an extreme example). An invariant model does not change its output when the input is changed by transformations to which it is deemed invariant. The most straightforward way to achieve this is perhaps by enlarging the dataset with transformed examples: a process known as data augmentation. A link between invariance and data augmentation in kernel space was made by Dao et al. [2019]. We show that this invariance can equivalently be described as transformations on the weights, similar to Cohen and Welling [2016] where a neural network is constrained to respect rotational symmetry through rotated weight copies. We then continue and adopt the principled method of marginal likelihood that has proven effective in learning invariance in Gaussian Processes (GPs) [van der Wilk et al., 2018], in order to learn the correct invariance from data. We do what is common in Bayesian model selection and find the invariance by optimizing the marginal likelihood. As the true marginal likelihood is not tractable for commonly used neural networks, we propose a lower bound of the marginal likelihood to allow this in practice. By learning distributions on affine groups, we can select the correct invariance for a particular task, without having to perform cross-validation or even requiring a separate validation set. We succesfully learn the correct invariance on different MNIST and CIFAR-10 tasks leading to better performing models.

2 RELATED WORK

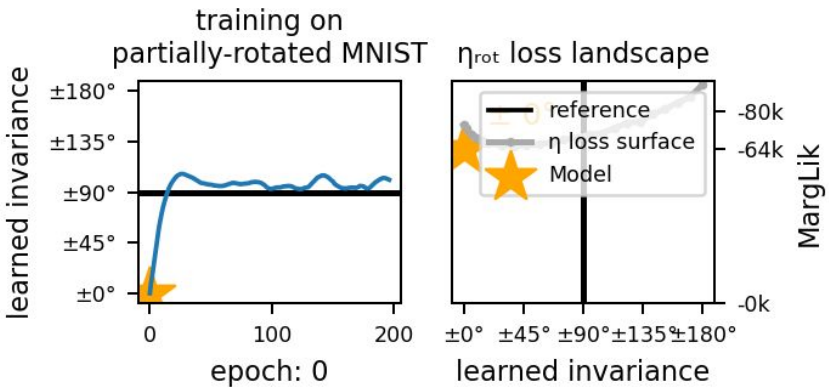
Convolutional neural networks (CNNs) have been successful in a wide range of problems and played a key role in the success of Deep Learning [LeCun et al., 2015]. It is commonly understood that the translational symmetries that arise from effective weight-sharing in CNNs is an important driver for its outstanding performance on many tasks.

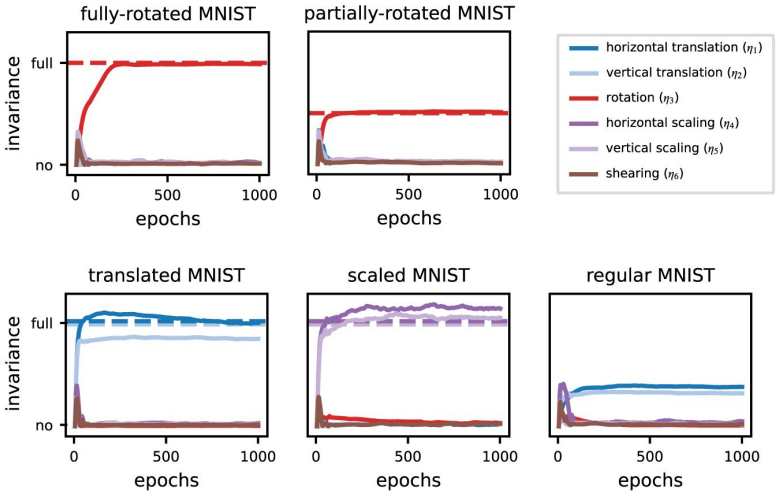
In Cohen and Welling [2016], a group-theoretical framework was proposed that extends CNNs beyond translational symmetries, and demonstrated this for discrete group actions. Many studies since have proposed ways to incorporate other symmetries, such as continuous rotation, scale and translation, into the weights of neural networks [Worrall et al., 2017, Weiler et al., 2018, Marcos et al., 2017, Estrach et al., 2017, Weiler and Cesa, 2019, Bekkers, 2019] and recent efforts allow practical equivariance in neural networks for arbitrary symmetry groups [Finzi et al., 2021]. Nevertheless, symmetries are typically fixed and must be known and explicitly specified in advance.

Some studies have proposed to learn invariances through data augmentations [Cubuk et al., 2018, Lorraine et al., 2020], but thus do not embed symmetries in the weights and rely on an additional validation loss. Zhou et al. [2020] does learn parameter sharing of layer weights but requires a meta-learning procedure that also relies on a validation loss. Benton et al. [2020] circumvents the need for valida-

[8] **Learning Invariant Weights in Neural Networks**
Tycho F.A. van der Ouderaa and Mark van der Wilk
In ICML 2021 (UDL Workshop) and UAI 2022 (Oral)

Invariance Learning in Deep Nets



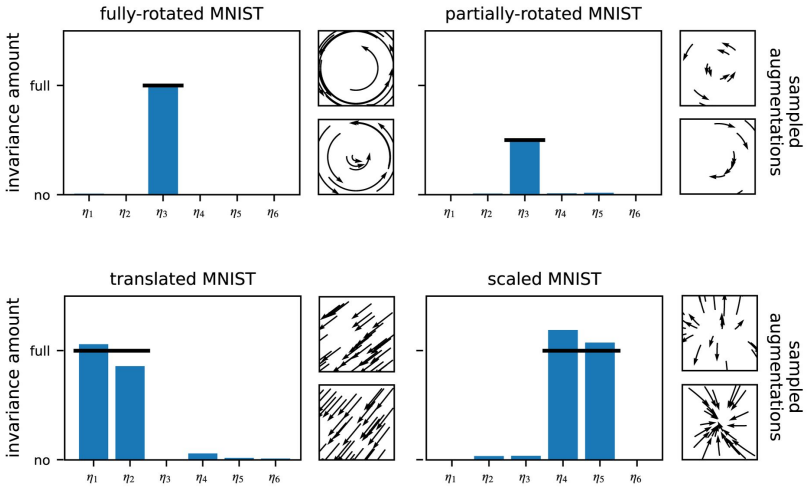


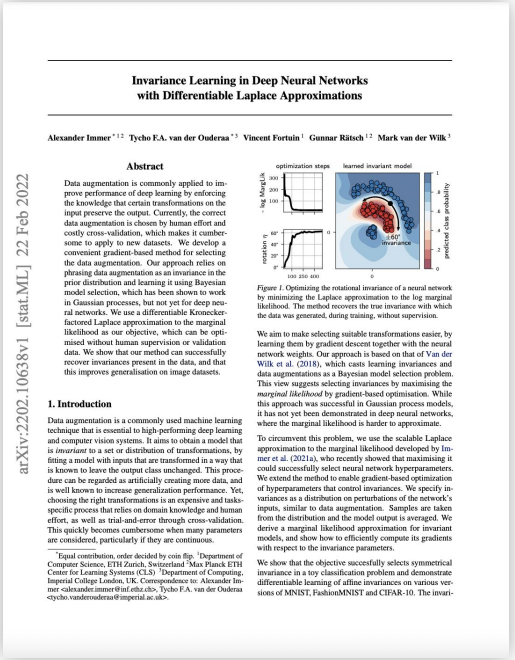
$$\log p(\mathcal{D}|\mathcal{H}) \approx \log p(\mathcal{D}, \boldsymbol{\theta}_*|\mathcal{H}) - \tfrac{1}{2} \log \left| \tfrac{1}{2\pi} \bar{\mathbf{H}}_{\boldsymbol{\theta}_*} \right|$$

Marginal likelihood MAP fit Hessian logdet

[9] **Invariance Learning in Deep Neural Networks with Differentiable Laplace Approximations**
Alexander Immer*, Tycho F.A. van der Ouderaa*, Vincent Fortuin, Gunnar Rätsch and Mark van der Wilk
In NeurIPS 2022

Invariance Learning in Deep Nets





arXiv:2202.10638v1 [stat.ML] 22 Feb 2022

Invariance Learning in Deep Neural Networks with Differentiable Laplace Approximations

Alexander Immer [*1 2] Tycho F.A. van der Ouderaa [*3] Vincent Fortuin [1] Gunnar Rätsch [1 2] Mark van der Wilk [3]

Abstract

Data augmentation is commonly applied to improve performance of deep learning by enforcing the knowledge that certain transformations on the input preserve the output. Currently, the correct data augmentation is chosen by human effort and costly cross-validation, which makes it cumbersome to apply to new datasets. We develop a convenient gradient-based method for selecting the data augmentation. Our approach relies on phrasing data augmentation as an invariance in the prior distribution and learning it using Bayesian model selection, which has been shown to work in Gaussian processes, but not yet for deep neural networks. We use a differentiable Kronecker-factored Laplace approximation to the marginal likelihood as our objective, which can be optimised without human supervision or validation data. We show that our method can successfully recover invariances present in the data, and that this improves generalisation on image datasets.

1. Introduction

Data augmentation is a commonly used machine learning technique that is essential to high-performing deep learning and computer vision systems. It aims to obtain a model that is *invariant* to a set or distribution of transformations, by fitting a model with inputs that are transformed in a way that is known to leave the output class unchanged. This procedure can be regarded as artificially creating more data, and is well known to increase generalization performance. Yet, choosing the right transformations is an expensive and tasks-specific process that relies on domain knowledge and human effort, as well as trial-and-error through cross-validation. This quickly becomes cumbersome when many parameters are considered, particularly if they are continuous.

[*]Equal contribution, order decided by coin flip. [1]Department of Computer Science, ETH Zurich, Switzerland [2]Max Planck ETH Center for Learning Systems (CLS) [3]Department of Computing, Imperial College London, UK. Correspondence to: Alexander Immer <alexander.immer@inf.ethz.ch>, Tycho F.A. van der Ouderaa <tycho.vanderouderaa@imperial.ac.uk>.

optimization steps
learned invariant model
$-\log p(\mathcal{D})$
rotation in °
predicted class probability
±60° invariance

Figure 1. Optimizing the rotational invariance of a neural network by minimizing the Laplace approximation to the log marginal likelihood. The method recovers the true invariance with which the data was generated, during training, without supervision.

We aim to make selecting suitable transformations easier, by learning them by gradient descent together with the neural network weights. Our approach is based on that of Van der Wilk et al. (2018), which casts learning invariances and data augmentations as a Bayesian model selection problem. This view suggests selecting invariances by maximising the *marginal likelihood* by gradient-based optimisation. While this approach was successful in Gaussian process models, it has not yet been demonstrated in deep neural networks, where the marginal likelihood is harder to approximate.

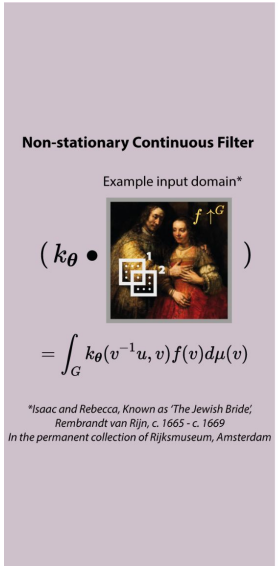
To circumvent this problem, we use the scalable Laplace approximation to the marginal likelihood developed by Immer et al. (2021a), who recently showed that maximising it could successfully select neural network hyperparameters. We extend the method to enable gradient-based optimization of hyperparameters that control invariances. We specify invariances as a distribution on perturbations of the network's inputs, similar to data augmentation. Samples are taken from the distribution and the model output is averaged. We derive a marginal likelihood approximation for invariant models, and show how to efficiently compute its gradients with respect to the invariance parameters.

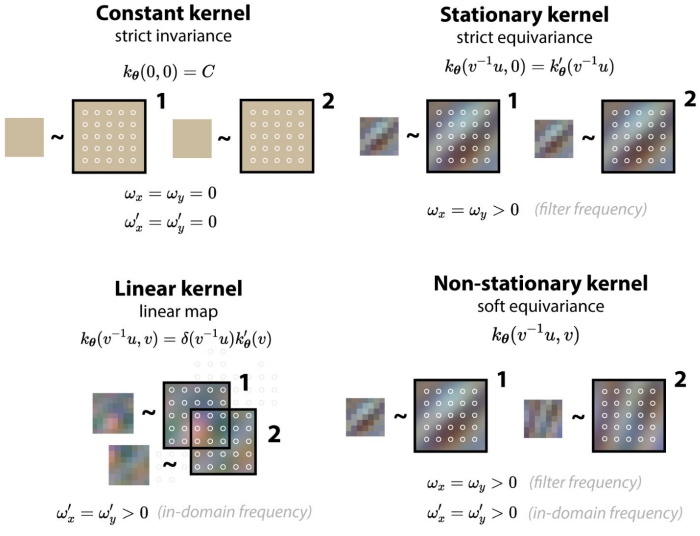
We show that the objective succesfully selects symmetrical invariance in a toy classification problem and demonstrate differentiable learning of affine invariances on various versions of MNIST, FashionMNIST and CIFAR-10. The invari-

[9] **Invariance Learning in Deep Neural Networks with Differentiable Laplace Approximations**
Alexander Immer*, Tycho F.A. van der Ouderaa*, Vincent Fortuin, Gunnar Rätsch and Mark van der Wilk
In NeurIPS 2022

What about learning equivariances?

Relaxing equivariance constraints





Relaxing Equivariance Constraints with Non-stationary Continuous Filters

Tycho F.A. van der Ouderaa
Imperial College London
United Kingdom

David W. Romero
Vrije Universiteit Amsterdam
The Netherlands

Mark van der Wilk
Imperial College London
United Kingdom

Abstract

Equivariances can provide useful inductive biases in neural network modeling, with translation equivariance of convolutional neural networks as canonical example. Equivariances can be embedded in architectures through weight-sharing and constrain the type of functions that a neural network can represent. The type of symmetry is typically fixed and has to be chosen in advance. Although some tasks are inherently equivariant, many tasks do not strictly obey such symmetries in which case equivariance constraints can be overly restrictive. In this work, we propose a parameter-efficient relaxation of equivariance that has a non-equivariant linear map, strict-equivariance and strict-invariance as special case, between which can interpolated in a parameter-efficient manner. The proposed parameterization can be thought of as a building block to allow adjustable symmetry structure in neural networks. Compared to non-equivariant or strict-equivariant baselines, we experimentally verify that soft equivariance can lead to improved performance in terms of test accuracy on CIFAR-10 and CIFAR-100 image classification tasks.

1 Introduction

Symmetries, such as equivariances and invariances, can be embedded into neural network architectures and provide an inductive bias that leads to better generalization, data-efficiency and higher overall performance. Convolutional layers are known to provide translational equivariance in simple euclidean spaces, and many recent works have extended the convolution to more complex groups and domains. When chosen correctly, symmetries can lead to large gains in model performance. Symmetries are typically fixed, must be specified in advance, and can not be adjusted.

Symmetries that are embedded in the network architecture enforce a hard constraint on the functions the network can represent. This is useful when a problem is inherently symmetric, but can become problematic if the task does not fully obey a symmetry. For example, convolutional layers can not encode absolute positional information due to translation equivariance, and '6's and a '9's become indistinguishable under strict rotational invariance. Relaxed symmetry constraints can mitigate such potential symmetry misspecification without losing the inductive bias that symmetries provide.

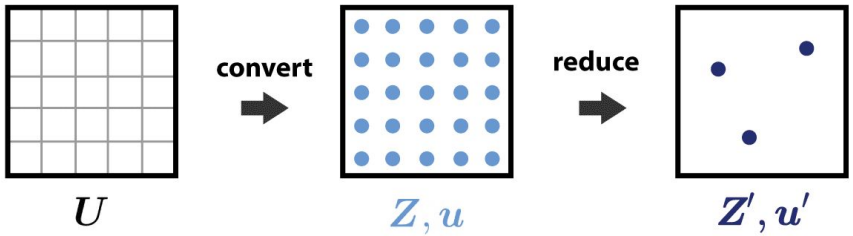
There have been attempts to automatically learn symmetry structures in neural networks from data. Existing approaches commonly rely on parameterizations that relax symmetry constraints, but often focus on invariances which are easier to parameterize than equivariances. Allowing parameterizations that are smooth in their equivariance constraints may provide a useful building block to extend such symmetry discovery methods to enable layer-by-layer learning of equivariant symmetry structures.

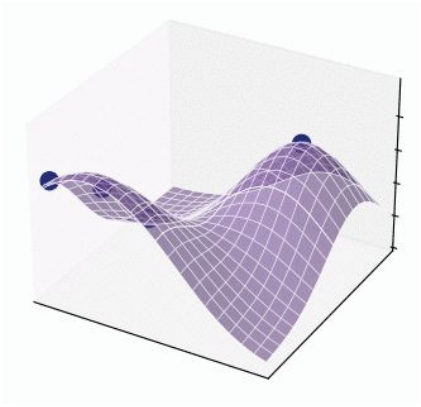
In this work, we propose to relax equivariance constraints by generalizing the convolution operator with non-stationary filters that also depend on the absolute input or output group element. Applied to deep learning, this results in a novel parameter-efficient layer able to interpolate between (*i*) non-invariant dot-products akin to a fully-connected layer, (*ii*) strict group equivariant convolutions and (*iii*) strict group invariant constraints (Fig. 1).

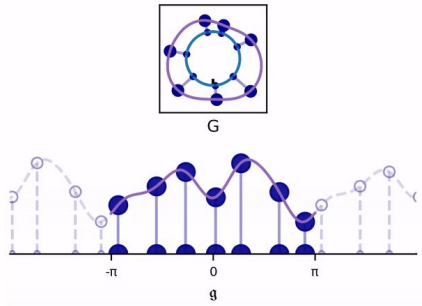
Preprint. Under review.

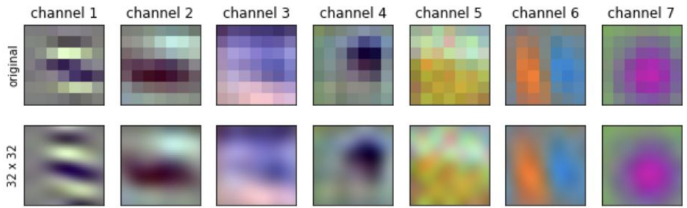
[10] **Relaxing Equivariance Constraints with Non-stationary Continuous Filters**
Tycho F.A. van der Ouderaa, David Romero and Mark van der Wilk
In NeurIPS 2022

Continuous kernels









[11] **Sparse Convolutions on Lie Groups**
Tycho F.A. van der Ouderaa and Mark van der Wilk
In NeurIPS 2022 (Workshop on Symmetry and Geometry in Neural Representations)

Learning layer-wise equivariances

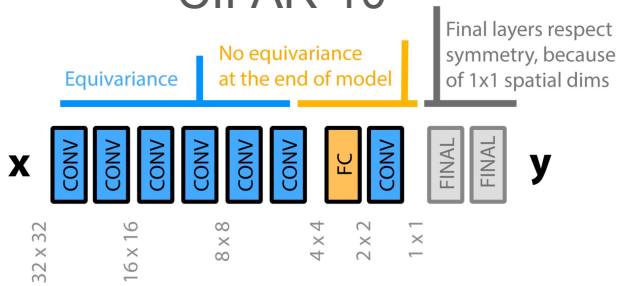
MNIST

Example:

(3, upper left)

Symmetry	Prediction task	MAP FC 112.7 M	MAP CONV 0.4 M	Diff. Laplace FC 112.7 M	Diff. Laplace CONV 0.4 M	Diff. Laplace Learned (**ours**) 1.8 M
Strict symmetry	*(digit)*	95.73	99.38	97.04	99.23	98.86
Partial symmetry	*(digit, quadrant)*	95.10	24.68	95.46	24.50	99.00

CIFAR-10

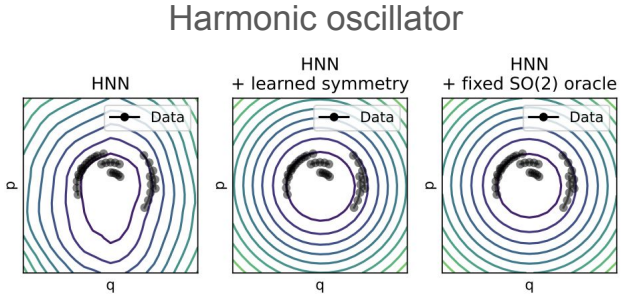


[12] **Learning Layer-wise Equivariances Automatically using Gradients**
Tycho F.A. van der Ouderaa, Alexander Immer, Mark van der Wilk
In NeurIPS 2023 (Awarded with spotlight)

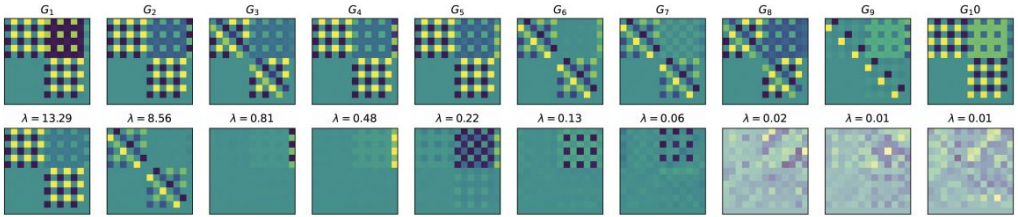
Symmetries in Dynamical Systems

Noether's razor

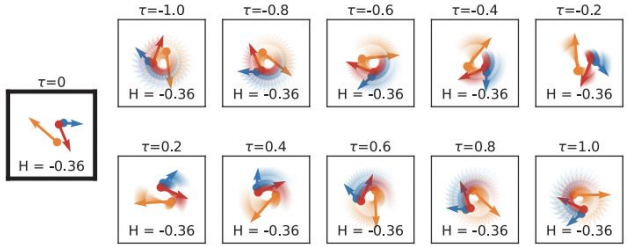
Automatic symmetry discovery in dynamical systems



Generators associated to learned conserved quantities and their singular value decomposition.



Example of rotational symmetry associated to generator G_9



Learned dynamics: **2d 3-body system**		Train data Train MSE	NLL/N	KL/N	-ELBO/N (↓)	Test data Test MSE (↓)	Test data (moved) Test MSE (↓)	Test data (wider) Test MSE (↓)
HNN		0.0028	-13.87	13.34	-9.52	0.0016	0.0035	0.0016
HNN + learned symmetry	(**ours**)	0.0017	-20.09	7.28	**-12.81**	**0.0006**	**0.0004**	**0.0006**
HNN + fixed SE(2)	(reference oracle)	0.0019	-19.27	7.96	**-11.32**	**0.0006**	**0.0006**	**0.0006**

[13] **Noether's razor**
Tycho F.A. van der Ouderaa, Mark van der Wilk, Pim de Haan
In submission.

Bayesian Neural Model Selection for Symmetry Learning

Tycho van der Ouderaa

@tychovdo

tychovdo@gmail.com

Supervised by Dr. Mark van der Wilk

Postgraduate student (PhD) at:

Imperial College London

Visiting researcher at:

UNIVERSITY OF OXFORD