Alumni talk @ CBL

Imperial College London

How *Accurate* Gaussian Processes can help Deep Learning

Mark van der Wilk

Department of Computing
Imperial College London

@markvanderwilk
m.vdwilk@imperial.ac.uk

Apr 30, 2021

It's nice to be "back"

- 2012-2017: PhD here at CBL, supervised by Carl

It's nice to be "back"

- 2012-2017: PhD here at CBL, supervised by Carl
  - 2015: Internship at Google in Mountain View
  - 2016: Occasional ML consulting jobs

It's nice to be "back"

- 2012-2017: PhD here at CBL, supervised by Carl
  - 2015: Internship at Google in Mountain View
  - 2016: Occasional ML consulting jobs
  - 2016: Qualcomm Innovation Fellowship

It's nice to be "back"

- 2012-2017: PhD here at CBL, supervised by Carl
  - 2015: Internship at Google in Mountain View
  - 2016: Occasional ML consulting jobs
  - 2016: Qualcomm Innovation Fellowship
- 2017-2019: Research Scientist at Secondmind, because James Hensman was there
- 2020–: Lecturer at Imperial College London

It's nice to be "back"

- 2012-2017: PhD here at CBL, supervised by Carl
  - 2015: Internship at Google in Mountain View
  - 2016: Occasional ML consulting jobs
  - 2016: Qualcomm Innovation Fellowship
- 2017-2019: Research Scientist at Secondmind, because James Hensman was there
- 2020–: Lecturer at Imperial College London
  - Building a research group, trying to reproduce the great environments I have seen before
  - Currently, 4 PhD candidates (who have never met each other in person)

It's nice to be "back"

- 2012-2017: PhD here at CBL, supervised by Carl
  - 2015: Internship at Google in Mountain View
  - 2016: Occasional ML consulting jobs
  - 2016: Qualcomm Innovation Fellowship
- 2017-2019: Research Scientist at Secondmind, because James Hensman was there
- 2020–: Lecturer at Imperial College London
  - Building a research group, trying to reproduce the great environments I have seen before
  - Currently, 4 PhD candidates (who have never met each other in person)

Today, I'll be talking about some research from the last few years.

Overview

Goal: Towards automatic model selection in deep learning.

Overview

Goal: Towards automatic model selection in deep learning.

Talk outline:

1. The promises of Bayesian Model Selection

Overview

Goal: Towards automatic model selection in deep learning.

Talk outline:

1. The promises of Bayesian Model Selection
2. Accurate Gaussian process inference

Overview

Goal: Towards automatic model selection in deep learning.

Talk outline:

1. The promises of Bayesian Model Selection
2. Accurate Gaussian process inference
3. GPs in Deep Learning

Overview

Goal: Towards automatic model selection in deep learning.

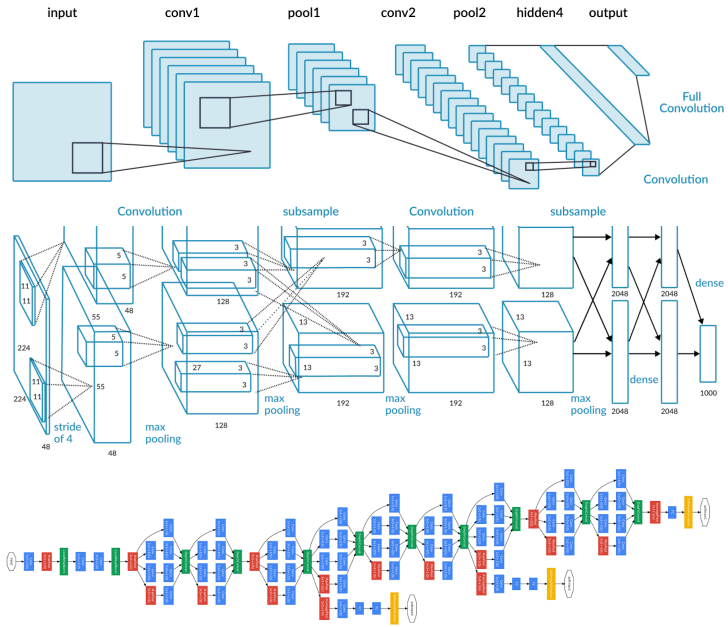
Talk outline:

1. The promises of Bayesian Model Selection
2. Accurate Gaussian process inference
3. GPs in Deep Learning
4. Other approaches: Ensembles and Architecture Search

Model Selection

Every time we train a NN we need to decide on hyperparameters:

- How many layers? How many units in a layer?
- What layer structure? Convolutional? Skip connections?
- Data augmentation parameters?



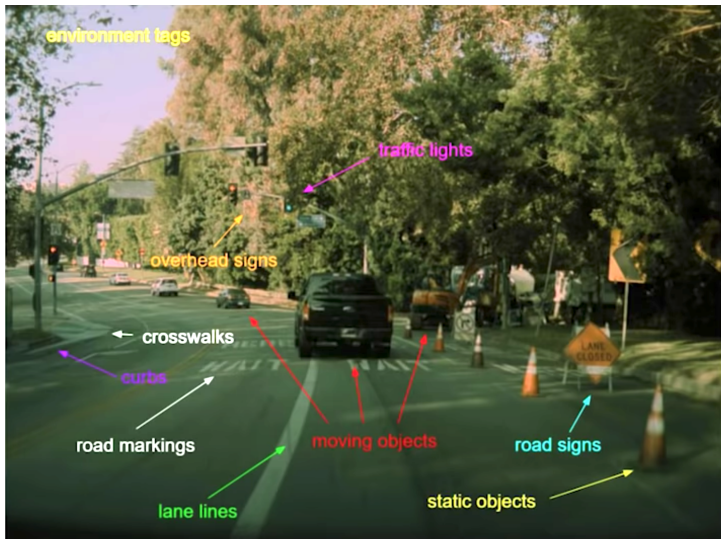
Model Selection

Every time we train a NN we need to decide on hyperparameters:

- How many layers? How many units in a layer?
- What layer structure? Convolutional? Skip connections?
- Data augmentation parameters?

As architectures get more complex, so does design! E.g. multitask.

- Which layers to share?
- What kind of task-specific layers?
- How much capacity to assign to each task?



[Karpathy, ICML 2019]

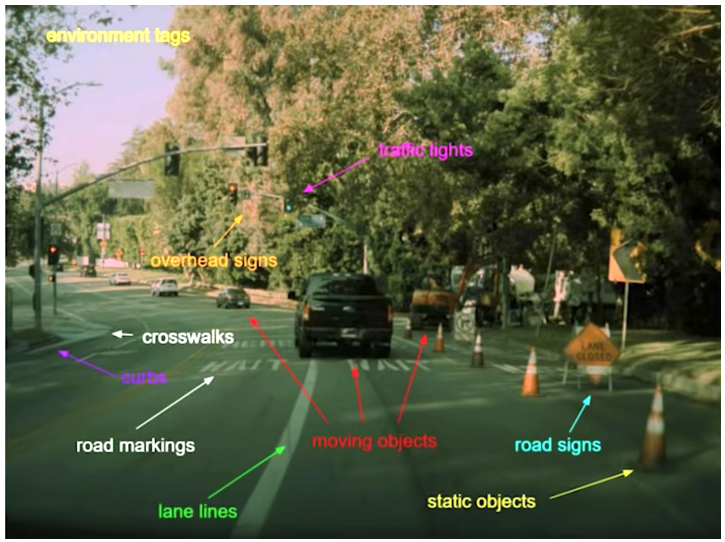
Model Selection

Every time we train a NN we need to decide on hyperparameters:

- How many layers? How many units in a layer?
- What layer structure? Convolutional? Skip connections?
- Data augmentation parameters?

As architectures get more complex, so does design! E.g. multitask.

- Which layers to share?
- What kind of task-specific layers?
- How much capacity to assign to each task?



[Karpathy, ICML 2019]

Main tool is **crossvalidation**.

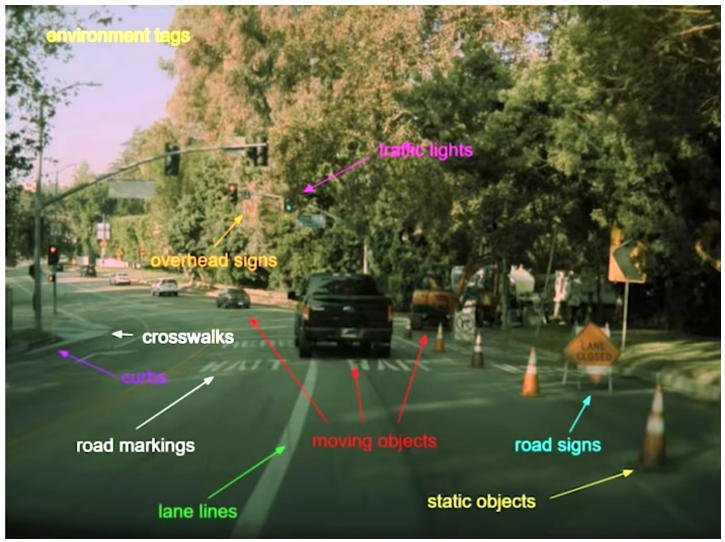
Model Selection

Every time we train a NN we need to decide on hyperparameters:

- How many layers? How many units in a layer?
- What layer structure? Convolutional? Skip connections?
- Data augmentation parameters?

As architectures get more complex, so does design! E.g. multitask.

- Which layers to share?
- What kind of task-specific layers?
- How much capacity to assign to each task?



[Karpathy, ICML 2019]

Main tool is **crossvalidation**.

Goal: Make it as easy as learning weights.

Overview

The Promise of Bayesian Model Selection

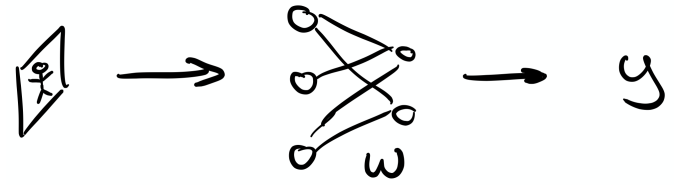
Accurate Inference in Gaussian Processes

Gaussian Processes in Deep Learning

Ensembles and Architecture Search

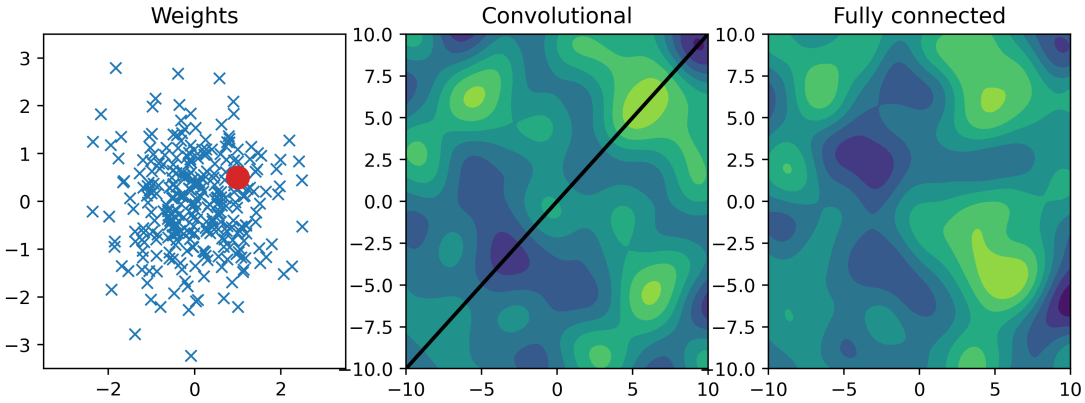
Conclusion

Bayesian Inference



$$f_{\mathbf{w}} : \mathbb{R}^D \to \mathbb{R}^C \quad (1)$$

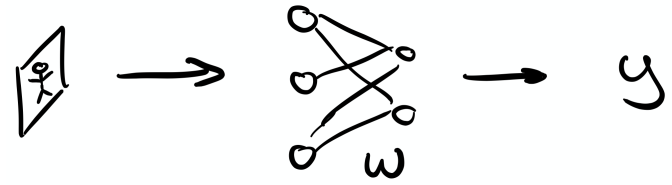
$$\mathbf{x} \mapsto f_{\mathbf{w}}(\mathbf{x}) \quad (2)$$



- A prior on parameters leads to a prior on functions
- Architectural **hyperparameters** influence prior on functions
- BDL focusses mostly on uncertainty in the function:

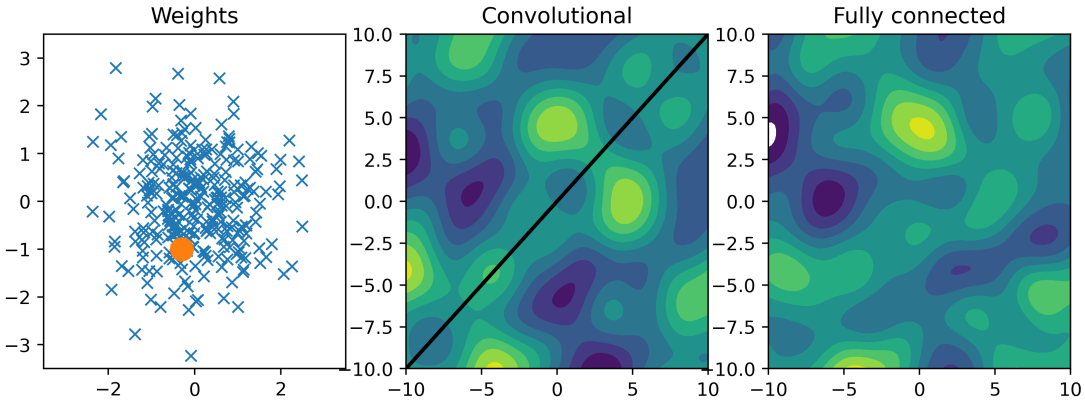
$$p(f|\mathbf{y},\theta) = \frac{p(\mathbf{y}|f)p(f|\theta)}{p(\mathbf{y}|\theta)} \quad (3)$$

Bayesian Inference



$$f_{\mathbf{w}} : \mathbb{R}^D \to \mathbb{R}^C \quad (1)$$

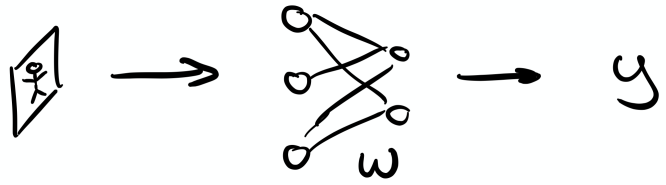
$$\mathbf{x} \mapsto f_{\mathbf{w}}(\mathbf{x}) \quad (2)$$



- A prior on parameters leads to a prior on functions
- Architectural **hyperparameters** influence prior on functions
- BDL focusses mostly on uncertainty in the function:

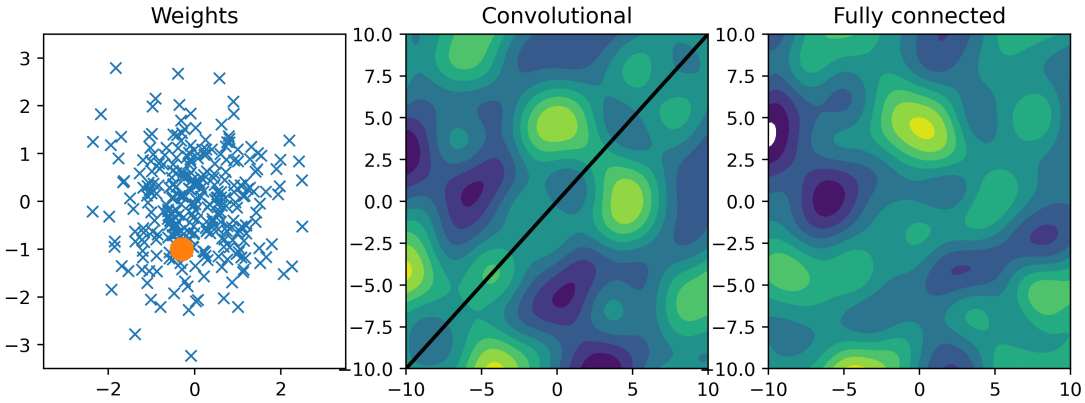
$$p(f|\mathbf{y},\theta) = \frac{p(\mathbf{y}|f)p(f|\theta)}{p(\mathbf{y}|\theta)} \quad (3)$$

Bayesian Inference



$$f_{\mathbf{w}} : \mathbb{R}^D \to \mathbb{R}^C \quad (1)$$

$$\mathbf{x} \mapsto f_{\mathbf{w}}(\mathbf{x}) \quad (2)$$



- A prior on parameters leads to a prior on functions
- Architectural **hyperparameters** influence prior on functions
- BDL focusses mostly on uncertainty in the function:

$$p(f|\mathbf{y},\theta) = \frac{p(\mathbf{y}|f)p(f|\theta)}{p(\mathbf{y}|\theta)} \quad (3)$$

But we want to determine the hyperparameters θ too!

Bayesian Model Selection

Bayes tells us: Just find the posterior over all your unknowns!

$$p(f, \theta | \mathbf{y}) = \frac{p(\mathbf{y}|f)p(f|\theta)p(\theta)}{p(\mathbf{y})} \tag{4}$$

Bayesian Model Selection

Bayes tells us: Just find the posterior over all your unknowns!

$$p(f, \theta|\mathbf{y}) = \frac{p(\mathbf{y}|f)p(f|\theta)p(\theta)}{p(\mathbf{y})} = \underbrace{\frac{p(\mathbf{y}|f)p(f|\theta)}{p(\mathbf{y}|\theta)}}_{\text{usual posterior}} \underbrace{\frac{p(\mathbf{y}|\theta)p(\theta)}{p(\mathbf{y})}}_{\text{hyper posterior}} \tag{4}$$

Bayesian Model Selection

Bayes tells us: Just find the posterior over all your unknowns!

$$p(f, \theta|\mathbf{y}) = \frac{p(\mathbf{y}|f)p(f|\theta)p(\theta)}{p(\mathbf{y})} = \underbrace{\frac{p(\mathbf{y}|f)p(f|\theta)}{p(\mathbf{y}|\theta)}}_{\text{usual posterior}} \underbrace{\frac{p(\mathbf{y}|\theta)p(\theta)}{p(\mathbf{y})}}_{\text{hyper posterior}} \tag{4}$$

- Posterior over functions is unchanged!
- Posterior over hyperparams requires **marginal likelihood**:

$$p(\mathbf{y}|\theta) = \int p(\mathbf{y} \mid f)p(f \mid \theta)\mathrm{d}\theta \tag{5}$$

Bayesian Model Selection

Bayes tells us: Just find the posterior over all your unknowns!

$$p(f, \theta|\mathbf{y}) = \frac{p(\mathbf{y}|f)p(f|\theta)p(\theta)}{p(\mathbf{y})} = \underbrace{\frac{p(\mathbf{y}|f)p(f|\theta)}{p(\mathbf{y}|\theta)}}_{\text{usual posterior}} \underbrace{\frac{p(\mathbf{y}|\theta)p(\theta)}{p(\mathbf{y})}}_{\text{hyper posterior}} \tag{4}$$

- Posterior over functions is unchanged!
- Posterior over hyperparams requires **marginal likelihood**:

$$p(\mathbf{y}|\theta) = \int p(\mathbf{y} \mid f)p(f \mid \theta)\mathrm{d}\theta \tag{5}$$

Bayesian model selection is commonly done by ML-II (Berger, 1985):

$$\theta^* = \operatorname*{argmax}_{\theta} \log p(\mathbf{y} \mid \theta)\,, \qquad \text{predict using } p(f|\mathbf{y}, \theta^*) \tag{6}$$

Bayesian Model Selection

Bayes tells us: Just find the posterior over all your unknowns!

$$p(f, \theta|\mathbf{y}) = \frac{p(\mathbf{y}|f)p(f|\theta)p(\theta)}{p(\mathbf{y})} = \underbrace{\frac{p(\mathbf{y}|f)p(f|\theta)}{p(\mathbf{y}|\theta)}}_{\text{usual posterior}} \underbrace{\frac{p(\mathbf{y}|\theta)p(\theta)}{p(\mathbf{y})}}_{\text{hyper posterior}} \tag{4}$$

- Posterior over functions is unchanged!
- Posterior over hyperparams requires **marginal likelihood**:

$$p(\mathbf{y}|\theta) = \int p(\mathbf{y} \mid f)p(f \mid \theta)\mathrm{d}\theta \tag{5}$$

Bayesian model selection is commonly done by ML-II (Berger, 1985):

$$\theta^* = \operatorname*{argmax}_{\theta} \log p(\mathbf{y} \mid \theta)\,, \qquad \text{predict using } p(f|\mathbf{y}, \theta^*) \tag{6}$$

Gradient-based optimisation is **super convenient**!

Bayesian Model Selection

Bayes tells us: Just find the posterior over all your unknowns!

$$p(f, \theta|\mathbf{y}) = \frac{p(\mathbf{y}|f)p(f|\theta)p(\theta)}{p(\mathbf{y})} = \underbrace{\frac{p(\mathbf{y}|f)p(f|\theta)}{p(\mathbf{y}|\theta)}}_{\text{usual posterior}} \underbrace{\frac{p(\mathbf{y}|\theta)p(\theta)}{p(\mathbf{y})}}_{\text{hyper posterior}} \tag{4}$$

- Posterior over functions is unchanged!
- Posterior over hyperparams requires **marginal likelihood**:

$$p(\mathbf{y}|\theta) = \int p(\mathbf{y} \mid f)p(f \mid \theta)\mathrm{d}\theta \tag{5}$$

Bayesian model selection is commonly done by ML-II (Berger, 1985):

$$\theta^* = \operatorname*{argmax}_{\theta} \log p(\mathbf{y} \mid \theta)\,, \qquad \text{predict using } p(f|\mathbf{y}, \theta^*) \tag{6}$$

Gradient-based optimisation is **super convenient**!

... if we can compute $p(\mathbf{y} \mid \theta)$

Variational Bayesian Model Selection

Bayes tells us what to do, but not how to do it. Variational inference actually does it, and gives us

- An approximate posterior
- An estimate of the marginal likelihood! (lower bound)

Variational Bayesian Model Selection

Bayes tells us what to do, but not how to do it. Variational inference actually does it, and gives us

- An approximate posterior
- An estimate of the marginal likelihood! (lower bound)

$$\log p(\mathbf{y} \mid \theta) = \mathcal{L}(\phi, \theta) + \mathrm{KL}\big[q_\phi(f) || p(f|\mathbf{y}, \theta)\big]$$

$$\geqslant \mathcal{L} = \sum_{n=1}^{N} \mathbb{E}_{q_\phi(f(\mathbf{x}_n))}[\log p(y_n|f(\mathbf{x}_n))] - \mathrm{KL}\big[q_\phi(f) || p(f|\theta)\big]$$

Variational Bayesian Model Selection

Bayes tells us what to do, but not how to do it. Variational inference actually does it, and gives us

- An approximate posterior
- An estimate of the marginal likelihood! (lower bound)

$$\log p(\mathbf{y} \mid \theta) = \mathcal{L}(\phi, \theta) + \mathrm{KL}\big[q_\phi(f) || p(f|\mathbf{y}, \theta)\big]$$
$$\geqslant \mathcal{L} = \sum_{n=1}^{N} \mathbb{E}_{q_\phi(f(\mathbf{x}_n))}[\log p(y_n|f(\mathbf{x}_n))] - \mathrm{KL}\big[q_\phi(f) || p(f|\theta)\big]$$

- Find posterior and hyperparameters simultaneously by

$$\underset{\phi,\theta}{\operatorname{argmax}}\, \mathcal{L}(\phi, \theta) \tag{7}$$

Variational Bayesian Model Selection

Bayes tells us what to do, but not how to do it. Variational inference actually does it, and gives us

- An approximate posterior
- An estimate of the marginal likelihood! (lower bound)

$$\log p(\mathbf{y} \mid \theta) = \mathcal{L}(\phi, \theta) + \mathrm{KL}\big[q_\phi(f) || p(f|\mathbf{y}, \theta)\big]$$
$$\geqslant \mathcal{L} = \sum_{n=1}^{N} \mathbb{E}_{q_\phi(f(\mathbf{x}_n))}[\log p(y_n|f(\mathbf{x}_n))] - \mathrm{KL}\big[q_\phi(f) || p(f|\theta)\big]$$

- Find posterior and hyperparameters simultaneously by

$$\underset{\phi,\theta}{\operatorname{argmax}}\, \mathcal{L}(\phi, \theta) \tag{7}$$

- Quality of posterior is linked to the accuracy of lower bound!

Learning Invariances

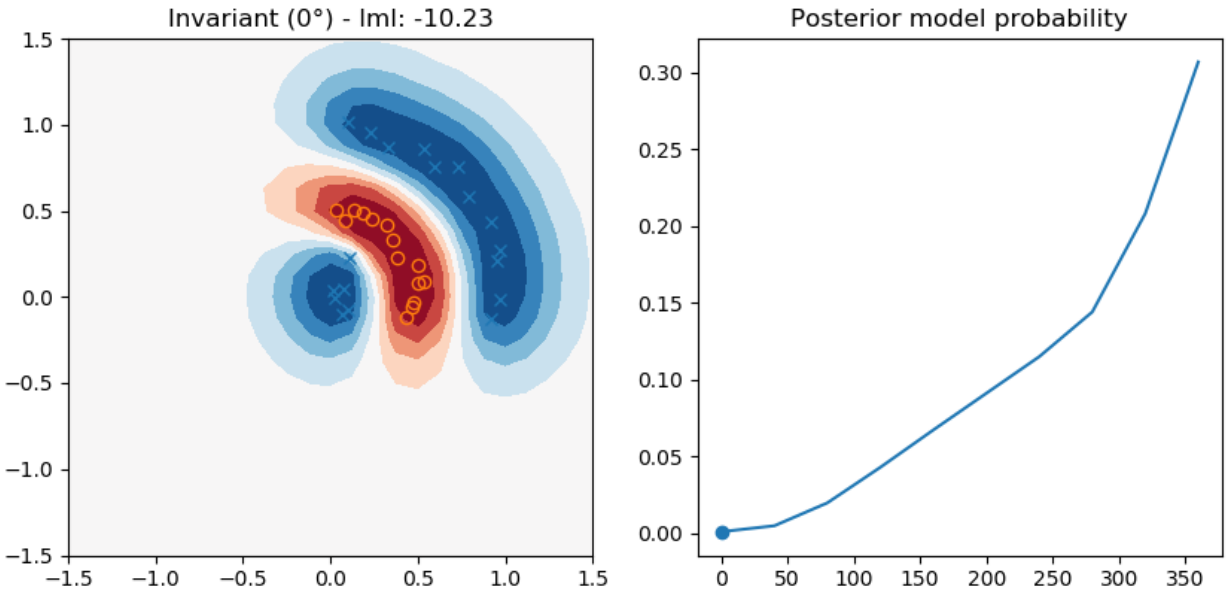
Learning Invariances using the Marginal Likelihood

Mark van der Wilk
PROWLER.io
Cambridge, UK
mark@prowler.io

Matthias Bauer
MPI for Intelligent Systems
University of Cambridge
msb55@cam.ac.uk

ST John
PROWLER.io
Cambridge, UK
st@prowler.io

James Hensman
PROWLER.io
Cambridge, UK
james@prowler.io



https://twitter.com/markvanderwilk/status/1184235600414683136

Learning Invariances

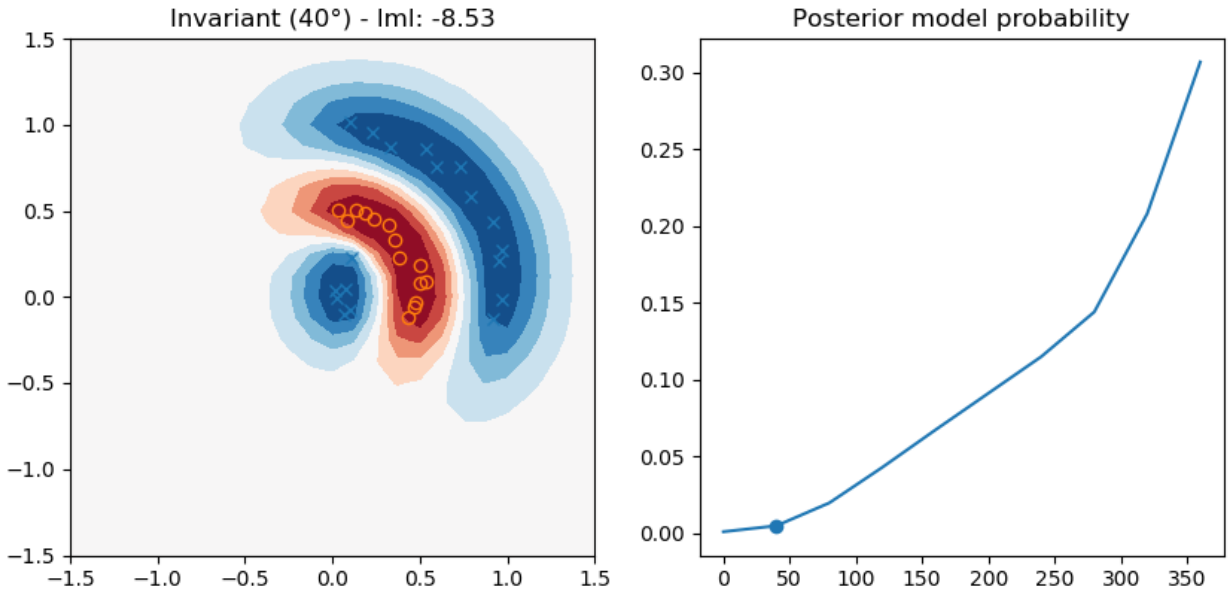
Learning Invariances using the Marginal Likelihood

Mark van der Wilk
PROWLER.io
Cambridge, UK
mark@prowler.io

Matthias Bauer
MPI for Intelligent Systems
University of Cambridge
msb55@cam.ac.uk

ST John
PROWLER.io
Cambridge, UK
st@prowler.io

James Hensman
PROWLER.io
Cambridge, UK
james@prowler.io



https://twitter.com/markvanderwilk/status/1184235600414683136

Learning Invariances

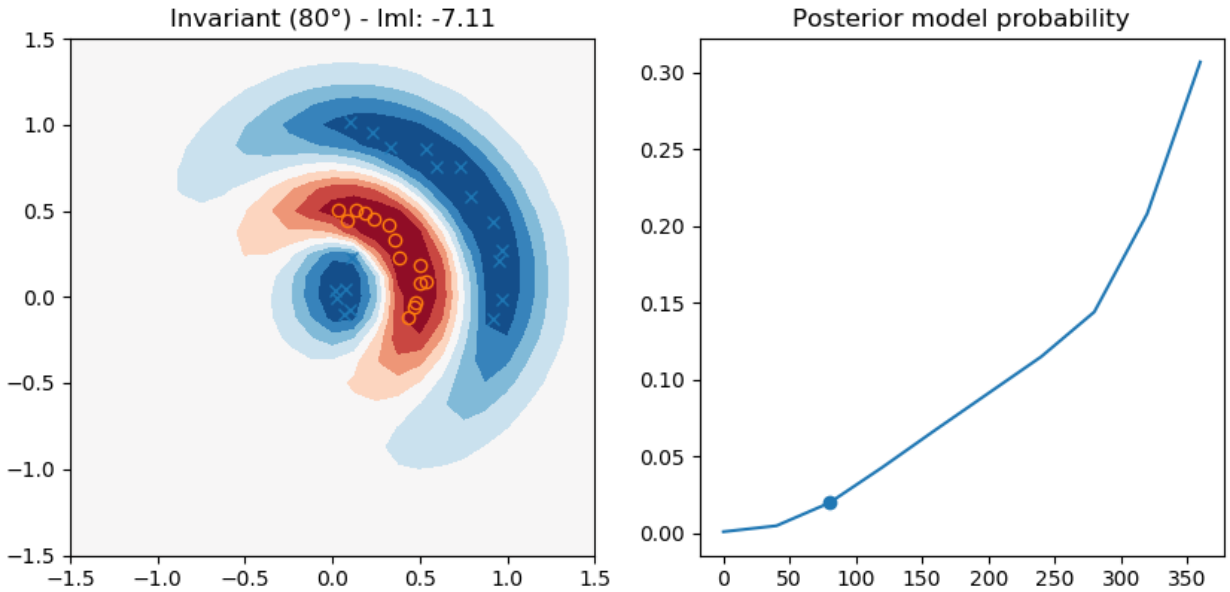
Learning Invariances using the Marginal Likelihood

Mark van der Wilk
PROWLER.io
Cambridge, UK
mark@prowler.io

Matthias Bauer
MPI for Intelligent Systems
University of Cambridge
msb55@cam.ac.uk

ST John
PROWLER.io
Cambridge, UK
st@prowler.io

James Hensman
PROWLER.io
Cambridge, UK
james@prowler.io



https://twitter.com/markvanderwilk/status/1184235600414683136

Learning Invariances

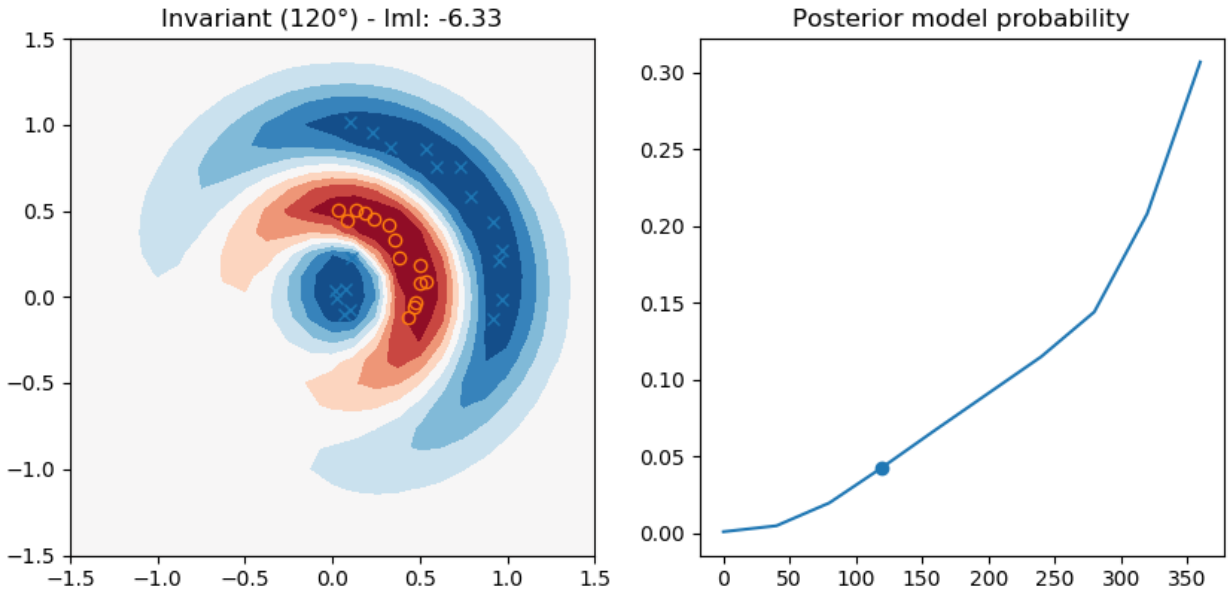
Learning Invariances using the Marginal Likelihood

Mark van der Wilk
PROWLER.io
Cambridge, UK
mark@prowler.io

Matthias Bauer
MPI for Intelligent Systems
University of Cambridge
msb55@cam.ac.uk

ST John
PROWLER.io
Cambridge, UK
st@prowler.io

James Hensman
PROWLER.io
Cambridge, UK
james@prowler.io



https://twitter.com/markvanderwilk/status/1184235600414683136

Learning Invariances

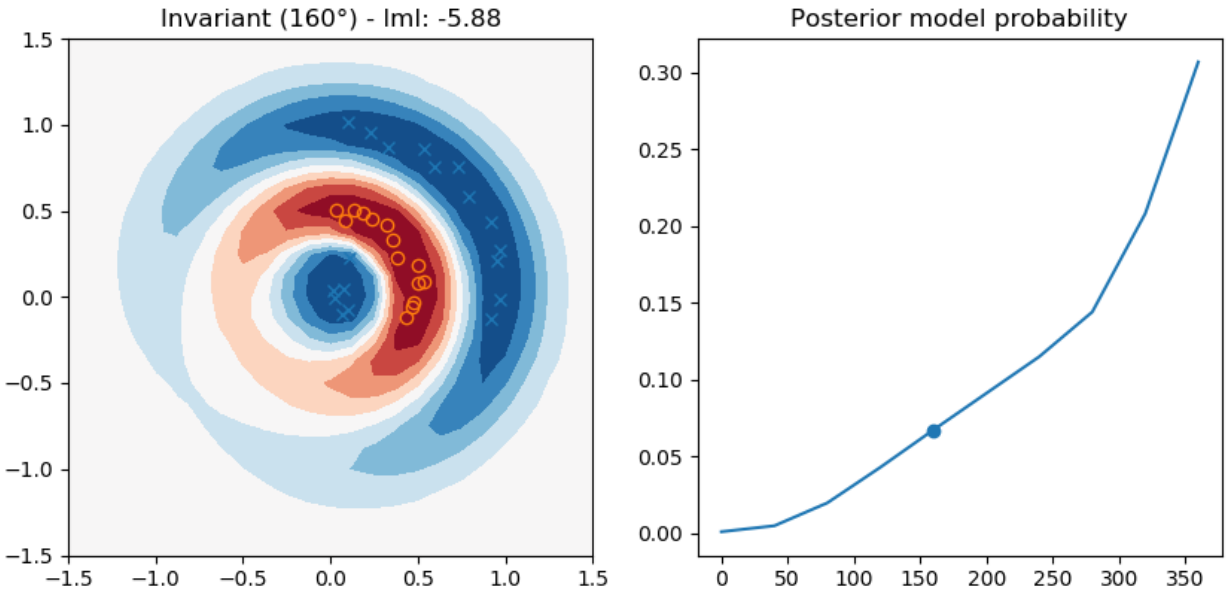
Learning Invariances using the Marginal Likelihood

Mark van der Wilk
PROWLER.io
Cambridge, UK
mark@prowler.io

Matthias Bauer
MPI for Intelligent Systems
University of Cambridge
msb55@cam.ac.uk

ST John
PROWLER.io
Cambridge, UK
st@prowler.io

James Hensman
PROWLER.io
Cambridge, UK
james@prowler.io



https://twitter.com/markvanderwilk/status/1184235600414683136

Learning Invariances

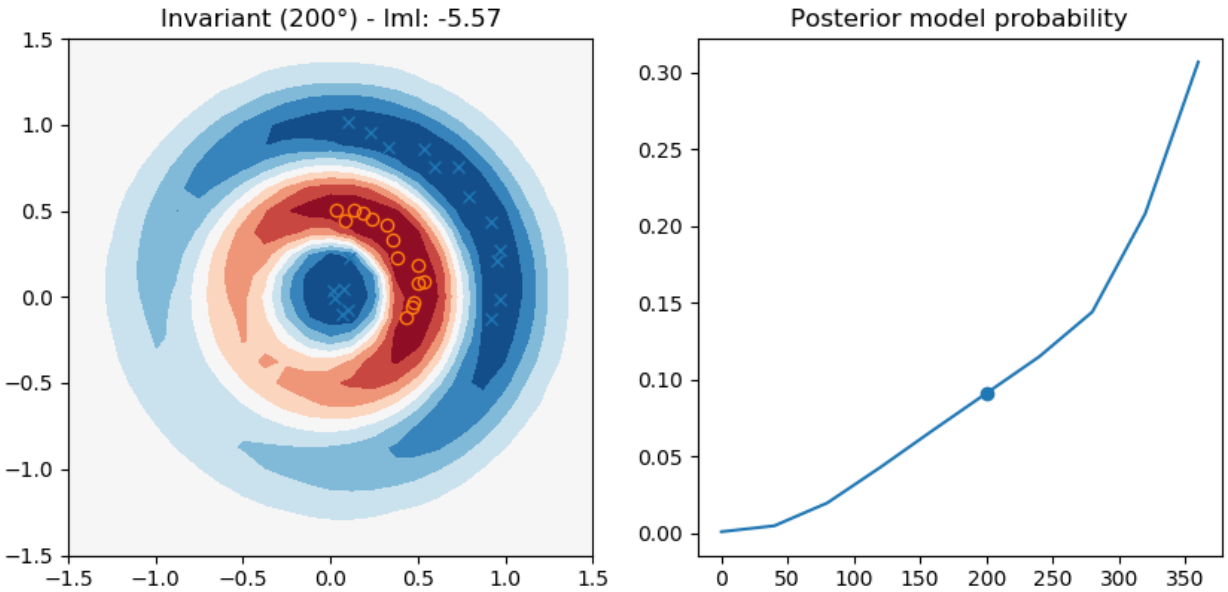
Learning Invariances using the Marginal Likelihood

Mark van der Wilk
PROWLER.io
Cambridge, UK
mark@prowler.io

Matthias Bauer
MPI for Intelligent Systems
University of Cambridge
msb55@cam.ac.uk

ST John
PROWLER.io
Cambridge, UK
st@prowler.io

James Hensman
PROWLER.io
Cambridge, UK
james@prowler.io



https://twitter.com/markvanderwilk/status/1184235600414683136

Learning Invariances

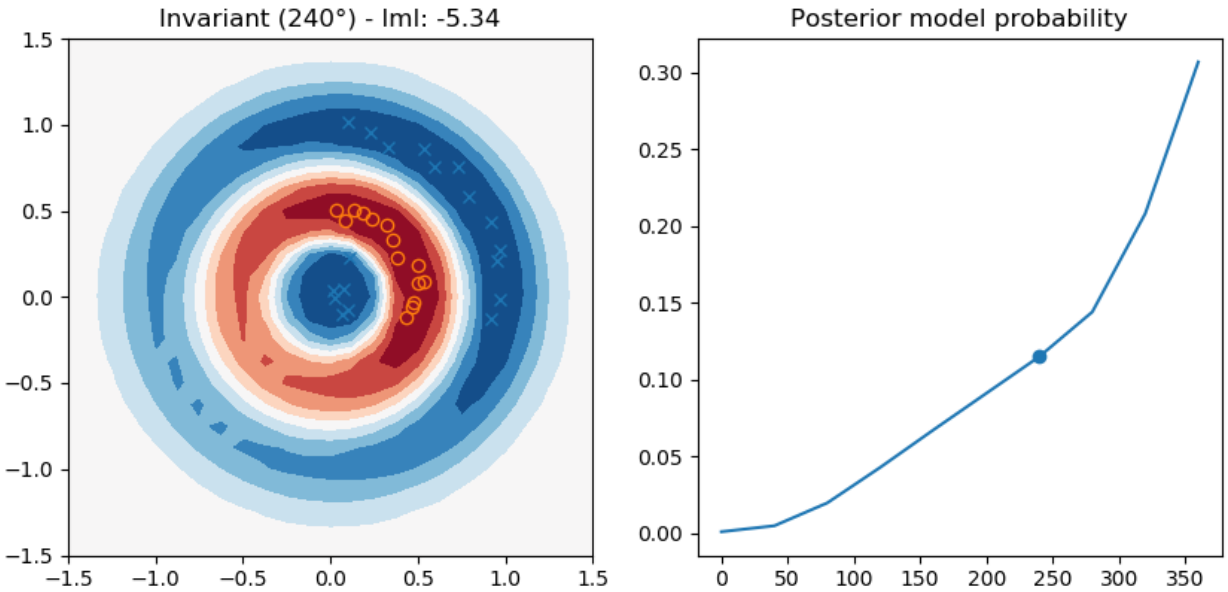
Learning Invariances using the Marginal Likelihood

Mark van der Wilk
PROWLER.io
Cambridge, UK
mark@prowler.io

Matthias Bauer
MPI for Intelligent Systems
University of Cambridge
msb55@cam.ac.uk

ST John
PROWLER.io
Cambridge, UK
st@prowler.io

James Hensman
PROWLER.io
Cambridge, UK
james@prowler.io



https://twitter.com/markvanderwilk/status/1184235600414683136

Learning Invariances

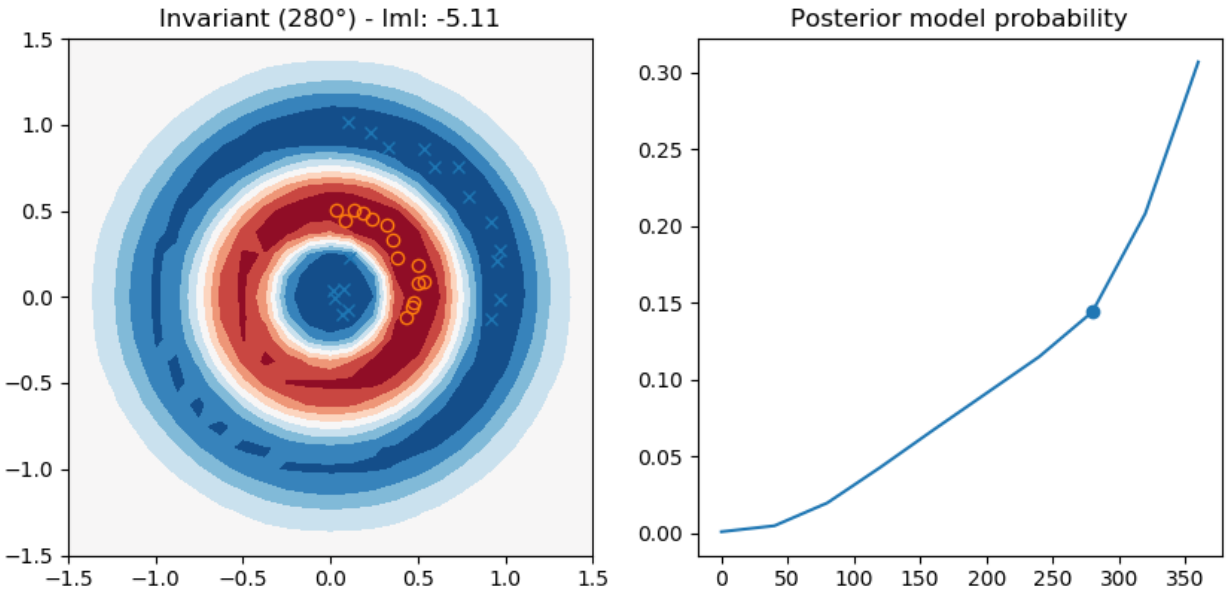
Learning Invariances using the Marginal Likelihood

Mark van der Wilk
PROWLER.io
Cambridge, UK
mark@prowler.io

Matthias Bauer
MPI for Intelligent Systems
University of Cambridge
msb55@cam.ac.uk

ST John
PROWLER.io
Cambridge, UK
st@prowler.io

James Hensman
PROWLER.io
Cambridge, UK
james@prowler.io



https://twitter.com/markvanderwilk/status/1184235600414683136

Learning Invariances

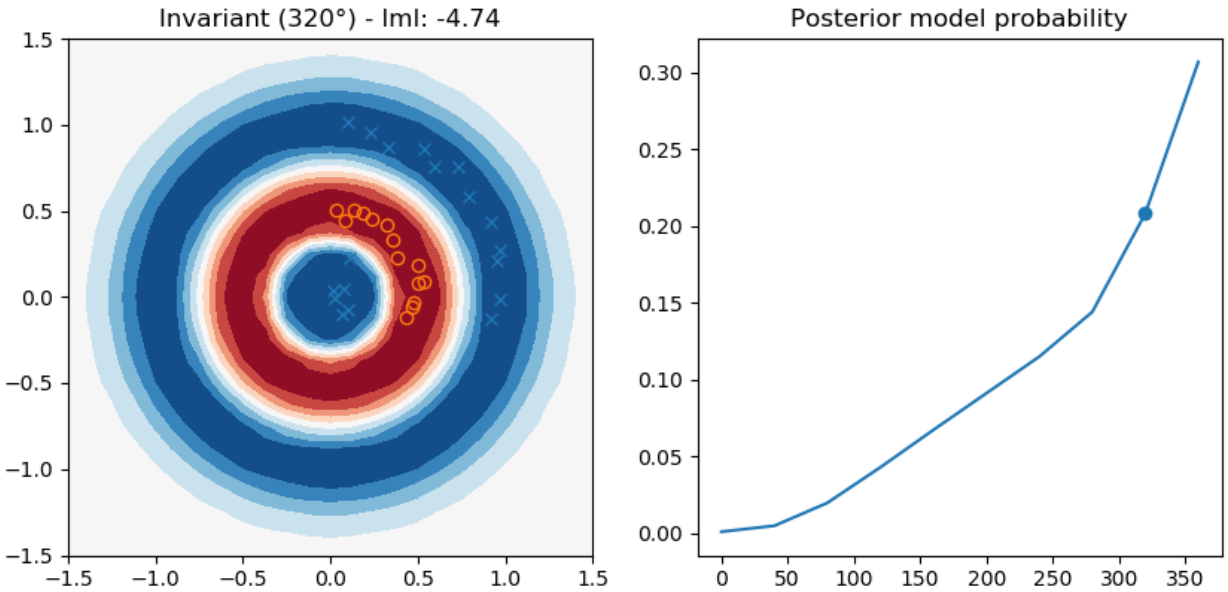
Learning Invariances using the Marginal Likelihood

Mark van der Wilk
PROWLER.io
Cambridge, UK
mark@prowler.io

Matthias Bauer
MPI for Intelligent Systems
University of Cambridge
msb55@cam.ac.uk

ST John
PROWLER.io
Cambridge, UK
st@prowler.io

James Hensman
PROWLER.io
Cambridge, UK
james@prowler.io



https://twitter.com/markvanderwilk/status/1184235600414683136

Learning Invariances

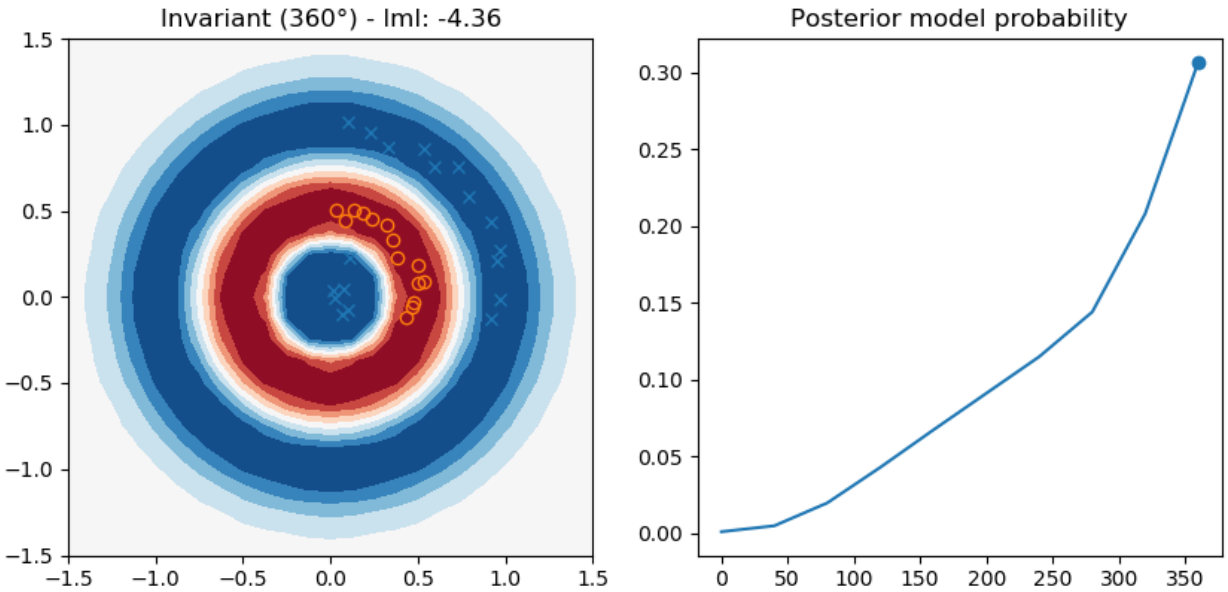
Learning Invariances using the Marginal Likelihood

Mark van der Wilk
PROWLER.io
Cambridge, UK
mark@prowler.io

Matthias Bauer
MPI for Intelligent Systems
University of Cambridge
msb55@cam.ac.uk

ST John
PROWLER.io
Cambridge, UK
st@prowler.io

James Hensman
PROWLER.io
Cambridge, UK
james@prowler.io



https://twitter.com/markvanderwilk/status/1184235600414683136

Learning Invariances

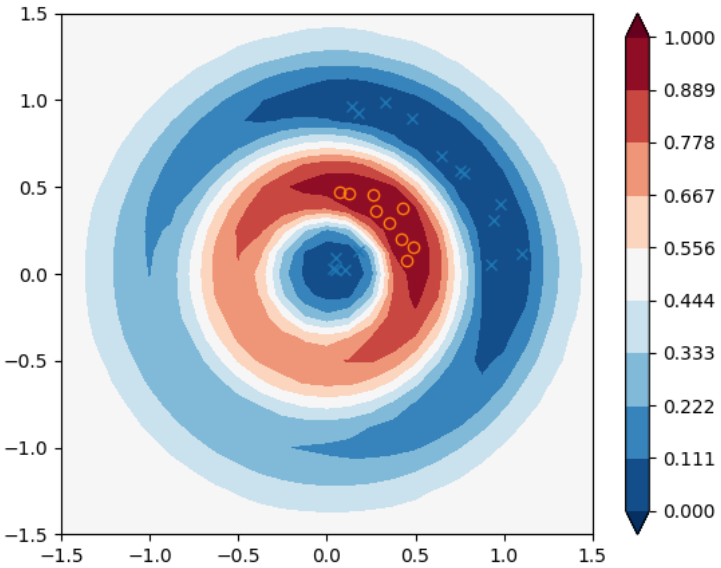
Learning Invariances using the Marginal Likelihood

Mark van der Wilk
PROWLER.io
Cambridge, UK
mark@prowler.io

Matthias Bauer
MPI for Intelligent Systems
University of Cambridge
msb55@cam.ac.uk

ST John
PROWLER.io
Cambridge, UK
st@prowler.io

James Hensman
PROWLER.io
Cambridge, UK
james@prowler.io



https://twitter.com/markvanderwilk/status/1184235600414683136

Learning Invariances

Learning Invariances using the Marginal Likelihood

Mark van der Wilk
PROWLER.io
Cambridge, UK
mark@prowler.io

Matthias Bauer
MPI for Intelligent Systems
University of Cambridge
msb55@cam.ac.uk

ST John
PROWLER.io
Cambridge, UK
st@prowler.io

James Hensman
PROWLER.io
Cambridge, UK
james@prowler.io

- (First?[a]) Bayesian treatment of data augmentation
- *Learnable* invariance / data augmentation using **gradients** and only the **training set**
  - Learnable means: Same set-up learns different invariance for different dataset

[a] https://statmodeling.stat.columbia.edu/2019/12/02/a-bayesian-view-of-data-augmentation/

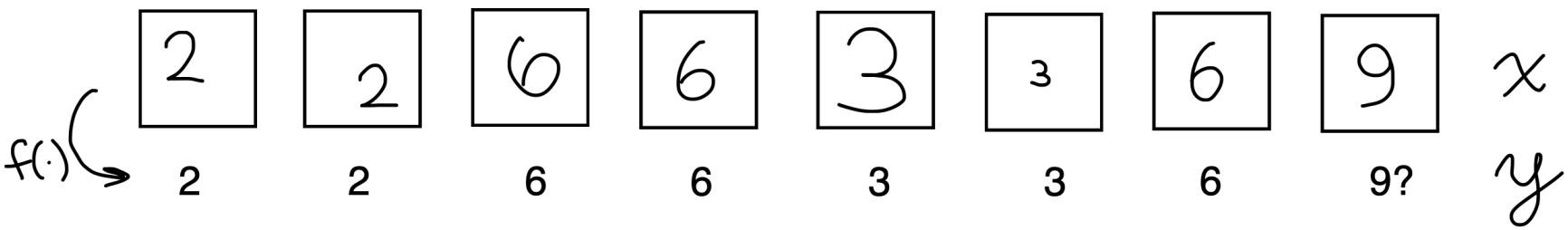
The Ingredients

We need

The Ingredients

We need

1. A way to **constrain** our learnable function to be invariant



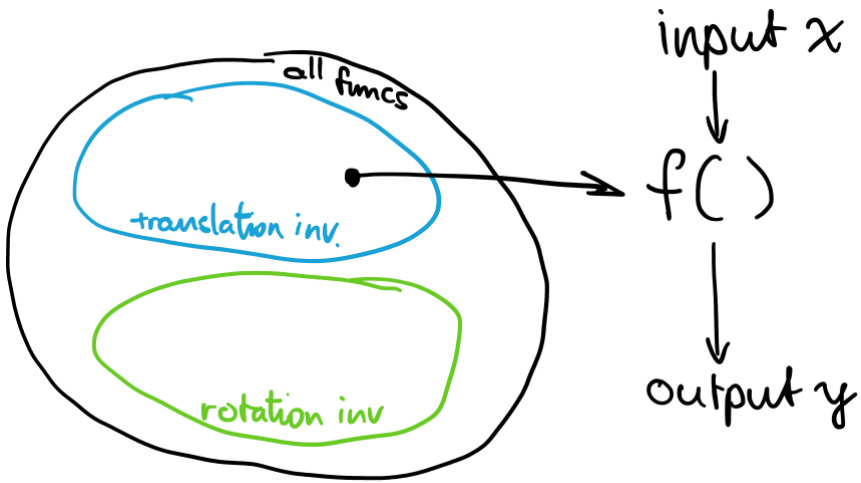
$$f(\mathbf{x}) \approx f(t(\mathbf{x};\alpha)) \qquad \forall \alpha \in \mathcal{A}_\theta$$

$$P\Big([f(t(\mathbf{x};\alpha)) - f(\mathbf{x})]^2 > L\Big) < \delta \qquad \alpha \sim p(\alpha \mid \theta)$$

The Ingredients

We need

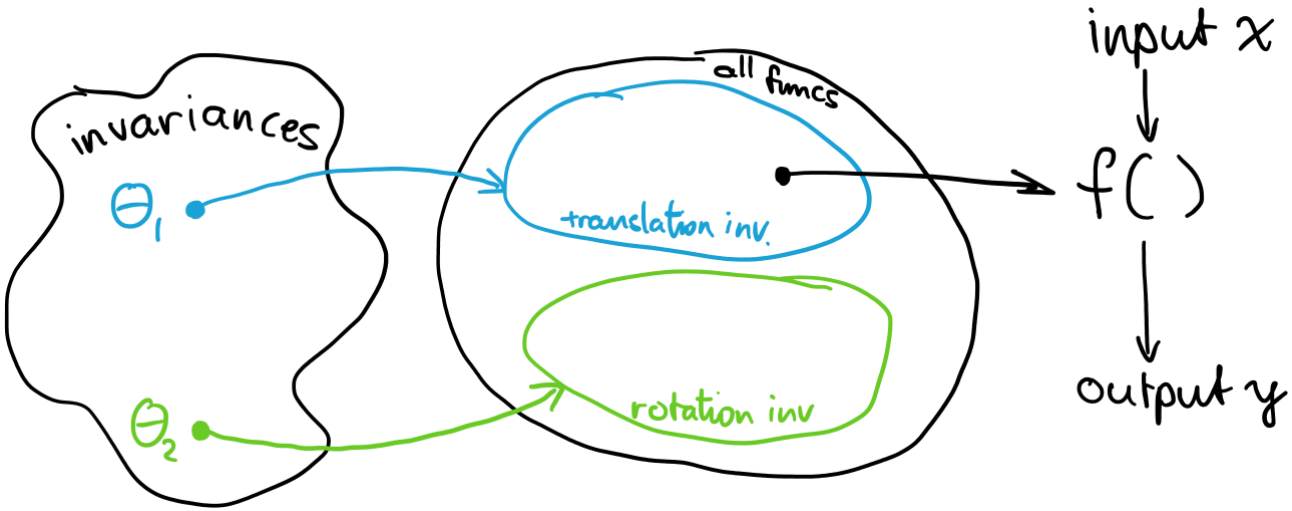
1. A way to **constrain** our learnable function to be invariant



The Ingredients

We need

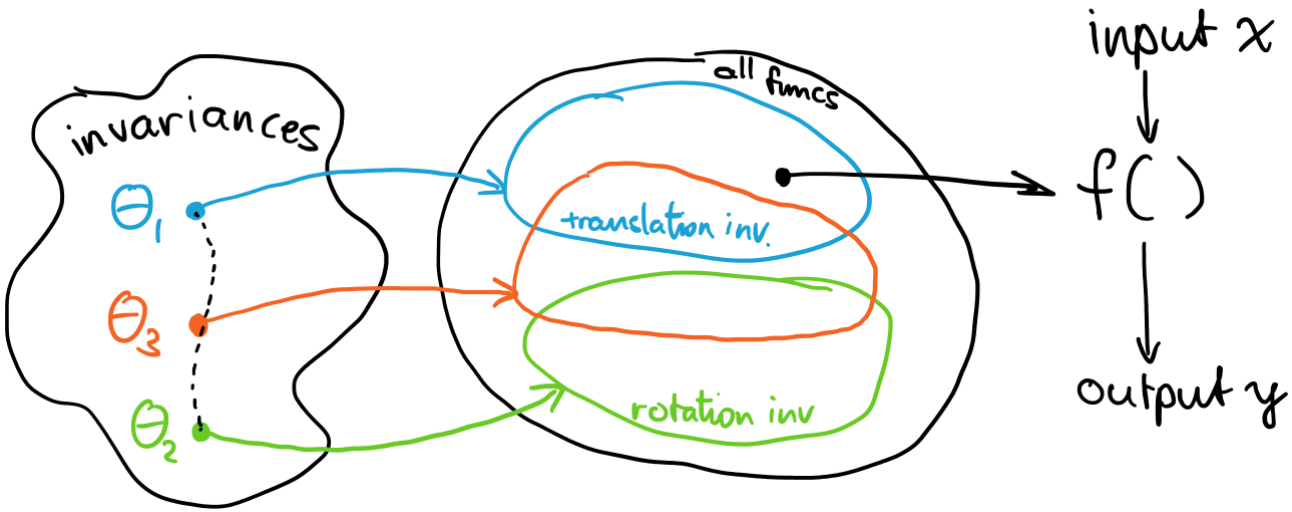
1. A way to **constrain** our learnable function to be invariant
2. A way to **parameterise** different sets of invariant functions.



The Ingredients

We need

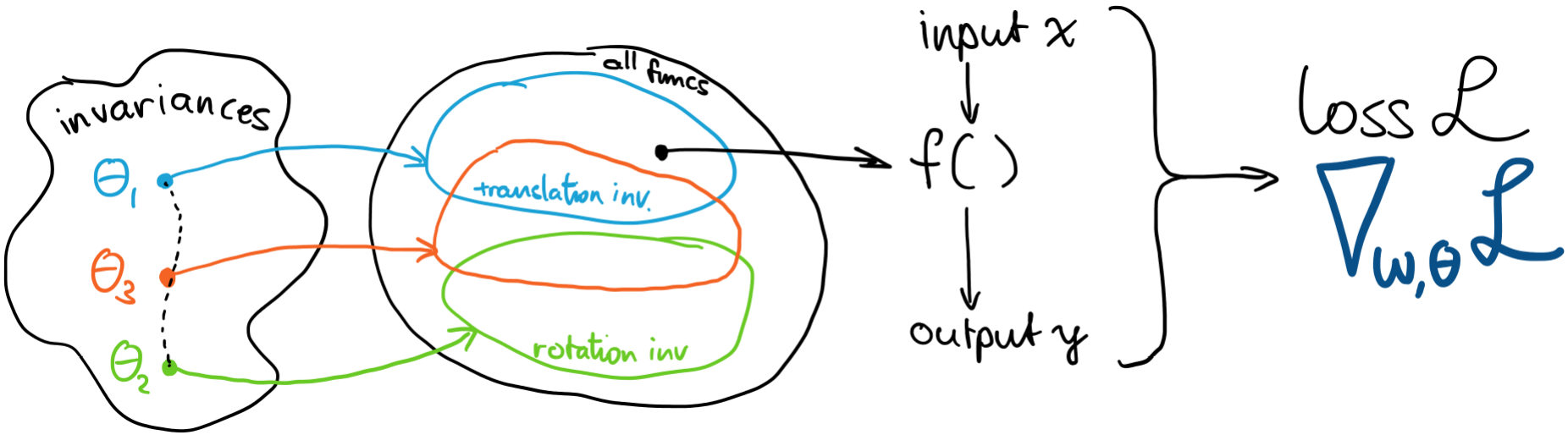
1. A way to **constrain** our learnable function to be invariant
2. A way to **parameterise** different sets of invariant functions. **Differentiably**.



The Ingredients

We need

1. A way to **constrain** our learnable function to be invariant
2. A way to **parameterise** different sets of invariant functions. **Differentiably**.
3. An objective function for learning **both** the function (i.e. weights), and invariance (i.e. θ): The ELBO $\mathcal{L}(q(f), \theta)$.



Invariant functions (skip)

How do we parameterise the invariant function $f(\cdot)$?

- Sum (convolve) a non-invariant function over set of transformations we want to be invariant to!

Strict invariance (i.e. $f(\mathbf{x}) = f(t(\mathbf{x}; \alpha))$ with exact equality):

$$f(\mathbf{x}; \mathbf{u}, \theta) = \sum_{\alpha \in \mathcal{A}_\theta} g(t(\mathbf{x}; \alpha); \mathbf{u})$$

Weak invariance / data augmentation:

$$f(\mathbf{x}; \mathbf{u}, \theta) = \int g(t(\mathbf{x}; \alpha); \mathbf{u}) p(\alpha \mid \theta) \mathrm{d}\alpha$$

The function $g(\cdot; \mathbf{u})$ is parameterised by $\mathbf{u}$ and can be seen as a Gaussian process or a single-layer NN.

Training procedure (skip)

1. Generate a sample of transformed images (reparam trick $p(\alpha \mid \theta)$):

$$\{\mathbf{x}^{(s)} = t(\mathbf{x}, \alpha^{(s)})\}_{s=1}^{S} \qquad \alpha^{(s)} = h(\epsilon^{(s)}, \theta) \qquad \epsilon^{(s)} \overset{iid}{\sim} p(\epsilon)$$

2. Monte Carlo estimate of invariant function $f(\mathbf{x})$:

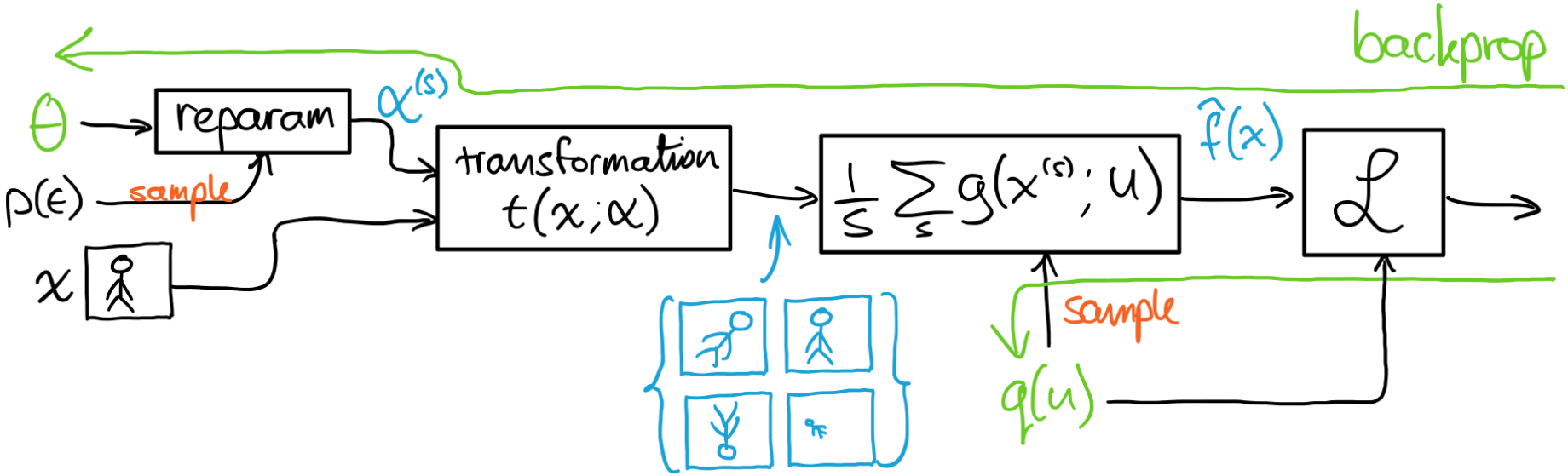
$$\hat{f}(\mathbf{x}) = \frac{1}{S} \sum_{s=1}^{S} g(\mathbf{x}^{(s)}; \mathbf{u})$$

3. Compute unbiased estimate ELBO using MC estimate of $f(\mathbf{x})$:

$$\mathcal{L} = N \cdot \mathbb{E}_{q(\mathbf{u})} \Big[\log p(y_n | \hat{f}(\mathbf{x}_n) \Big] - \mathrm{KL}[q(\mathbf{u}) || p(\mathbf{u}|\theta)]$$

4. Backpropagate to get gradients!

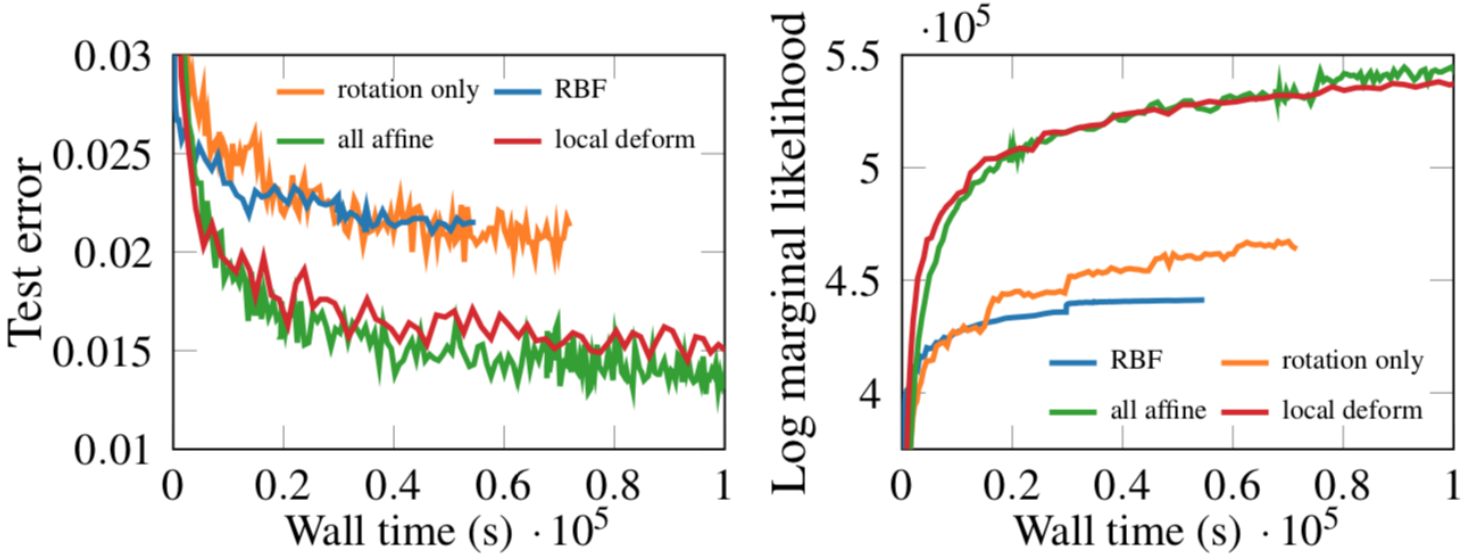
Training procedure



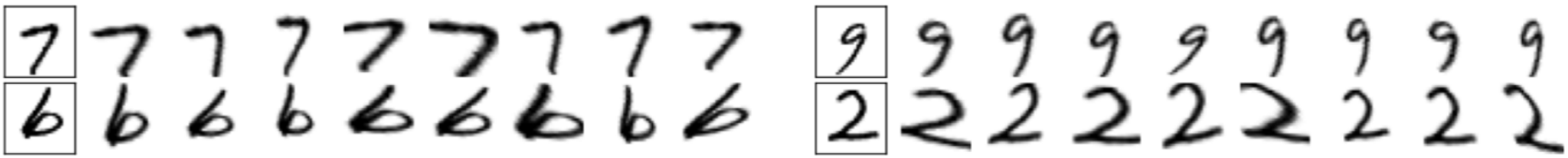
- Be Bayesian about the function $g(\cdot; \mathbf{u})$
- Averaging the output of $g(\cdot; \mathbf{u})$ (data aug)
- Compute ELBO (approximate marginal likelihood)
- Backpropagate

Results: MNIST

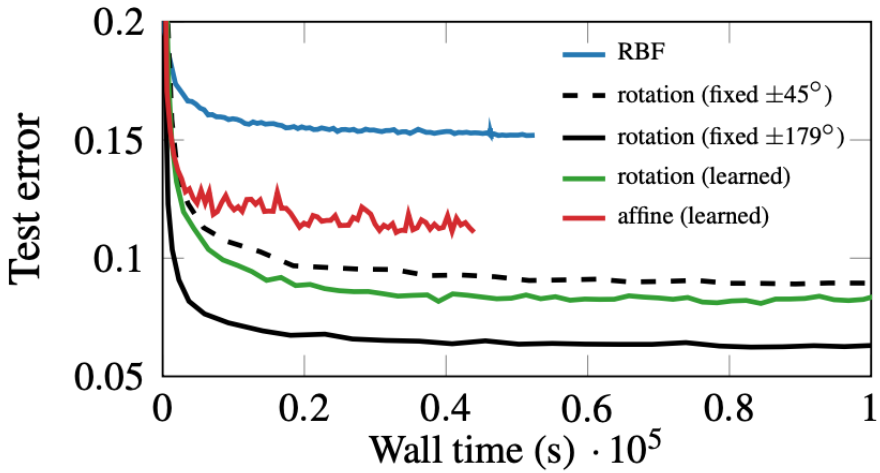
- Learns the invariance parameters through backprop
- Makes a weak Gaussian process classifier much stronger

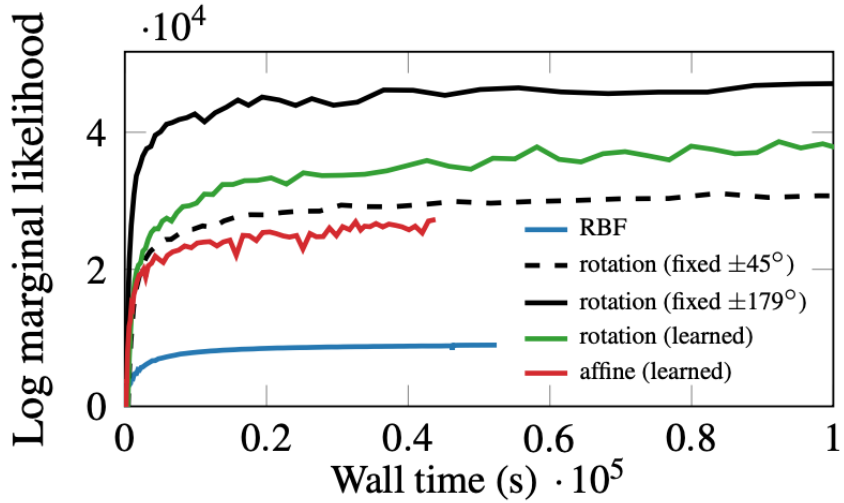


Examples of $t(\mathbf{x};\alpha)$ with $\alpha \sim p(\alpha|\theta)$, describing the **learned invariances**:



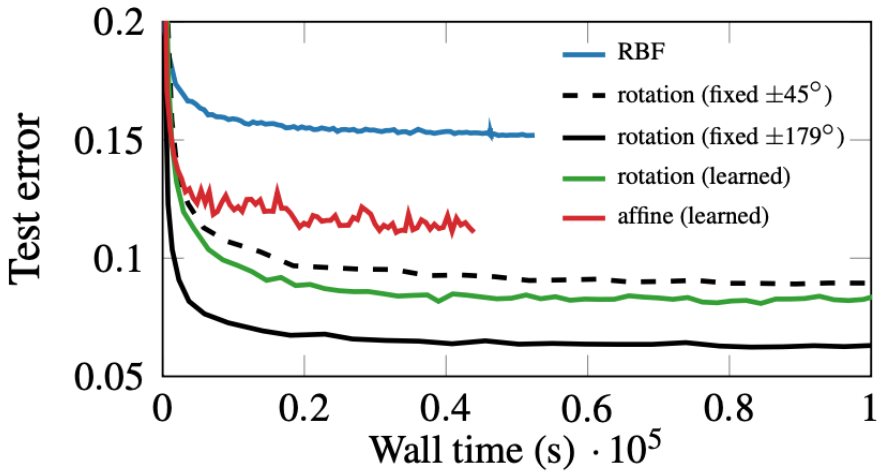
Results: MNIST-rot

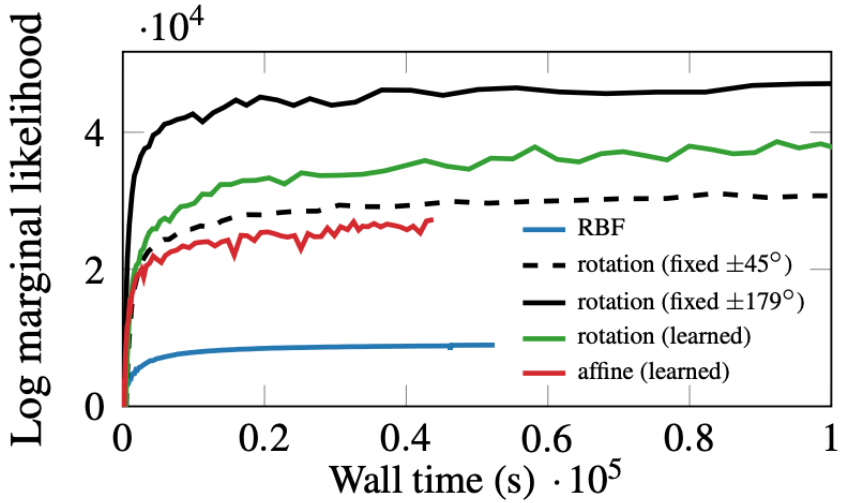




- **Same** model on rotated MNIST dataset recovers **different** invariance (much more rotational invariance)
- No changes needed to the method or architecture.

Results: MNIST-rot





- **Same** model on rotated MNIST dataset recovers **different** invariance (much more rotational invariance)
- No changes needed to the method or architecture.
- Optimisation is difficult when using gradients, but the objective function correctly identifies the solution

Overview

The Promise of Bayesian Model Selection

Accurate Inference in Gaussian Processes

Gaussian Processes in Deep Learning

Ensembles and Architecture Search

Conclusion

The easiest paper I ever wrote

There were quite a few untested moving parts in the "Learning Invariances" paper. We needed:

The easiest paper I ever wrote

There were quite a few untested moving parts in the "Learning Invariances" paper. We needed:

- "Reparameterisation trick"-friendly distributions on image transformations.

The easiest paper I ever wrote

There were quite a few untested moving parts in the "Learning Invariances" paper. We needed:

- "Reparameterisation trick"-friendly distributions on image transformations.
- A *differentiable* implementation of image transformations and resampling.

The easiest paper I ever wrote

There were quite a few untested moving parts in the "Learning Invariances" paper. We needed:

- "Reparameterisation trick"-friendly distributions on image transformations.
- A *differentiable* implementation of image transformations and resampling.
- Stochastic evaluation of the kernel.

The easiest paper I ever wrote

There were quite a few untested moving parts in the "Learning Invariances" paper. We needed:

- "Reparameterisation trick"-friendly distributions on image transformations.
- A *differentiable* implementation of image transformations and resampling.
- Stochastic evaluation of the kernel.
- Unbiased estimate of ELBO that only needs unbiased estimates of the kernel.

The easiest paper I ever wrote

There were quite a few untested moving parts in the "Learning Invariances" paper. We needed:

- "Reparameterisation trick"-friendly distributions on image transformations.
- A *differentiable* implementation of image transformations and resampling.
- Stochastic evaluation of the kernel.
- Unbiased estimate of ELBO that only needs unbiased estimates of the kernel.

The method worked on the first run of the first implementation!

The easiest paper I ever wrote

"Learning invariance" was such an easy paper to write because

we had strong, working theory!

VI in GPs is very mature. In 2018 I could rely on[1]:

- Every (hyper)parameter having a clear guideline on how to set it
- Accurate, reliable approximation of the marginal likelihood
- Optimisation that converges

[1] This is not to say that training these models is easy! But their behaviour is predicted by well-understood theory.

Trouble in Paradise

So are GPs easy?

2

Trouble in Paradise

So are GPs easy? **NO!** They are a pain to train!

2

Trouble in Paradise

So are GPs easy? **NO!** They are a pain to train!

- Cholesky errors make you lose your results

2

Trouble in Paradise

So are GPs easy? **NO!** They are a pain to train!

- Cholesky errors make you lose your results
- How many inducing points to use? (train many models)

2

Trouble in Paradise

So are GPs easy? **NO!** They are a pain to train!

- Cholesky errors make you lose your results
- How many inducing points to use? (train many models)
- Where to place inducing points?

2

Trouble in Paradise

So are GPs easy? **NO!** They are a pain to train!

- Cholesky errors make you lose your results
- How many inducing points to use? (train many models)
- Where to place inducing points?

2

Trouble in Paradise

So are GPs easy? **NO!** They are a pain to train!

- Cholesky errors make you lose your results
- How many inducing points to use? (train many models)
- Where to place inducing points?

GP researchers had systematic solutions to these problems which worked quite well, but they were hard to automate.

2

Trouble in Paradise

So are GPs easy? **NO!** They are a pain to train!

- Cholesky errors make you lose your results
- How many inducing points to use? (train many models)
- Where to place inducing points?

GP researchers had systematic solutions to these problems which worked quite well, but they were hard to automate.

This is a **real problem**:

- GPs should be **the** baseline for (Bayesian) deep learning

2

Trouble in Paradise

So are GPs easy? **NO!** They are a pain to train!

- Cholesky errors make you lose your results
- How many inducing points to use? (train many models)
- Where to place inducing points?

GP researchers had systematic solutions to these problems which worked quite well, but they were hard to automate.

This is a **real problem**:

- GPs should be **the** baseline for (Bayesian) deep learning
- Many BDL papers can be beaten by a well-trained GP on UCI[2]

[2] And, even worse, on MNIST. A SqExp GP easily gets 98%.

Trouble in Paradise

So are GPs easy? **NO!** They are a pain to train!

- Cholesky errors make you lose your results
- How many inducing points to use? (train many models)
- Where to place inducing points?

GP researchers had systematic solutions to these problems which worked quite well, but they were hard to automate.

This is a **real problem**:

- GPs should be **the** baseline for (Bayesian) deep learning
- Many BDL papers can be beaten by a well-trained GP on UCI[2]
- This really holds back progress!

[2] And, even worse, on MNIST. A SqExp GP easily gets 98%.

Trouble in Paradise

So are GPs easy? **NO!** They are a pain to train!

- Cholesky errors make you lose your results
- How many inducing points to use? (train many models)
- Where to place inducing points?

GP researchers had systematic solutions to these problems which worked quite well, but they were hard to automate.

This is a **real problem**:

- GPs should be **the** baseline for (Bayesian) deep learning
- Many BDL papers can be beaten by a well-trained GP on UCI[2]
- This really holds back progress!
- How much can we blame people for not running a difficult method?

[2] And, even worse, on MNIST. A SqExp GP easily gets 98%.

Theory Gives Solutions

In "*Convergence of Sparse Variational Inference in Gaussian Processes Regression*" (Burt et al., 2020) we

- discussed an automatic inducing point initialisation scheme
- proved that it would give *arbitrarily accurate* results as $N \to \infty$
- proved that the asymptotic complexity was reasonable $O(N(\log N)^{2D}(\log\log N)^2))$ for SqExp, barely above linear[3]

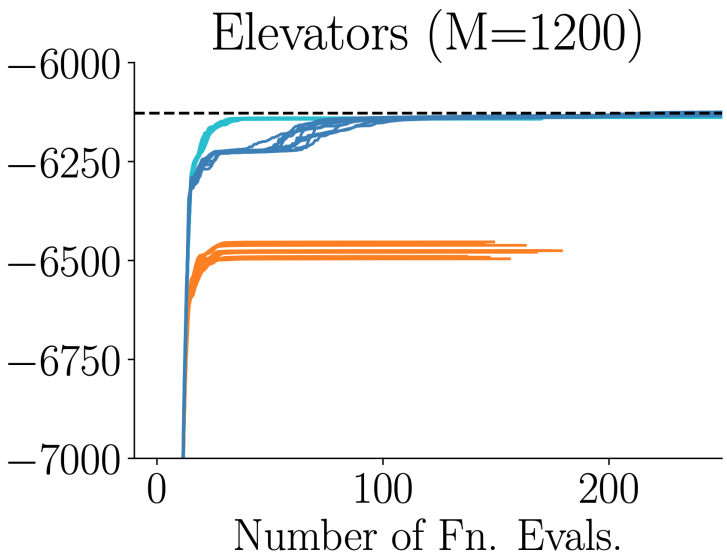
[3] Recall that $O(N(\log N)^D) = O(N^{1+\epsilon})$ for any $D \in \mathbb{N}$ and $\epsilon > 0$, and that D is fixed in our problem.

Theory Gives Solutions

In "*Convergence of Sparse Variational Inference in Gaussian Processes Regression*" (Burt et al., 2020) we

- discussed an automatic inducing point initialisation scheme
- proved that it would give *arbitrarily accurate* results as $N \to \infty$
- proved that the asymptotic complexity was reasonable $O(N(\log N)^{2D}(\log\log N)^2))$ for SqExp, barely above linear[3]

Practical implications for the Titsias (2009) method:



[3] Recall that $O(N(\log N)^D) = O(N^{1+\epsilon})$ for any $D \in \mathbb{N}$ and $\epsilon > 0$, and that D is fixed in our problem.

How solved is efficient GP inference?

We have a *proof* that inducing points are *arbitrarily exact* at *reasonable cost* as $N \to \infty$. Is GP inference solved?

How solved is efficient GP inference?

We have a *proof* that inducing points are *arbitrarily exact* at *reasonable cost* as $N \to \infty$. Is GP inference solved? **No!**

- Given a dataset of fixed size, required number of inducing points may still be too large.

How solved is efficient GP inference?

We have a *proof* that inducing points are *arbitrarily exact* at *reasonable cost* as $N \to \infty$. Is GP inference solved? **No!**

- Given a dataset of fixed size, required number of inducing points may still be too large.
- Inducing points may not give the best speed-accuracy trade-off.

How solved is efficient GP inference?

We have a *proof* that inducing points are *arbitrarily exact* at *reasonable cost* as $N \to \infty$. Is GP inference solved? **No!**

- Given a dataset of fixed size, required number of inducing points may still be too large.
- Inducing points may not give the best speed-accuracy trade-off.
- We need to make running this **super easy** for people.

How solved is efficient GP inference?

We have a *proof* that inducing points are *arbitrarily exact* at *reasonable cost* as $N \to \infty$. Is GP inference solved? **No!**

- Given a dataset of fixed size, required number of inducing points may still be too large.
- Inducing points may not give the best speed-accuracy trade-off.
- We need to make running this **super easy** for people.

How solved is efficient GP inference?

We have a *proof* that inducing points are *arbitrarily exact* at *reasonable cost* as $N \to \infty$. Is GP inference solved? **No!**

- Given a dataset of fixed size, required number of inducing points may still be too large.
- Inducing points may not give the best speed-accuracy trade-off.
- We need to make running this **super easy** for people.

Currently, little impact without good software!

How solved is efficient GP inference?

We have a *proof* that inducing points are *arbitrarily exact* at *reasonable cost* as $N \to \infty$. Is GP inference solved? **No!**

- Given a dataset of fixed size, required number of inducing points may still be too large.
- Inducing points may not give the best speed-accuracy trade-off.
- We need to make running this **super easy** for people.

Currently, little impact without good software!

- The credit you get for good software is non-linear

How solved is efficient GP inference?

We have a *proof* that inducing points are *arbitrarily exact* at *reasonable cost* as $N \to \infty$. Is GP inference solved? **No!**

- Given a dataset of fixed size, required number of inducing points may still be too large.
- Inducing points may not give the best speed-accuracy trade-off.
- We need to make running this **super easy** for people.

Currently, little impact without good software!

- The credit you get for good software is non-linear
- Very little initially, until it becomes generally useful

The hardest paper I ever wrote

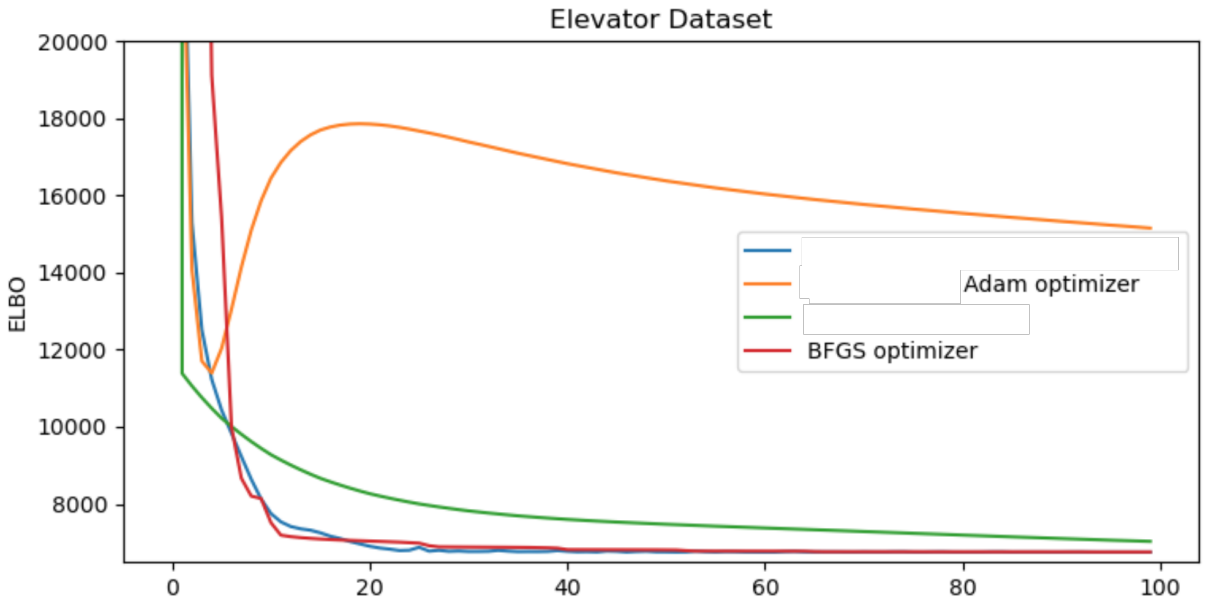
Variational Gaussian Process Models without Matrix Inverses

Mark van der Wilk MARK@PROWLER.IO
ST John ST@PROWLER.IO
Artem Artemev ARTEM@PROWLER.IO
James Hensman JAMES@PROWLER.IO
PROWLER.io

- Had idea in 2018, presented at AABI 2020, still not published.
- Very exciting: Remove bottleneck of matrix inverse.
- Can compute bound in $O(M^2)$ instead of $O(M^3)$.
- Takes far too many iterations to optimise!

Stochastic Optimisation in GPs is really bad!

Open problem: Fast stochastic optimisation in GPs



Courtesy of Maëlhann Rozé, MEng student

- Adam is a terrible stochastic optimiser for GPs!
- Stochastic optimisation is supposed to be faster than full batch!
- We really need to solve this problem if we want to make GPs usable with deep learning.

Conjugate Gradients

Alternative approach to inducing points. Tries to find $\mathbf{K}^{-1}\mathbf{v}$ by solving

$$\underset{\mathbf{x}}{\operatorname{argmin}} \frac{1}{2}\mathbf{x}^\top \mathbf{K}\mathbf{x} - \mathbf{x}^\top \mathbf{b} \tag{8}$$

- **Conjugate Gradients** gives iterative solution that is exact in the limit.
- May give better speed-accuracy trade-off than inducing points.
- Has given genuinely impressive results[4]:

[4] The results and the scale are genuinely impressive. However I disagree that they can be called exact.

Conjugate Gradients

Alternative approach to inducing points. Tries to find $\mathbf{K}^{-1}\mathbf{v}$ by solving

$$\underset{\mathbf{x}}{\operatorname{argmin}} \frac{1}{2}\mathbf{x}^\top \mathbf{K}\mathbf{x} - \mathbf{x}^\top \mathbf{b} \tag{8}$$

- **Conjugate Gradients** gives iterative solution that is exact in the limit.
- May give better speed-accuracy trade-off than inducing points.
- Has given genuinely impressive results[4]:

Exact Gaussian Processes on a Million Data Points

Ke Alexander Wang[1*] **Geoff Pleiss**[1*] **Jacob R. Gardner**[2]
Stephen Tyree[3] **Kilian Q. Weinberger**[1] **Andrew Gordon Wilson**[1,4]
[1]Cornell University, [2]Uber AI Labs, [3]NVIDIA, [4]New York University

[4]The results and the scale are genuinely impressive. However I disagree that they can be called exact.

Conjugate Gradients

An exact method must include the exactness of the training procedure.

Conjugate Gradients

An exact method must include the exactness of the training procedure.
Open questions about CG approach of Wang et al. (2019):

Conjugate Gradients

An exact method must include the exactness of the training procedure.
Open questions about CG approach of Wang et al. (2019):

- How is (the convergence of) hyperparameters affected by error in the estimation of the gradients?

Conjugate Gradients

An exact method must include the exactness of the training procedure.
Open questions about CG approach of Wang et al. (2019):

- How is (the convergence of) hyperparameters affected by error in the estimation of the gradients?
  - Wang et al. (2019) directly approximate the gradients with CG. With error introduced, it is not clear whether following them will lead to convergence.

Conjugate Gradients

An exact method must include the exactness of the training procedure.
Open questions about CG approach of Wang et al. (2019):

- How is (the convergence of) hyperparameters affected by error in the estimation of the gradients?
  - Wang et al. (2019) directly approximate the gradients with CG. With error introduced, it is not clear whether following them will lead to convergence.
- What convergence tolerances should be used to obtain good accuracy-speed tradeoff?

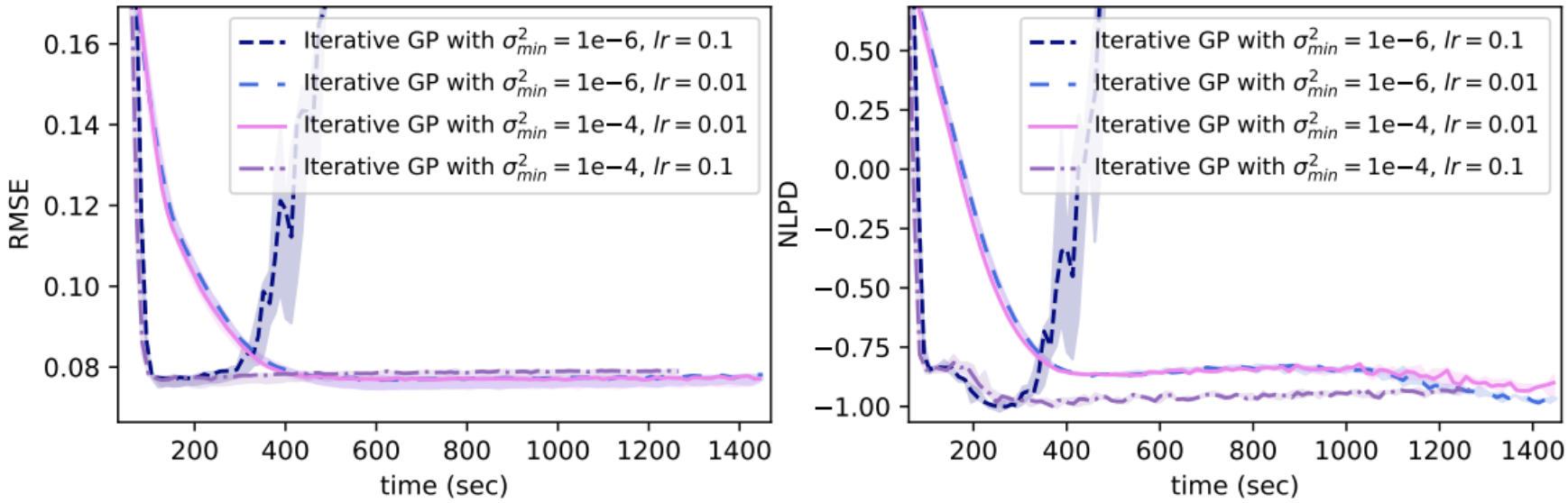
Conjugate Gradients

An exact method must include the exactness of the training procedure.
Open questions about CG approach of Wang et al. (2019):

- How is (the convergence of) hyperparameters affected by error in the estimation of the gradients?
  - Wang et al. (2019) directly approximate the gradients with CG. With error introduced, it is not clear whether following them will lead to convergence.
- What convergence tolerances should be used to obtain good accuracy-speed tradeoff?
  - An "arbitrarily exact" method would require a well-defined rule, a proof, and an asymptotic computational cost analysis, as inducing points currently has (Burt et al., 2019, 2020)

Conjugate Gradients

This has practical consequences, with behaviour that you would not expect from an exact method:



This said, CG is a good idea, and inspired by the results of Wang et al. (2019), we (+ Artem Artemev, David Burt) propose some improvements.

Conjugate Gradient Lower Bound

- VI measures quality of approximation and hyperparameters in a single objective: The ELBO.

Conjugate Gradient Lower Bound

- VI measures quality of approximation and hyperparameters in a single objective: The ELBO.
- Makes it easy to design convergent algorithms: Just optimise the ELBO!

Conjugate Gradient Lower Bound

- VI measures quality of approximation and hyperparameters in a single objective: The ELBO.
- Makes it easy to design convergent algorithms: Just optimise the ELBO!
- Q: How to set parameters? A: Just optimise the ELBO!

Conjugate Gradient Lower Bound

- VI measures quality of approximation and hyperparameters in a single objective: The ELBO.
- Makes it easy to design convergent algorithms: Just optimise the ELBO!
- Q: How to set parameters? A: Just optimise the ELBO!

Conjugate Gradient Lower Bound

- VI measures quality of approximation and hyperparameters in a single objective: The ELBO.
- Makes it easy to design convergent algorithms: Just optimise the ELBO!
- Q: How to set parameters? A: Just optimise the ELBO!

Can we do something similar for Conjugate Gradient methods?

Conjugate Gradient Lower Bound

- VI measures quality of approximation and hyperparameters in a single objective: The ELBO.
- Makes it easy to design convergent algorithms: Just optimise the ELBO!
- Q: How to set parameters? A: Just optimise the ELBO!

Can we do something similar for Conjugate Gradient methods? **Yes!**

Conjugate Gradient Lower Bound

- VI measures quality of approximation and hyperparameters in a single objective: The ELBO.
- Makes it easy to design convergent algorithms: Just optimise the ELBO!
- Q: How to set parameters? A: Just optimise the ELBO!

Can we do something similar for Conjugate Gradient methods? **Yes!**

We develop the **Conjugate Gradient Lower Bound** (CGLB).

Conjugate Gradient Lower Bound

- VI measures quality of approximation and hyperparameters in a single objective: The ELBO.
- Makes it easy to design convergent algorithms: Just optimise the ELBO!
- Q: How to set parameters? A: Just optimise the ELBO!

Can we do something similar for Conjugate Gradient methods? **Yes!**

We develop the **Conjugate Gradient Lower Bound** (CGLB).

- The partial solution $\mathbf{v}$ to the CG optimisation problem to find $\mathbf{K}^{-1}\mathbf{y}$ is unified with the hyperparameter objective
  $\implies$ easier to guarantee convergence

$$\boldsymbol{\theta}^*, \mathbf{v}^* = \text{argmax}_{\boldsymbol{\theta},\mathbf{v}}\, L(\boldsymbol{\theta}, \mathbf{v}), \text{ with } \mathbf{v}^* = \text{argmax}_{\mathbf{v}}\, L(\boldsymbol{\theta}, \mathbf{v}) = \mathbf{K}^{-1}\mathbf{y}\,, \forall \boldsymbol{\theta}.$$

Conjugate Gradient Lower Bound

- VI measures quality of approximation and hyperparameters in a single objective: The ELBO.
- Makes it easy to design convergent algorithms: Just optimise the ELBO!
- Q: How to set parameters? A: Just optimise the ELBO!

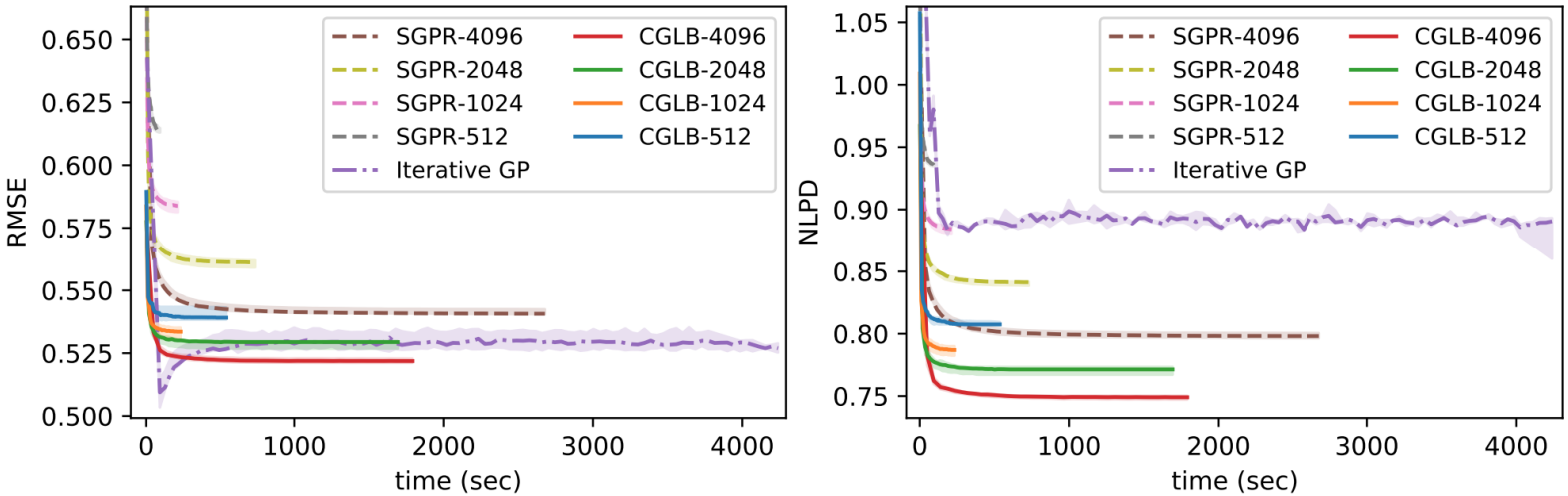
Can we do something similar for Conjugate Gradient methods? **Yes!**

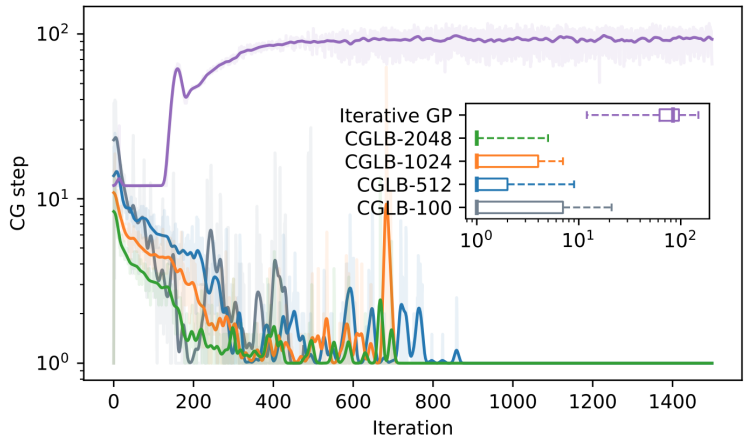
We develop the **Conjugate Gradient Lower Bound** (CGLB).

- The partial solution $\mathbf{v}$ to the CG optimisation problem to find $\mathbf{K}^{-1}\mathbf{y}$ is unified with the hyperparameter objective
  $\implies$ easier to guarantee convergence
- Additional upper bound to automatically determine number of CG iterations.

$\boldsymbol{\theta}^*, \mathbf{v}^* = \text{argmax}_{\boldsymbol{\theta},\mathbf{v}} L(\boldsymbol{\theta}, \mathbf{v})$, with $\mathbf{v}^* = \text{argmax}_{\mathbf{v}} L(\boldsymbol{\theta}, \mathbf{v}) = \mathbf{K}^{-1}\mathbf{y}\,, \forall \boldsymbol{\theta}$.

Conjugate Gradient Lower Bound





- Fewer iterations of CG $\implies$ faster.
- More guarantees for optimisation $\implies$ better performance.

Overview

The Promise of Bayesian Model Selection

Accurate Inference in Gaussian Processes

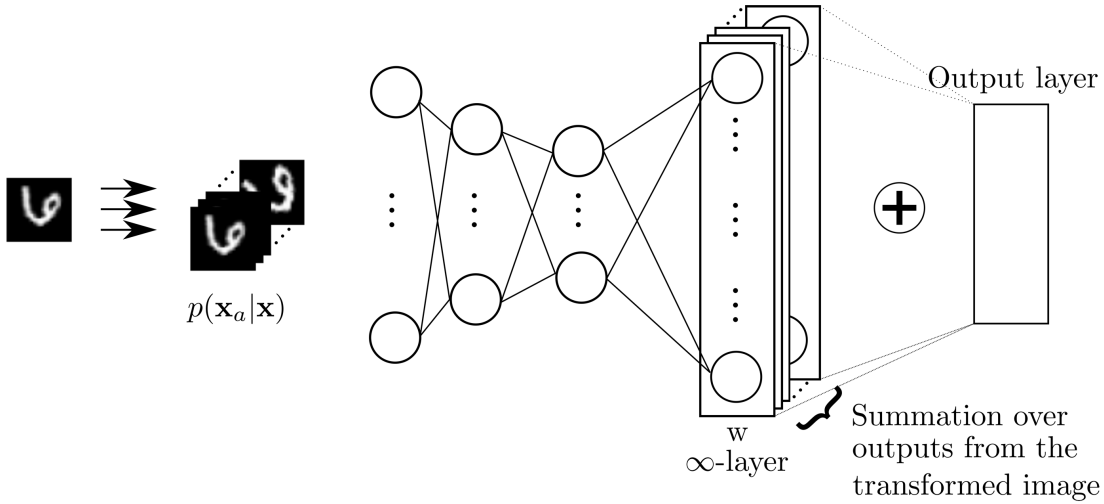
Gaussian Processes in Deep Learning

Ensembles and Architecture Search

Conclusion

Learning Invariances in DNNs

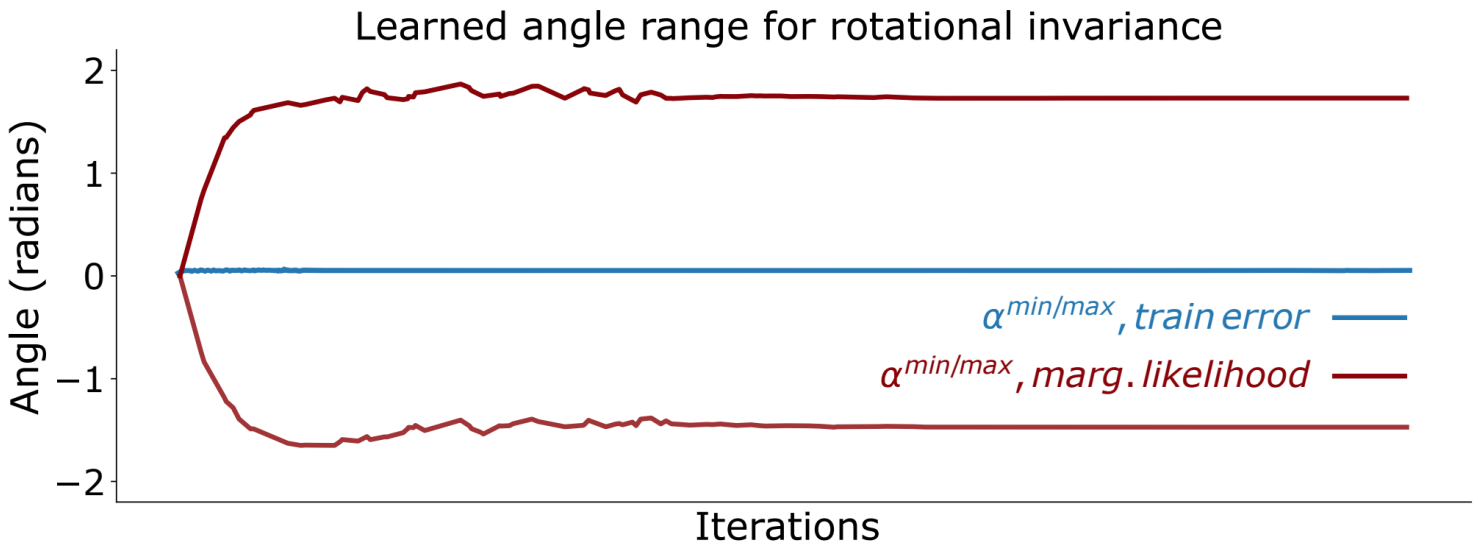
Find invariances through backprop for a Deep Neural Network, by only computing the marginal likelihood for the last layer.



- Can find invariance / data augmentation by backpropagation!
- Difficult training procedure...
- Pola Schwöbel (DTU), Martin Jørgensen (DTU), MvdW.

Learning Invariances in DNNs

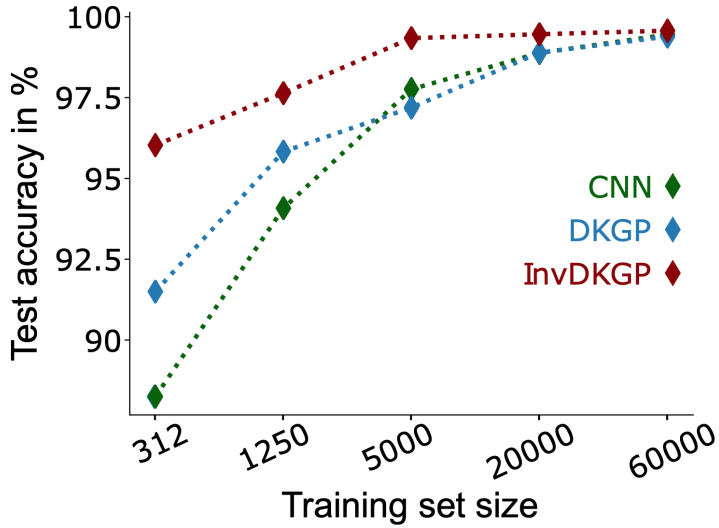
Find invariances through backprop for a Deep Neural Network, by only computing the marginal likelihood for the last layer.



- Can find invariance / data augmentation by backpropagation!
- Difficult training procedure...
- Pola Schwöbel (DTU), Martin Jørgensen (DTU), MvdW.

Learning Invariances in DNNs

Find invariances through backprop for a Deep Neural Network, by only computing the marginal likelihood for the last layer.



- Can find invariance / data augmentation by backpropagation!
- Difficult training procedure...
- Pola Schwöbel (DTU), Martin Jørgensen (DTU), MvdW.

Successful Deep Bayesian Model Selection

Currently, there are few usable marginal likelihood estimates for DNNs, although there is some exciting recent work (Ober and Aitchison, 2020; Immer et al., 2021).

Successful Deep Bayesian Model Selection

Currently, there are few usable marginal likelihood estimates for DNNs, although there is some exciting recent work (Ober and Aitchison, 2020; Immer et al., 2021).
There is one deep model where variational bounds work for Bayesian model selection...

Successful Deep Bayesian Model Selection

Currently, there are few usable marginal likelihood estimates for DNNs, although there is some exciting recent work (Ober and Aitchison, 2020; Immer et al., 2021).
There is one deep model where variational bounds work for Bayesian model selection...

- **Deep Gaussian Processes** (Damianou and Lawrence, 2013)!

Successful Deep Bayesian Model Selection

Currently, there are few usable marginal likelihood estimates for DNNs, although there is some exciting recent work (Ober and Aitchison, 2020; Immer et al., 2021).
There is one deep model where variational bounds work for Bayesian model selection...

- **Deep Gaussian Processes** (Damianou and Lawrence, 2013)!
- **Theory seems to work**: Derive VI bound, get uncertainty & hyperparameters.

Successful Deep Bayesian Model Selection

Currently, there are few usable marginal likelihood estimates for DNNs, although there is some exciting recent work (Ober and Aitchison, 2020; Immer et al., 2021).
There is one deep model where variational bounds work for Bayesian model selection...

- **Deep Gaussian Processes** (Damianou and Lawrence, 2013)!
- **Theory seems to work**: Derive VI bound, get uncertainty & hyperparameters.
- Deep Convolutional GPs (Blomqvist et al., 2020; Dutordoir et al., 2020) are **improving significantly**.

Successful Deep Bayesian Model Selection

Currently, there are few usable marginal likelihood estimates for DNNs, although there is some exciting recent work (Ober and Aitchison, 2020; Immer et al., 2021).
There is one deep model where variational bounds work for Bayesian model selection...

- **Deep Gaussian Processes** (Damianou and Lawrence, 2013)!
- **Theory seems to work**: Derive VI bound, get uncertainty & hyperparameters.
- Deep Convolutional GPs (Blomqvist et al., 2020; Dutordoir et al., 2020) are **improving significantly**.
- New methods make DGPs easier to use, and look remarkably similar to DNNs. **Convergence?**

Overview

Ensembles and Posteriors

For linear models, ensembles and posteriors can be identical (Matthews et al., 2017):

$$\mathbf{w}^{(k)} \overset{\text{iid}}{\sim} \mathcal{N}(0, \boldsymbol{I}) \tag{9}$$

$$\mathbf{w}_p^{(k)} := \text{GradDesc}(\mathbf{w}^{(k)}, ||\mathbf{y} - \Phi(X)\mathbf{w}||^2) \tag{10}$$

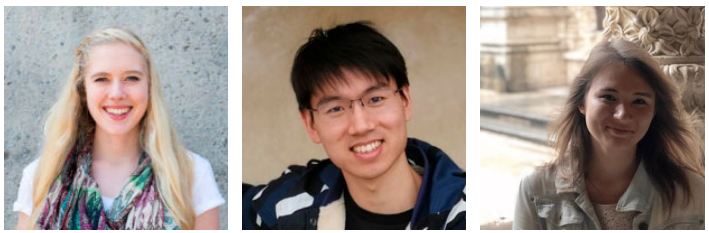
$$\implies \mathbf{w}_p^{(k)} \overset{\text{iid}}{\sim} p(\mathbf{w}|\mathbf{y}) \tag{11}$$

where $p(y_n|\mathbf{w}) = \mathcal{N}\left(y_n; \boldsymbol{\phi}(\mathbf{x}_n)^\top \mathbf{w}, \sigma^2\right)$, with $\sigma^2 \to 0$.

Can optimisation do Bayesian Model Selection?

$$\log p(\mathbf{y}) = \sum_{i=1}^{n} \log p(y_i \mid \mathbf{y}_{<i}) = \sum_i \log \mathbb{E}_{p(\mathbf{w}|\mathbf{y}_{<i})}[p(y_i|\mathbf{w})]$$
$$\geqslant \sum_i \mathbb{E}_{p(\mathbf{w}|\mathbf{y}_{<i})}[\log p(y_i|\mathbf{w})]$$

For linear models the answer is an unambiguous **yes**:



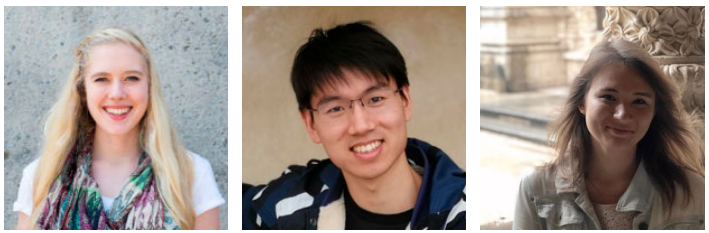
Clare Lyle, Lisa Schut, Binxin Ru, Yarin Gal, MvdW

Can optimisation do Bayesian Model Selection?

$$\log p(\mathbf{y}) = \sum_{i=1}^{n} \log p(y_i \mid \mathbf{y}_{<i}) = \sum_i \log \mathbb{E}_{p(\mathbf{w}|\mathbf{y}_{<i})}[p(y_i|\mathbf{w})]$$
$$\geqslant \sum_i \mathbb{E}_{p(\mathbf{w}|\mathbf{y}_{<i})}[\log p(y_i|\mathbf{w})]$$

For linear models the answer is an unambiguous **yes**:

- exact posterior samples can be produced to minimizing an unregularized loss (Matthews et al., 2017),



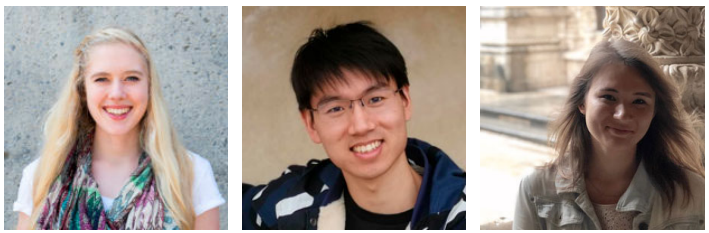
Clare Lyle, Lisa Schut, Binxin Ru, Yarin Gal, MvdW

Can optimisation do Bayesian Model Selection?

$$\log p(\mathbf{y}) = \sum_{i=1}^{n} \log p(y_i \mid \mathbf{y}_{<i}) = \sum_i \log \mathbb{E}_{p(\mathbf{w}|\mathbf{y}_{<i})}[p(y_i|\mathbf{w})]$$
$$\geqslant \sum_i \mathbb{E}_{p(\mathbf{w}|\mathbf{y}_{<i})}[\log p(y_i|\mathbf{w})]$$

For linear models the answer is an unambiguous **yes**:

- exact posterior samples can be produced to minimizing an unregularized loss (Matthews et al., 2017),
- so we can get a lower bound to the marginal likelihood by summing training losses.



Clare Lyle, Lisa Schut, Binxin Ru, Yarin Gal, MvdW

Can optimisation do Bayesian Model Selection?

$$\log p(\mathbf{y}) = \sum_{i=1}^{n} \log p(y_i \mid \mathbf{y}_{<i}) = \sum_i \log \mathbb{E}_{p(\mathbf{w}|\mathbf{y}_{<i})}[p(y_i|\mathbf{w})]$$
$$\geqslant \sum_i \mathbb{E}_{p(\mathbf{w}|\mathbf{y}_{<i})}[\log p(y_i|\mathbf{w})]$$

For linear models the answer is an unambiguous **yes**:

- exact posterior samples can be produced to minimizing an unregularized loss (Matthews et al., 2017),
- so we can get a lower bound to the marginal likelihood by summing training losses.

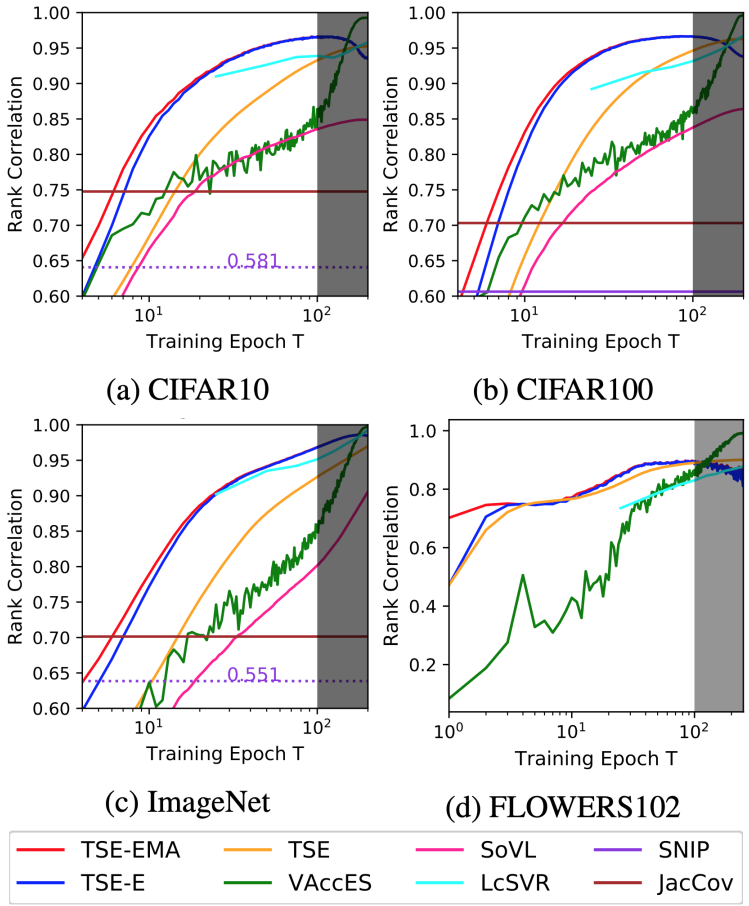
- Bayesian Perspective on Training Speed and Model Selection



Clare Lyle, Lisa Schut, Binxin Ru, Yarin Gal, MvdW

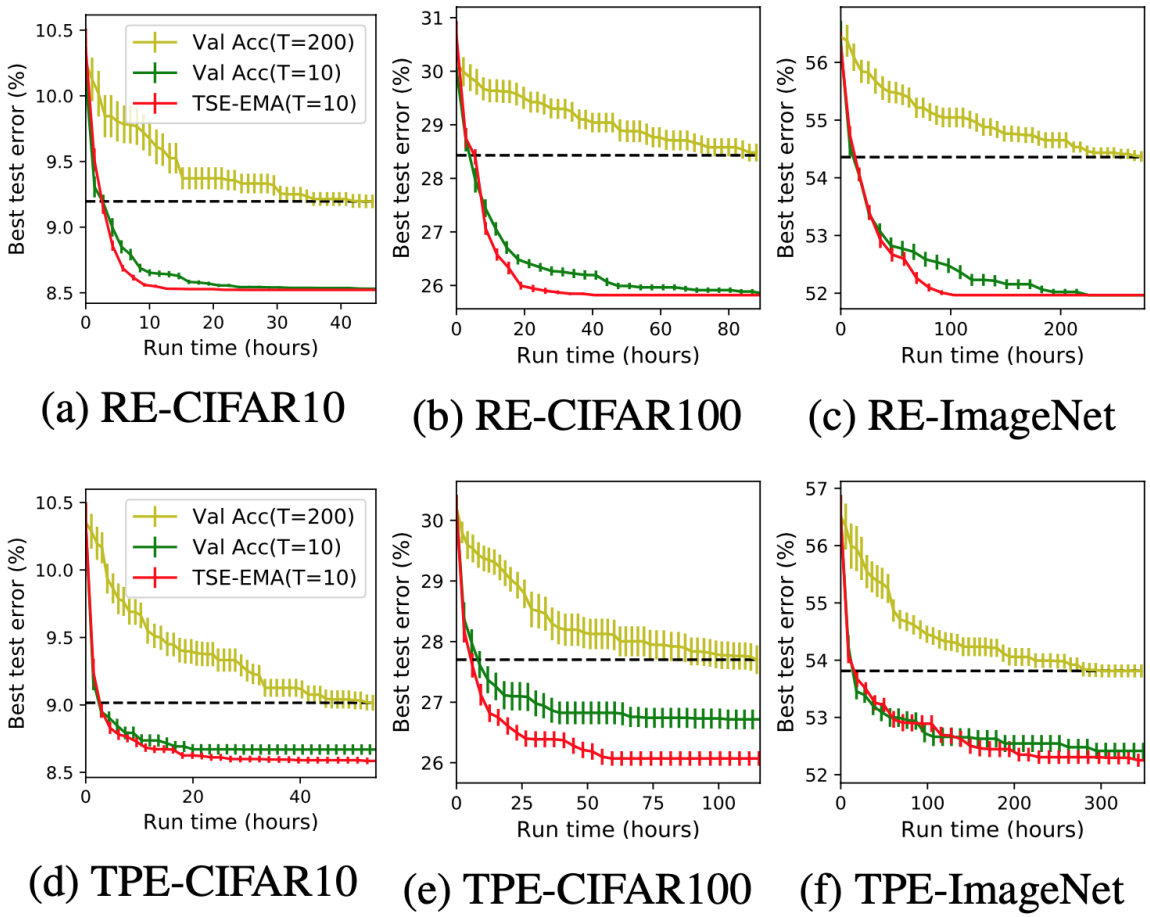
Neural Architecture Search

Inspired by this, we investigated whether training speed could predict testing accuracy of a network.



Neural Architecture Search

Inspired by this, we investigated whether training speed could predict testing accuracy of a network.



Overview

The Promise of Bayesian Model Selection

Accurate Inference in Gaussian Processes

Gaussian Processes in Deep Learning

Ensembles and Architecture Search

Conclusion

Summary

Papers we discussed, which I was involved in:

- **Learning Invariance using the Marginal Likelihood**
  v.d.Wilk et al. (2018), NeurIPS
- **Convergence of Sparse Variational Inference in GPR**
  Burt et al. (2019, 2020), ICML, JMLR
- **Software for automatic & robust training of GPs**
  Work in Progress
- **Variational GP Models without Matrix Inverses**
  van der Wilk et al. (2020), AABI, Work in Progress
- **Tighter Bounds on the LML of GPR Using CG**
  Artemev et al. (2021), Preprint
- **Last Layer Marginal Likelihood for Invariance Learning**
  Work in Progress (arxiv soon)
- **Bayesian Image Classification with Deep Convolutional GPs**
  Dutordoir et al. (2020), AISTATS
- **A Bayesian Perspective on Training Speed and Model Selection**
  Lyle et al. (2020), NeurIPS
- **An Efficient Performance Estimator for Neural Architecture Search**
  Ru et al. (2020), Preprint

Conclusion & Ways Forward

Conclusion:

- Bayesian model selection could provide great benefits in DNNs.
- Interesting model selection (invariances) is already possible in GPs.
- Model selection using a marginal likelihood approx already works in deep GPs.

Ways forward:

- GP inference needs to be faster per iteration, and converge faster.
- Alternatively: More accurate marginal likelihood estimates of NNs.
- Deep GPs and DNNs may converge!

Alternative approaches

Two approaches based on back-propagating through a validation set:

- Meta-learning (Zhou et al., 2020)
- Implicit function theorem (Lorraine et al., 2020)
- Straightforward regularization (Benton et al., 2020)

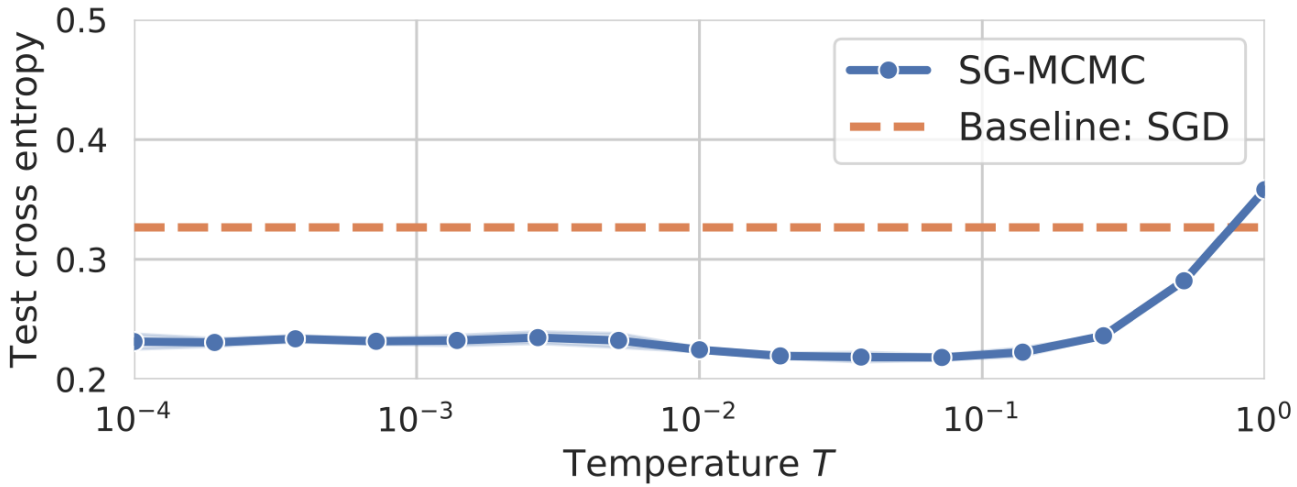
What is going to be best? Only research will tell!

Bonus Content / Directors Cut

Example

TODO: Animations of overfitting, correct fitting, and periodic

Cold Posteriors



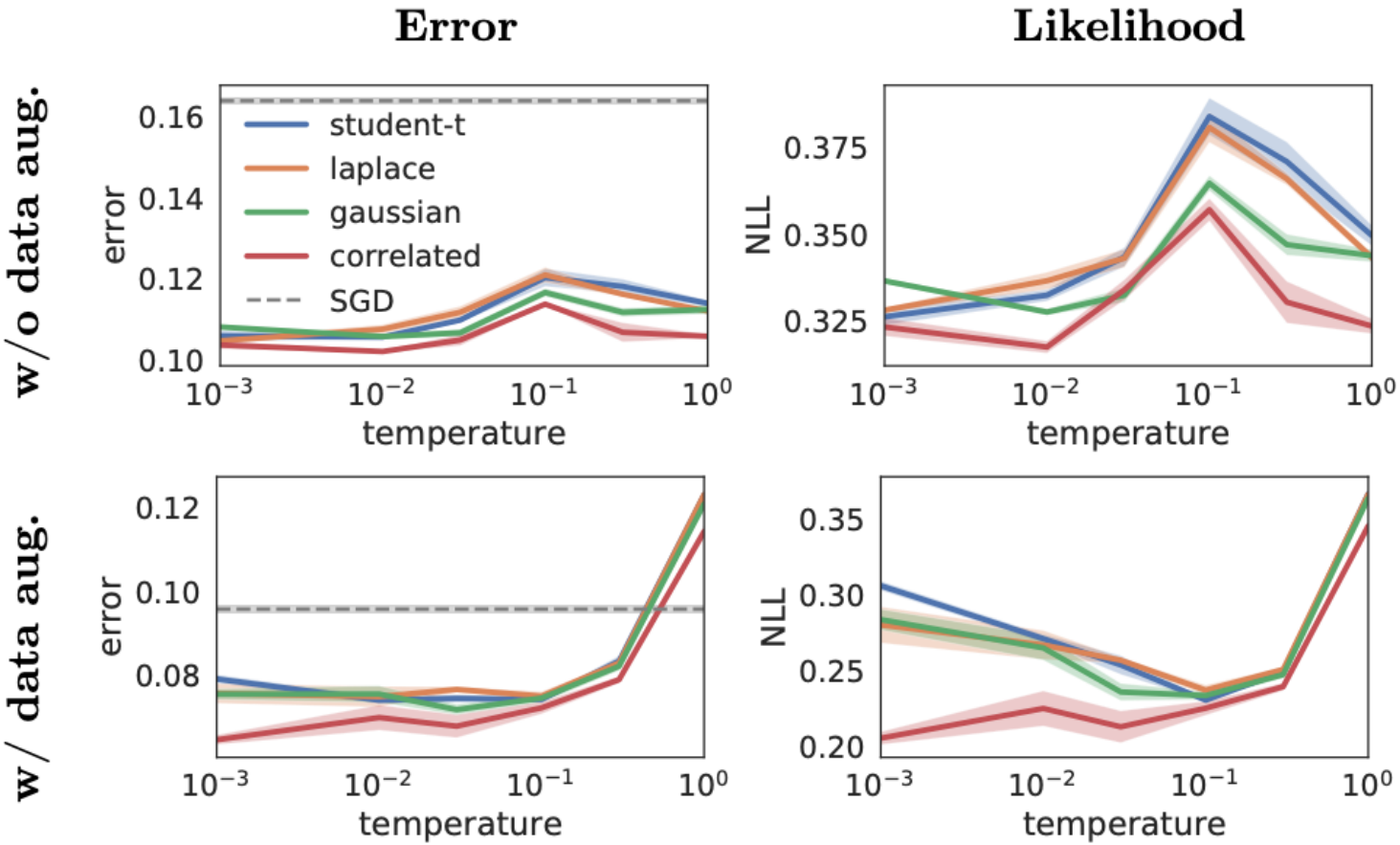
From Wenzel et al. (2020)

$$p_T(\boldsymbol{\theta}|\mathbf{y}) \propto p(\boldsymbol{\theta}|\mathbf{y})^{1/T} \tag{12}$$

- Posterior performs worse than point estimate!
- Bayes is sensitive to the prior as well!
- Does the prior make things worse?

Weight Priors

Investigate **different weight priors** in neural networks:



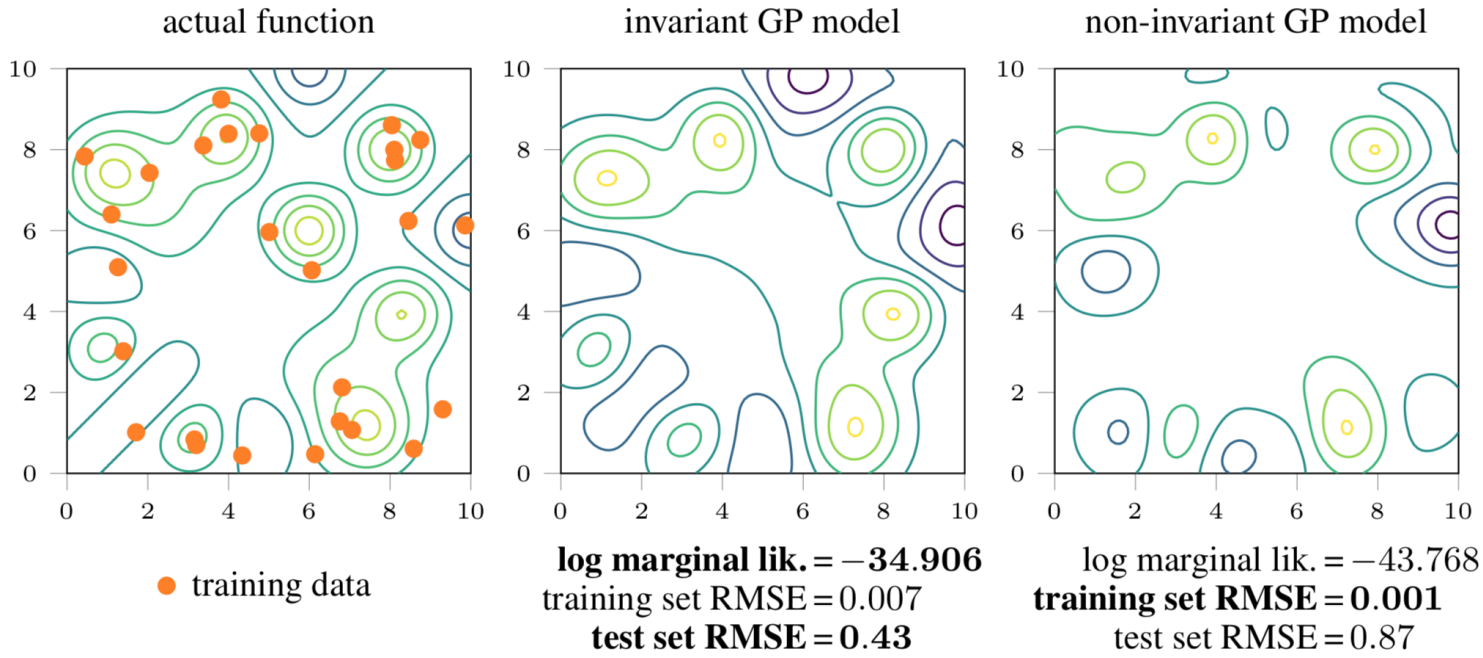
Bayesian Neural Network Priors Revisited

Fortuin*, Garriga-Alonso*, Wenzel, Rätsch, Turner, vdW†, Aitchison†

Why can't we just use training loss?

For the same reason as why we need cross-validation:

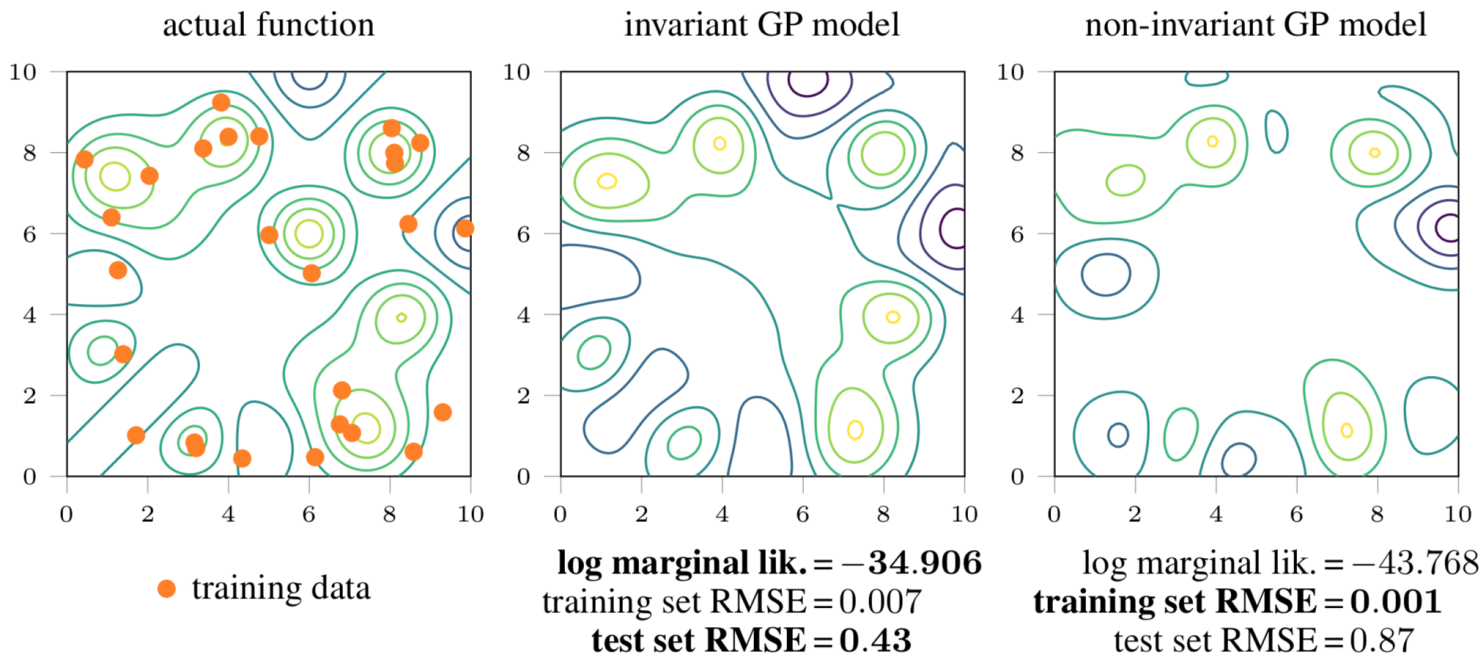
- The training loss is minimised with the most flexible model
- Inductive biases are **constraints**



Why can't we just use training loss?

For the same reason as why we need cross-validation:

- The training loss is minimised with the most flexible model
- Inductive biases are **constraints**



Log marginal likelihood measures **generalisation**:

$$\log p(\mathbf{y} \mid \theta) = \log p(y_1|\theta) + \log p(y_2|\theta, y_1) + \log p(y_3|\theta, \{y_i\}_{i=1}^2) \ldots$$

(It's also related to cross-validation (Fong and Holmes, 2020).)

References I

Artemev, A., Burt, D. R., and van der Wilk, M. (2021). Tighter bounds on the log marginal likelihood of gaussian process regression using conjugate gradients.

Benton, G., Finzi, M., Izmailov, P., and Wilson, A. G. (2020). Learning invariances in neural networks.

Berger, J. O. (1985). Statistical decision theory and Bayesian analysis. Springer.

Blomqvist, K., Kaski, S., and Heinonen, M. (2020). Deep convolutional gaussian processes. In Brefeld, U., Fromont, E., Hotho, A., Knobbe, A., Maathuis, M., and Robardet, C., editors, Machine Learning and Knowledge Discovery in Databases, pages 582–597, Cham. Springer International Publishing.

References II

Burt, D., Rasmussen, C. E., and Van Der Wilk, M. (2019). Rates of convergence for sparse variational Gaussian process regression. In Chaudhuri, K. and Salakhutdinov, R., editors, Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pages 862–871. PMLR.

Burt, D. R., Rasmussen, C. E., and van der Wilk, M. (2020). Convergence of sparse variational inference in gaussian processes regression. Journal of Machine Learning Research, 21(131):1–63.

Damianou, A. C. and Lawrence, N. D. (2013). Deep Gaussian processes. In Proceedings of the 16th International Conference on Artificial Intelligence and Statistics, pages 207–215.

Dutordoir, V., van der Wilk, M., Artemev, A., and Hensman, J. (2020). Bayesian image classification with deep convolutional gaussian processes. In Chiappa, S. and Calandra, R., editors, Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, volume 108 of Proceedings of Machine Learning Research, pages 1529–1539, Online. PMLR.

References III

Fong, E. and Holmes, C. C. (2020). On the marginal likelihood and cross-validation. Biometrika, 107(2):489–496.

Immer, A., Bauer, M., Fortuin, V., Rätsch, G., and Khan, M. E. (2021). Scalable marginal likelihood estimation for model selection in deep learning.

Lorraine, J., Vicol, P., and Duvenaud, D. (2020). Optimizing millions of hyperparameters by implicit differentiation. In Chiappa, S. and Calandra, R., editors, Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, volume 108 of Proceedings of Machine Learning Research, pages 1540–1552, Online. PMLR.

Lyle, C., Schut, L., Ru, B., Gal, Y., and van der Wilk, M. (2020). A bayesian perspective on training speed and model selection.

Matthews, A. G. d. G., Hron, J., Turner, R. E., and Ghahramani, Z. (2017). Sample-then-optimize posterior sampling for bayesian linear models. In NeurIPS Workshop on Advances in Approximate Bayesian Inference.

Ober, S. W. and Aitchison, L. (2020). Global inducing point variational posteriors for bayesian neural networks and deep gaussian processes.

References IV

Ru, B., Lyle, C., Schut, L., van der Wilk, M., and Gal, Y. (2020). Revisiting the train loss: an efficient performance estimator for neural architecture search.

Titsias, M. K. (2009). Variational learning of inducing variables in sparse Gaussian processes. In Proceedings of the 12th International Conference on Artificial Intelligence and Statistics, pages 567–574.

van der Wilk, M., John, S., Artemev, A., and Hensman, J. (2020). Variational gaussian process models without matrix inverses. In Zhang, C., Ruiz, F., Bui, T., Dieng, A. B., and Liang, D., editors, Proceedings of The 2nd Symposium on Advances in Approximate Bayesian Inference, volume 118 of Proceedings of Machine Learning Research, pages 1–9. PMLR.

v.d.Wilk, M., Bauer, M., John, S., and Hensman, J. (2018). Learning invariances using the marginal likelihood. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, Advances in Neural Information Processing Systems 31, pages 9938–9948. Curran Associates, Inc.

References V

Wang, K., Pleiss, G., Gardner, J., Tyree, S., Weinberger, K. Q., and Wilson, A. G. (2019). Exact gaussian processes on a million data points. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc.

Wenzel, F., Roth, K., Veeling, B., Swiatkowski, J., Tran, L., Mandt, S., Snoek, J., Salimans, T., Jenatton, R., and Nowozin, S. (2020). How good is the Bayes posterior in deep neural networks really? In III, H. D. and Singh, A., editors, Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 10248–10259, Virtual. PMLR.

Zhou, A., Knowles, T., and Finn, C. (2020). Meta-learning symmetries by reparameterization.