Imperial College London

# Uncertainty in Deep Models
## using Gaussian Processes

**Mark van der Wilk**

Department of Computing
Imperial College London

m.vdwilk@imperial.ac.uk

March 10, 2020

- This talk is loosely based on the paper **Bayesian Image Classification with Deep Convolutional Gaussian Processes**, Vincent Dutordoir, Mark van der Wilk, Artem Artemev, James Hensman; AISTATS 2020.

# Overview

# Uncertainty: a matter of life or death


TESLA MODEL S MODEL 3 MODEL X MODEL Y ROADSTER

Blog Videos

A Tragic Loss

The Tesla Team • 30 June 2016

We learned yesterday evening that NHTSA is opening a preliminary evaluation into the performance of Autopilot during a recent fatal crash that occurred in a Model S. This is the first known fatality in just over 130 million miles where Autopilot was activated. Among all vehicles in the US, there is a fatality every 94 million miles. Worldwide, there is a fatality approximately every 60 million miles. It is important to emphasize that the NHTSA action is simply a preliminary evaluation to determine whether the system worked according to expectations.

Following our standard practice, Tesla informed NHTSA about the incident immediately after it occurred. What we know is that the vehicle was on a divided highway with Autopilot engaged when a tractor trailer drove across the highway perpendicular to the Model S. Neither Autopilot nor the driver noticed the white side of the tractor trailer against a brightly lit sky, so the brake was not applied. The high ride height of the trailer combined with its positioning across the road and the extremely rare circumstances of



Deep learning applied in the wild, but what would you do

- in a previously unseen situation, or ambiguous stimulus?
- if you were 10% sure there was an obstruction?

# Automatic machine learning

Current learning procedure:

1. Obtain a large dataset
2. Design data augmentations
3. Train multiple models with different hyperparameters (layers, topology, ...)
4. Cross-validate and deploy model with best performance

# Automatic machine learning

Current learning procedure:

1. Obtain a large dataset
2. Design data augmentations
3. Train multiple models with different hyperparameters (layers, topology, ...)
4. Cross-validate and deploy model with best performance

Can we

- **automatically** pick hyperparameters and data augmentation?
- **update** model based on new observations?

Goals:

Goals:

1. Good uncertainty

Goals:

1. Good uncertainty
2. Automatic model selection

Goals:

1. Good uncertainty
2. Automatic model selection

# Related problems

in the Bayesian framework

# Overview

Goals

Bayesian Deep Learning

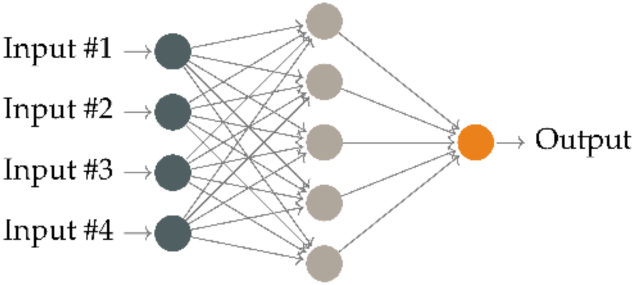
Gaussian Processes

Deep Gaussian Processes

Application & results

Conclusions

# Neural networks are basis function models

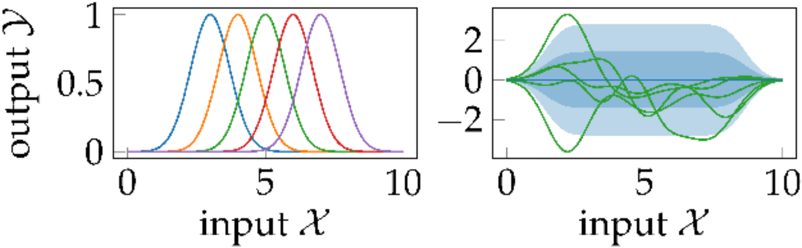



$$f(\mathbf{x}) = \sum_{b=1}^{B} w_b \phi_b(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x})$$

$$\phi_b(\mathbf{x}) = \sigma\left(\sum_{d=1}^{D} \tilde{w}_d x_d\right) = \sigma(\tilde{\mathbf{w}}^\top \mathbf{x})$$

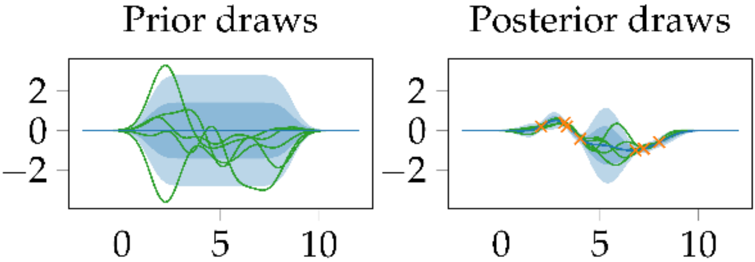
# Bayesian Neural Networks are a prior over functions

Placing priors on **w** gives us a distribution over functions:




$$f(\mathbf{x}) = \sum_{b=1}^{B} w_b \phi_b(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}), \qquad \mathbf{w} \sim \mathcal{N}\left(\mathbf{w}; 0, \sigma_w^2 I\right).$$

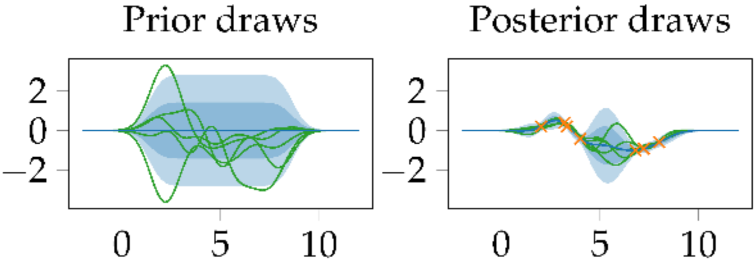
# Bayesian advantages




Using the prior, we can obtain the **posterior** to quantify **uncertainty**:

$$p(\mathbf{w}|\mathbf{y},\theta) = \frac{\prod_n p(y_n|\mathbf{w},\theta)p(\mathbf{w}|\theta)}{p(\mathbf{y}|\theta)}$$

# Bayesian advantages




Using the prior, we can obtain the **posterior** to quantify **uncertainty**:

$$p(\mathbf{w}|\mathbf{y},\theta) = \frac{\prod_n p(y_n|\mathbf{w},\theta)p(\mathbf{w}|\theta)}{p(\mathbf{y}|\theta)}$$

Using the **marginal likelihood** we can find hyperparameters (properties of the prior):

$$p(\theta\,|\,\mathbf{y}) = \frac{\prod_n p(\mathbf{y}\,|\,\theta)p(\theta)}{p(\mathbf{y})}$$

# Variational Inference

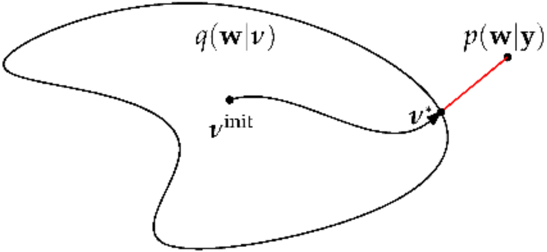



Figure adopted from Blei et al.'s NIPS-2016 tutorial

- Find approximation of a probability distribution (e.g., posterior) by optimization:
  1. Define a (parametrized) family of approximating distributions $q_\nu$
  2. Define KL[approx||posterior] to be measure of similarity
  3. Optimise measure w.r.t. variational parameters $\nu$
- Inference ⏩ Optimization

# Variational Inference in Bayesian Neural Networks

Variational inference is most commonly used for approximate inference in BNNs:

$$q(\mathbf{w}) = \underset{q(\mathbf{w})\in\mathcal{Q}}{\operatorname{argmin}} \mathrm{KL}[q(\mathbf{w})||p(\mathbf{w}\,|\,\mathbf{y},\theta)]$$

$$\log p(\mathbf{y}\,|\,\theta) - \mathrm{KL}[q(\mathbf{w})||p(\mathbf{w}\,|\,\mathbf{y},\theta)] = \mathrm{ELBO} = \mathcal{L}$$

ELBO becomes:

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{w})}[\log p(\mathbf{y}\,|\,\mathbf{w},\theta)] - \mathrm{KL}[q(\mathbf{w})||p(\mathbf{w})]$$

$$\text{with e.g. } q(\mathbf{w}) = \prod_{p=1}^{P} \mathcal{N}\left(w_i; \mu_i, \sigma_i^2\right)$$

E.g. Blundell et al. *Weight Uncertainty in Neural Networks* [2015]

# Is variational inference working?

From Blundell et al. *Weight Uncertainty in Neural Networks* [2015]:

> cross-validation where possible. Empirically we found optimising the parameters of a prior $P(\mathbf{w})$ (by taking derivatives of (1)) to not be useful, and yield worse results.

# Is variational inference working?

From Blundell et al. *Weight Uncertainty in Neural Networks* [2015]:

> cross-validation where possible. Empirically we found optimising the parameters of a prior $P(\mathbf{w})$ (by taking derivatives of (1)) to not be useful, and yield worse results.

- ELBOs not tight enough for model comparison

# Is variational inference working?

From Blundell et al. *Weight Uncertainty in Neural Networks* [2015]:

> cross-validation where possible. Empirically we found optimising the parameters of a prior $P(\mathbf{w})$ (by taking derivatives of (1)) to not be useful, and yield worse results.

- ELBOs not tight enough for model comparison
- Observation: Bounds are so loose that they prefer a noise model over fitting the data (i.e. variance of $\mathbb{V}_{p(\mathbf{w}\,|\,\theta_{\text{opt}})} = 0$)

$$\mathcal{L} + \text{KL}[q(\mathbf{w})||p(\mathbf{w}\,|\,\mathbf{y})] = \log p(\mathbf{y}\,|\,\theta)$$

$$\mathcal{L}(\mathbf{v}_{\text{opt}}, \theta_{\text{opt}}) \gg \mathcal{L}(\mathbf{v}_{\text{good}}, \theta_{\text{good}})$$

$$\implies \text{KL}[q(\mathbf{w})||p(\mathbf{w}\,|\,\mathbf{y}, \theta)] = \text{large!}$$

# Problems

Bayesian deep learning using Variational Inference

# Problems

Bayesian deep learning using Variational Inference

1. does not give good estimates of the marginal likelihood (so no model selection!)

# Problems

Bayesian deep learning using Variational Inference

1. does not give good estimates of the marginal likelihood (so no model selection!)
2. has an inaccurate approximation to the true posterior

# Problems

Bayesian deep learning using Variational Inference

1. does not give good estimates of the marginal likelihood (so no model selection!)
2. has an inaccurate approximation to the true posterior

We could be doing a lot better!

# Overview

# Gaussian Processes

A Gaussian process is a **distribution over functions** with Gaussian marginals. Its properties are defined by the **kernel function** $k(\mathbf{x}, \mathbf{x}')$:

$$p(f(\mathbf{x}_1), f(\mathbf{x}_2), f(\mathbf{x}_3), \dots) = p(f(X)) = \mathcal{N}(f(X); 0, \boldsymbol{K})$$
$$[\boldsymbol{K}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$$

# Gaussian Processes

A Gaussian process is a **distribution over functions** with Gaussian marginals. Its properties are defined by the **kernel function** $k(\mathbf{x}, \mathbf{x}')$:

$$p(f(\mathbf{x}_1), f(\mathbf{x}_2), f(\mathbf{x}_3), \dots) = p(f(X)) = \mathcal{N}(f(X); 0, \boldsymbol{K})$$
$$[\boldsymbol{K}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$$

- Behaves as a basis function model

# Gaussian Processes

A Gaussian process is a **distribution over functions** with Gaussian marginals. Its properties are defined by the **kernel function** $k(\mathbf{x}, \mathbf{x}')$:

$$p(f(\mathbf{x}_1), f(\mathbf{x}_2), f(\mathbf{x}_3), \dots) = p(f(X)) = \mathcal{N}(f(X); 0, \boldsymbol{K})$$
$$[\boldsymbol{K}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$$

- Behaves as a basis function model
- Can have infinite basis functions

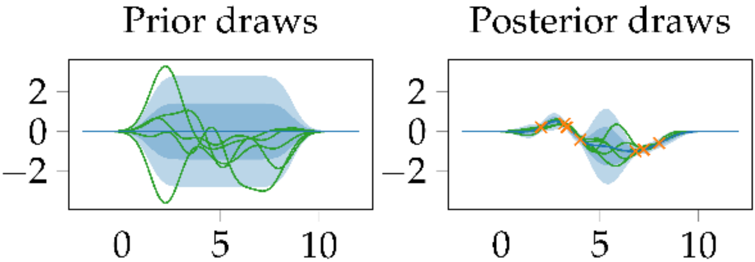
# Gaussian Processes

A Gaussian process is a **distribution over functions** with Gaussian marginals. Its properties are defined by the **kernel function** $k(\mathbf{x}, \mathbf{x}')$:

$$p(f(\mathbf{x}_1), f(\mathbf{x}_2), f(\mathbf{x}_3), \dots) = p(f(X)) = \mathcal{N}(f(X); 0, \boldsymbol{K})$$
$$[\boldsymbol{K}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$$

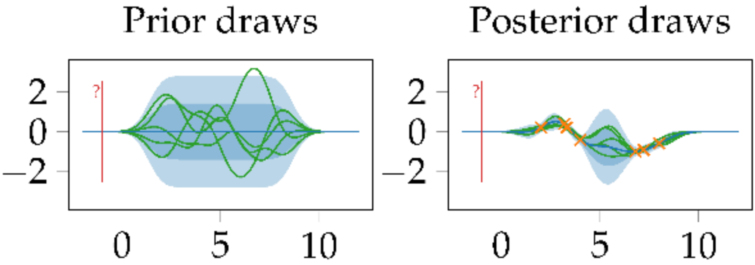
- Behaves as a basis function model
- Can have infinite basis functions
- Posteriors can be represented accurately

# Finite basis functions



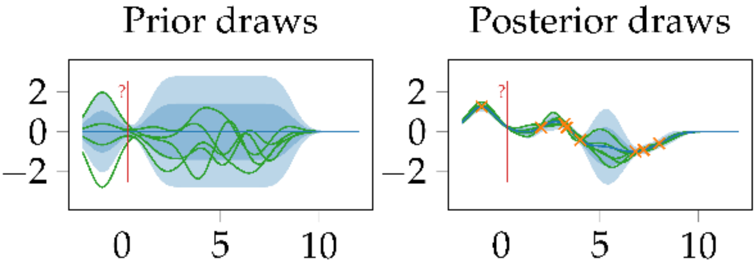

- Should we be so certain far from the data?
- How many basis functions?

# Finite basis functions



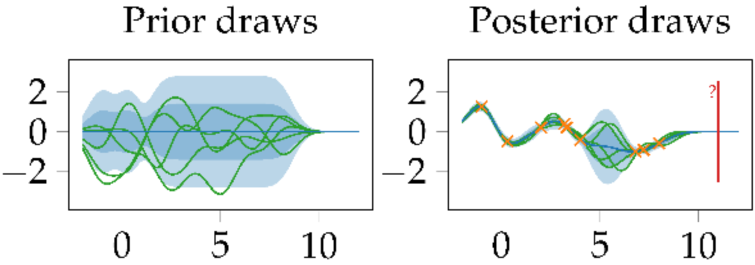

- Should we be so certain far from the data?
- How many basis functions?

# Finite basis functions



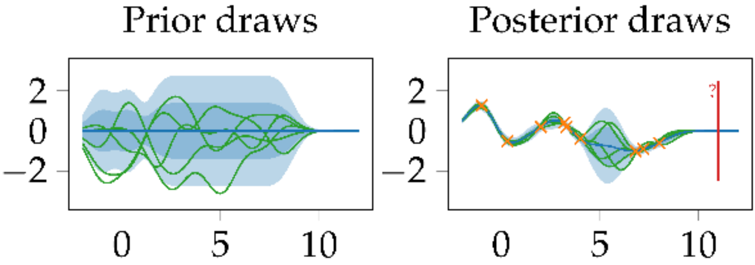

- Should we be so certain far from the data?
- How many basis functions?

# Finite basis functions




- Should we be so certain far from the data?
- How many basis functions?

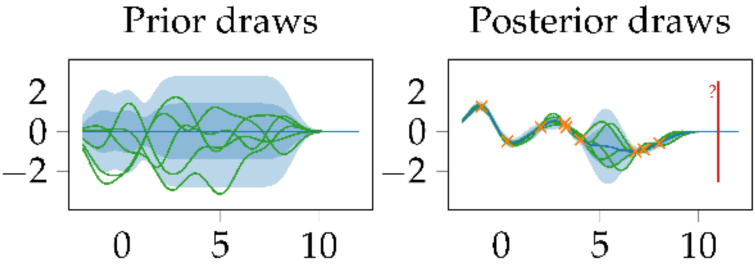
# Finite basis functions




- Should we be so certain far from the data?
- How many basis functions?

Solution: Use large number of basis functions

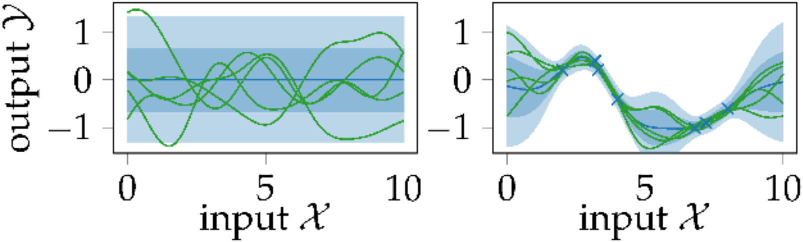
# Finite basis functions




- Should we be so certain far from the data?
- How many basis functions?

Solution: Use an infinite number of basis functions?

# Infinite basis functions




- Should we be so certain far from the data? → **No, and we don't have to be!**
- How may basis functions? → **infinite!**

# Inference in Gaussian Processes

Predictions are made using the posterior:

$$p(f(X^*) \mid \mathbf{y}, \theta) = \int p(f(X^*) \mid f(X), \theta) \frac{\prod_n p(y_n \mid f(\mathbf{x}_n)) p(f(X) \mid \theta)}{p(\mathbf{y} \mid \theta)} \mathrm{d} f(X)$$

# Inference in Gaussian Processes

Predictions are made using the posterior:

$$p(f(X^*) \,|\, \mathbf{y}, \theta) = \int p(f(X^*) \,|\, f(X), \theta) \frac{\prod_n p(y_n \,|\, f(\mathbf{x}_n)) p(f(X) \,|\, \theta)}{p(\mathbf{y} \,|\, \theta)} \mathrm{d}f(X)$$

- Prior is computationally costly. Covariance matrix inverse and determinant scale as $O(N^3)$.
- Need approximate inference for non-Gaussian likelihoods.

# Variational Inference for Gaussian Processes

In VI for GPs, we minimise the KL divergence between the approximate posterior over functions $q(f(\cdot))$ and the true posterior over functions $p(f(\cdot) \mid \mathbf{y}, \theta)$:

$$\mathrm{KL}[q(f(\cdot))||p(f(\cdot) \mid \mathbf{y}, \theta)]$$

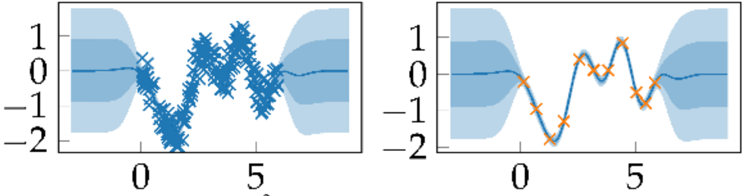
This is well-defined, [Matthews et al. 2016], and leads to a tractable ELBO [Hensman et al. 2013]:

$$\mathcal{L} = \sum_{n=1}^{N} \mathbb{E}_{q(f(\mathbf{x}_n))}[\log p(y_n \mid f(\mathbf{x}_n)] - \mathrm{KL}[q(f(\cdot))||p(f(\cdot))]$$

(We abuse notation of densities over functions to mean the appropriate Gaussian process measures, or a distribution over an arbitrary set of function values.)

# Set of approximate posteriors

Gaussian process prior, but with constrained behaviour at $M$ points

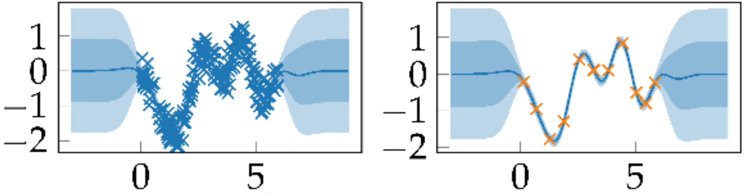



$$q(f(\mathbf{x}_n)) = \int p(f(\mathbf{x}_n) \mid f(Z)) q(f(Z)) \mathrm{d}f(Z)$$

$$\mathcal{L} = \sum_{n=1}^{N} \mathbb{E}_{q(f(\mathbf{x}_n))}[\log p(y_n \mid f(\mathbf{x}_n)] - \mathrm{KL}[q(f(Z)) || p(f(Z))]$$

# Set of approximate posteriors

Gaussian process prior, but with constrained behaviour at $M$ points




$$q(f(\mathbf{x}_n)) = \int p(f(\mathbf{x}_n) \mid f(Z)) q(f(Z)) \mathrm{d}f(Z)$$

$$\mathcal{L} = \sum_{n=1}^{N} \mathbb{E}_{q(f(\mathbf{x}_n))}[\log p(y_n \mid f(\mathbf{x}_n)] - \mathrm{KL}[q(f(Z)) || p(f(Z))]$$

- Computationally efficient [Titsias 2009]
- Can be minibatched [Hensman et al. 2013]
- Works with arbitrary likelihoods [Hensman et al. 2016]
- Can be arbitrarily accurate [Burt et al. 2019]

# Overview

Goals

Bayesian Deep Learning

Gaussian Processes

Deep Gaussian Processes

Application & results

Conclusions

# Gaussian Processes as a Layer

A Gaussian process has nicer properties than a **single layer** neural network, but has **limited performance** in high-dimensional tasks.

# Gaussian Processes as a Layer

A Gaussian process has nicer properties than a **single layer** neural network, but has **limited performance** in high-dimensional tasks.

Can we use a GP as a layer in a deep model?

# Gaussian Processes as a Layer

A Gaussian process has nicer properties than a **single layer** neural network, but has **limited performance** in high-dimensional tasks.

Can we use a GP as a layer in a deep model?

Possible advantages:

- Better uncertainty per layer (infinite basis functions)?
- More accurate inference?

# Gaussian Processes as a Layer

A Gaussian process has nicer properties than a **single layer** neural network, but has **limited performance** in high-dimensional tasks.

Can we use a GP as a layer in a deep model?

Possible advantages:

- Better uncertainty per layer (infinite basis functions)?
- More accurate inference?

[Damianou & Lawrence 2013]

# Deep Gaussian Processes

Define model through

- function composition (like deep NNs),
- Gaussian process priors on each layer.

$$f(\mathbf{x}) = f_L(f_{L-1}(f_{L-2}(\dots f_1(\mathbf{x})\dots))) = (f_L \circ f_{L-1} \circ \dots f_1)(\mathbf{x})$$
$$f_\ell(\cdot) \sim \mathcal{GP}(0, k_\ell(\cdot, \cdot'))$$

# Deep Gaussian Processes

Define model through

- function composition (like deep NNs),
- Gaussian process priors on each layer.

$$f(\mathbf{x}) = f_L(f_{L-1}(f_{L-2}(\dots f_1(\mathbf{x})\dots))) = (f_L \circ f_{L-1} \circ \dots f_1)(\mathbf{x})$$
$$f_\ell(\cdot) \sim \mathcal{GP}(0, k_\ell(\cdot, \cdot'))$$

How do we find the posterior?

$$p(f_1(\cdot), f_2(\cdot), \dots \mid \mathbf{y}) = \frac{\prod_{n=1}^{N} p(y_n \mid f(\mathbf{x}_n), \mathbf{x}_n) \prod_{\ell=1}^{L} p(f_\ell(\cdot) \mid \theta)}{p(\mathbf{y} \mid \theta)} \quad (1)$$

# Variational Inference for Gaussian Processes

We again minimise the KL divergence between the distributions over functions, only we have more now.

$$\mathrm{KL}[q(f_1, \ldots, f_L) \| p(f_1, \ldots, f_L \,|\, \mathbf{y})]$$

$$q(f_1, \ldots, f_L) = \prod_{\ell=1}^{L} q(f_\ell(\cdot))$$

# Variational Inference for Gaussian Processes

We again minimise the KL divergence between the distributions over functions, only we have more now.

$$\mathrm{KL}[q(f_1, \ldots, f_L) \| p(f_1, \ldots, f_L \mid \mathbf{y})]$$

$$q(f_1, \ldots, f_L) = \prod_{\ell=1}^{L} q(f_\ell(\cdot))$$

The ELBO has a similar structure, and can be optimised using Monte Carlo estimates of the expectations:

$$\mathcal{L} = \sum_{n=1}^{N} \mathbb{E}_{q(f_1,\ldots,f_L)}[\log p(y_n \mid (f_L \circ \cdots \circ f_1)(\mathbf{x}_n))] - \sum_{\ell=1}^{L} \mathrm{KL}[q(f_\ell(Z)) \| p(f_\ell(Z))]$$

Monte Carlo estimate only needs to evaluate $f_\ell(\cdot)$ at the output of $f_{\ell-1}(\cdot)$, starting with $f_1(\mathbf{x})$ [Salimbeni & Deisenroth 2017].

# Overview

# Deep Convolutional Gaussian Process

In Dutordoir, v.d. Wilk, Artemev & Hensman [2020], we

- stack Gaussian process layers with convolutional structure [v.d. Wilk 2017],

# Deep Convolutional Gaussian Process

In Dutordoir, v.d. Wilk, Artemev & Hensman [2020], we

- stack Gaussian process layers with convolutional structure [v.d. Wilk 2017],
- introduce modelling capacity compared to Blomqvist et al. [2018],

# Deep Convolutional Gaussian Process

In Dutordoir, v.d. Wilk, Artemev & Hensman [2020], we

- stack Gaussian process layers with convolutional structure [v.d. Wilk 2017],
- introduce modelling capacity compared to Blomqvist et al. [2018],
- apply the straightforward variational inference procedure from Salimbeni & Deisenroth [2017].

# Deep Convolutional Gaussian Process

In Dutordoir, v.d. Wilk, Artemev & Hensman [2020], we

- stack Gaussian process layers with convolutional structure [v.d. Wilk 2017],
- introduce modelling capacity compared to Blomqvist et al. [2018],
- apply the straightforward variational inference procedure from Salimbeni & Deisenroth [2017].

We obtain

- an ELBO that we maximise for **selecting hyperparameters**,
- **competitive performance** on MNIST,
- **better uncertainty estimates** compared to NNs.

# Deep Convolutional Gaussian Processes: Uncertainty


Applying theory simply works!

1. ELBO tight enough for hyperparameter optimisation
2. Evidence supporting KL[approx||posterior] is small

# Deep Convolutional Gaussian Processes: Results

Table 2: DCGP [Blomqvist et al., 2019] (reproduced with our code) and Deep TICK-GP (our method) on MNIST and CIFAR-10.

| | | MNIST | | CIFAR-10 | |
|---|---|---|---|---|---|
| depth | metric | Conv | TICK | Conv | TICK |
| 1 | top-1 error (%) | 1.87 | **1.19** | 41.06 | **37.10** |
| | NLL full | 0.06 | **0.04** | 1.17 | **1.08** |
| | neg. ELBO ($\times 10^3$) | 8.29 | **5.83** | 65.72 | **63.51** |
| 2 | top-1 error (%) | 0.96 | **0.67** | 28.60 | **25.59** |
| | NLL full | 0.04 | **0.02** | 0.84 | **0.75** |
| | neg. ELBO ($\times 10^3$) | 5.37 | **4.25** | 52.81 | **48.31** |
| 3 | top-1 error (%) | 0.93 | **0.64** | 25.33 | **23.83** |
| | NLL full | 0.03 | **0.02** | 0.74 | **0.69** |
| | neg. ELBO ($\times 10^3$) | 5.045 | **4.19** | 49.38 | **47.53** |

# Overview

# Conclusion

How does our approach compare to common Bayesian Deep Learning practice?

# Conclusion

How does our approach compare to common Bayesian Deep Learning practice?

- DGPs are behind in performance, but steadily improving.

# Conclusion

How does our approach compare to common Bayesian Deep Learning practice?

- DGPs are behind in performance, but steadily improving.
- DGPs are slow, but getting faster.

# Conclusion

How does our approach compare to common Bayesian Deep Learning practice?

- DGPs are behind in performance, but steadily improving.
- DGPs are slow, but getting faster.
- BDL starts from current methods that perform well, and try to make inference work.

# Conclusion

How does our approach compare to common Bayesian Deep Learning practice?

- DGPs are behind in performance, but steadily improving.
- DGPs are slow, but getting faster.
- BDL starts from current methods that perform well, and try to make inference work.
- DGPs start from inference that works, and try to make it perform well.

# Future work

- Faster models, so we can train bigger models.

# Future work

- Faster models, so we can train bigger models.
- Bigger models, so we can get better performance.

# Future work

- Faster models, so we can train bigger models.
- Bigger models, so we can get better performance.
- Automatic learning of model structure and invariances.

# Future work

- Faster models, so we can train bigger models.
- Bigger models, so we can get better performance.
- Automatic learning of model structure and invariances.

We recently released a **review paper** on arXiv:

*A Framework for Interdomain and Multioutput Gaussian Processes*

Mark van der Wilk, Vincent Dutordoir, ST John, Artem Artemev, Vincent Adam, James Hensman

**https://arxiv.org/abs/2003.01115**

From theory, past derivations, all the way to implementation.

# Future work

- Faster models, so we can train bigger models.
- Bigger models, so we can get better performance.
- Automatic learning of model structure and invariances.

We recently released a **review paper** on arXiv:

*A Framework for Interdomain and Multioutput Gaussian Processes*

Mark van der Wilk, Vincent Dutordoir, ST John, Artem Artemev, Vincent Adam, James Hensman

**https://arxiv.org/abs/2003.01115**

From theory, past derivations, all the way to implementation.

Thank you!

# References

Key references. See paper for more.


- **Variational Learning of Inducing Variables in Sparse Gaussian Processes**; Michalis K. Titsias; AISTATS (2009).
- **Gaussian Processes for Big Data**; James Hensman, Nicolo Fusi, James D. Hensman; UAI (2013).
- **Scalable Variational Gaussian Process Classification**; James Hensman, Alexander G. de G. Matthews, Zoubin Ghahramani; AISTATS 2014
- **Weight Uncertainty in Neural Networks**; Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, Daan Wierstra; ICML 2015
- **On Sparse Variational Methods and the Kullback-Leibler Divergence between Stochastic Processes**; Alexander G. de G. Matthews, James Hensman, Richard Turner, Zoubin Ghahramani; AISTATS 2016
- **Doubly stochastic variational inference for deep Gaussian processes**; Hugh Salimbeni, Marc Deisenroth; NIPS 2017
- **Convolutional Gaussian Processes**; Mark van der Wilk, Carl E. Rasmussen, James Hensman; NIPS 2017
- **Deep convolutional Gaussian processes**; Kenneth Blomqvist, Samuel Kaski, Markus Heinonen; arXiv
- **Rates of Convergence for Sparse Variational Gaussian Process Regression**; David R. Burt, Carl E. Rasmussen, Mark van der Wilk; ICML 2019