Deep Gaussian Processes

Maurizio Filippone

EURECOM, Sophia Antipolis, France

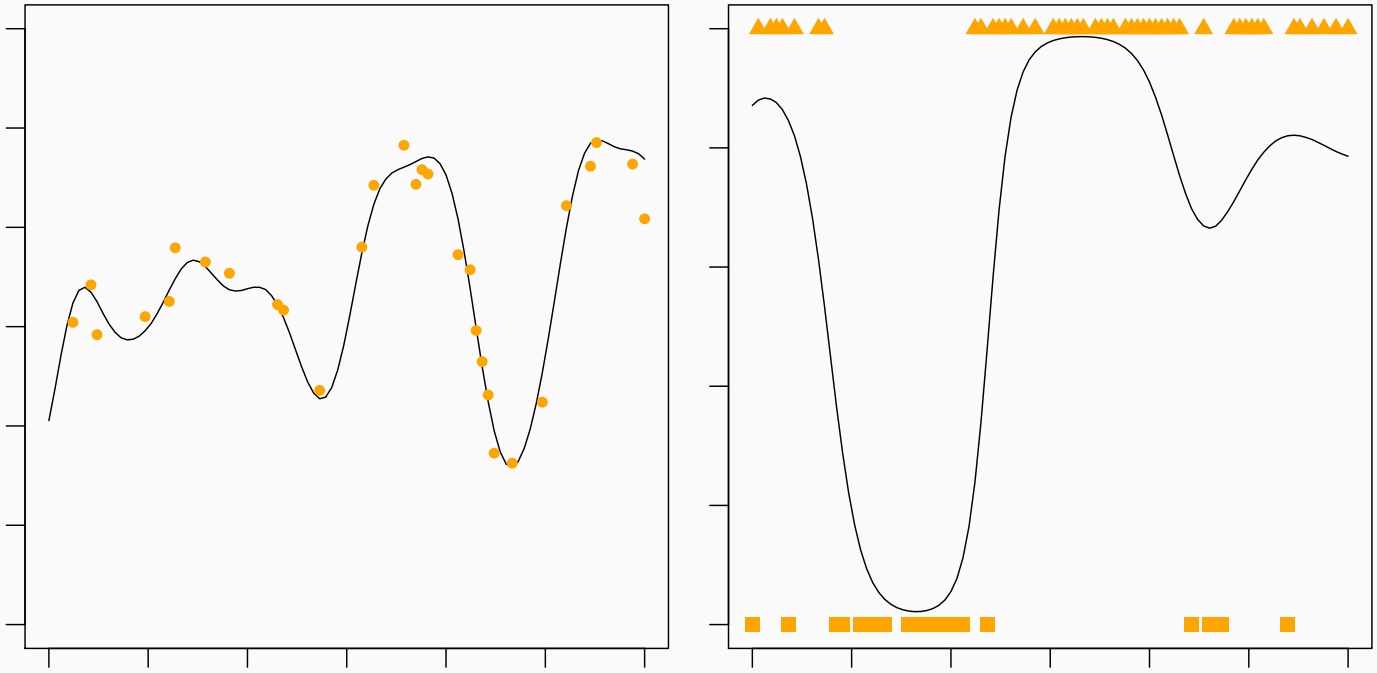
August 23rd, 2019


9
10
9
9
9
9
10
10
9
11
9
8
11
10
9
8
8
11
9
9
5
9
12
6
4
10
11
8
10
21
12
11
8

Introduction

Learning from Data — Function Estimation

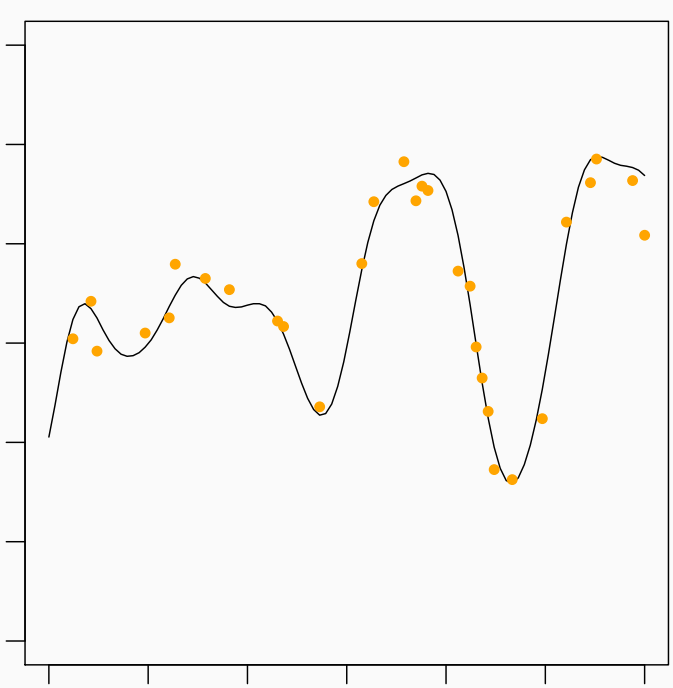
- Take these two examples

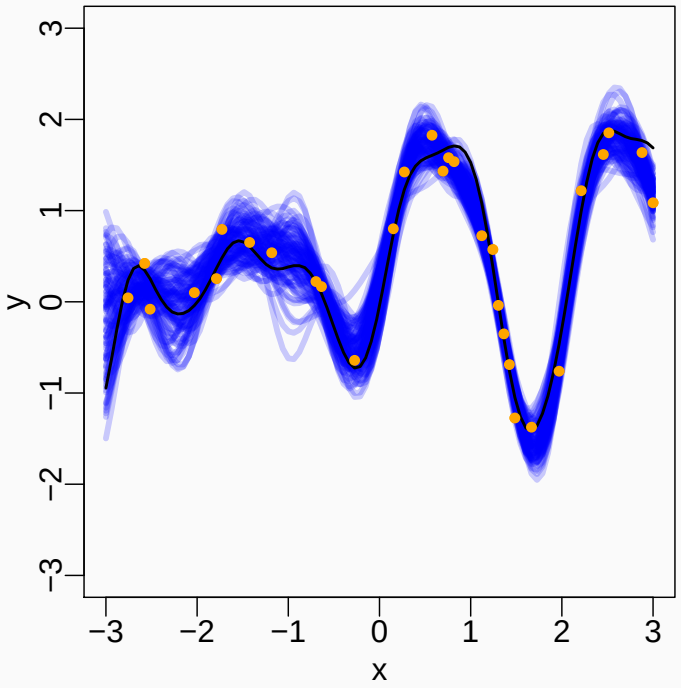


- We are interested in estimating a function $f(\mathbf{x})$ from data
- Most problems in Machine Learning can be cast this way!

What do Bayesian Models Have to Offer?

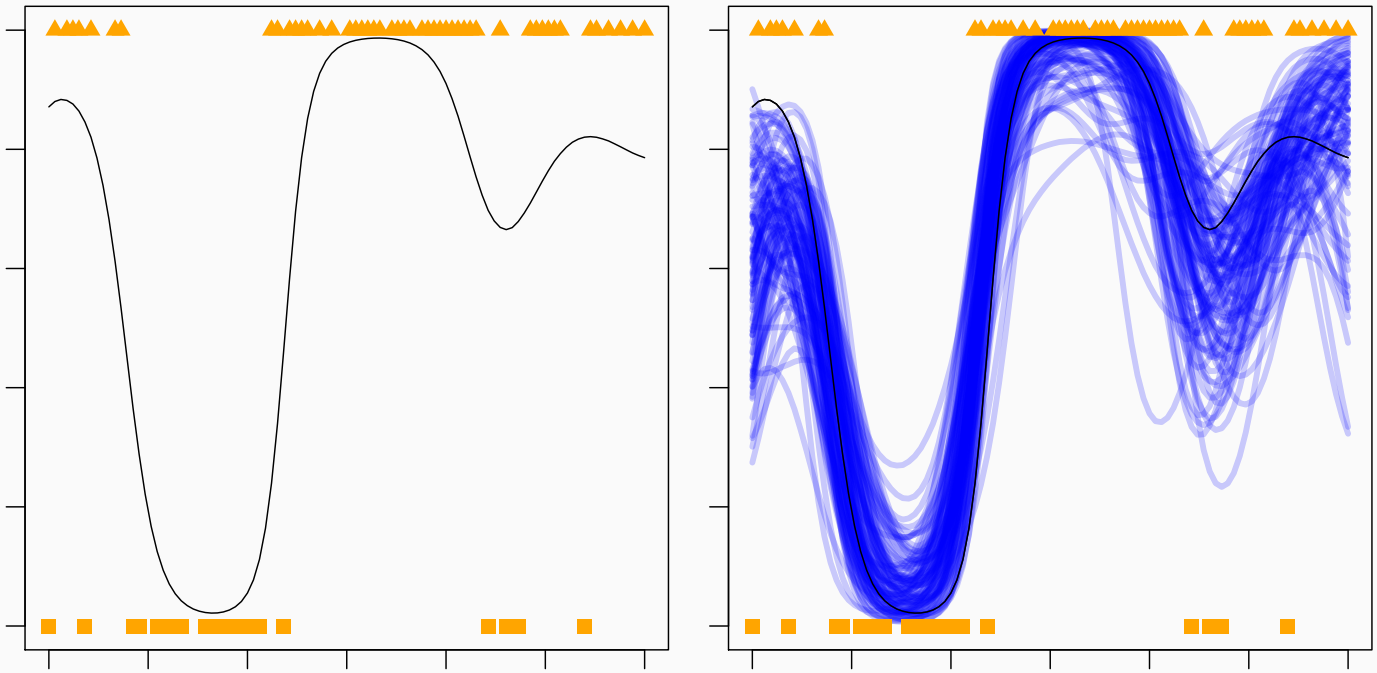
- Regression example





What do Bayesian Models Have to Offer?

- Classification example



Introduction

Bayesian Linear Models

Linear Models

- Implement a linear combination of basis functions

$$f(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\varphi}(\mathbf{x})$$

with given

$$\boldsymbol{\varphi}(\mathbf{x}) = (\varphi_1(\mathbf{x}), \ldots, \varphi_D(\mathbf{x}))^\top$$

Linear Models

- Implement a linear combination of basis functions

$$f(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\varphi}(\mathbf{x})$$

with given

$$\boldsymbol{\varphi}(\mathbf{x}) = (\varphi_1(\mathbf{x}), \ldots, \varphi_D(\mathbf{x}))^\top$$

- In regression, for example, observations are modeled as:

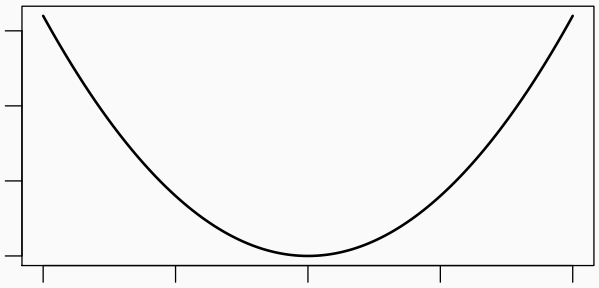
$$p(\mathbf{y}|\mathbf{w}, \mathbf{X}, \lambda) = \mathcal{N}(\boldsymbol{\Phi}\mathbf{w}, \lambda\mathbf{I})$$

$$\boldsymbol{\Phi} := \boldsymbol{\Phi}(\mathbf{X}) = \begin{bmatrix} \varphi_1(\mathbf{x}_1) & \cdots & \varphi_D(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ \varphi_1(\mathbf{x}_N) & \cdots & \varphi_D(\mathbf{x}_N) \end{bmatrix}$$

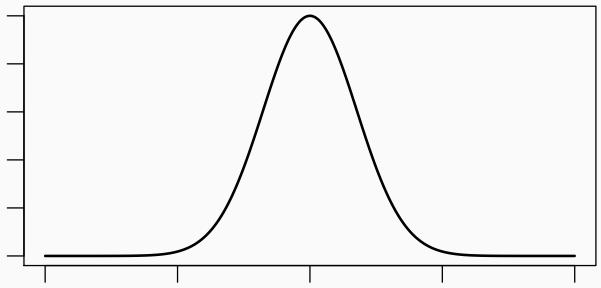
Probabilistic Interpretation of Loss Minimization

- Inputs : $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top$
- Labels : $\mathbf{y} = (y_1, \ldots, y_N)^\top$
- Weights : $\mathbf{w} = (w_1, \ldots, w_D)^\top$

Quadratic Loss



$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) \propto \exp(-\text{Loss})$

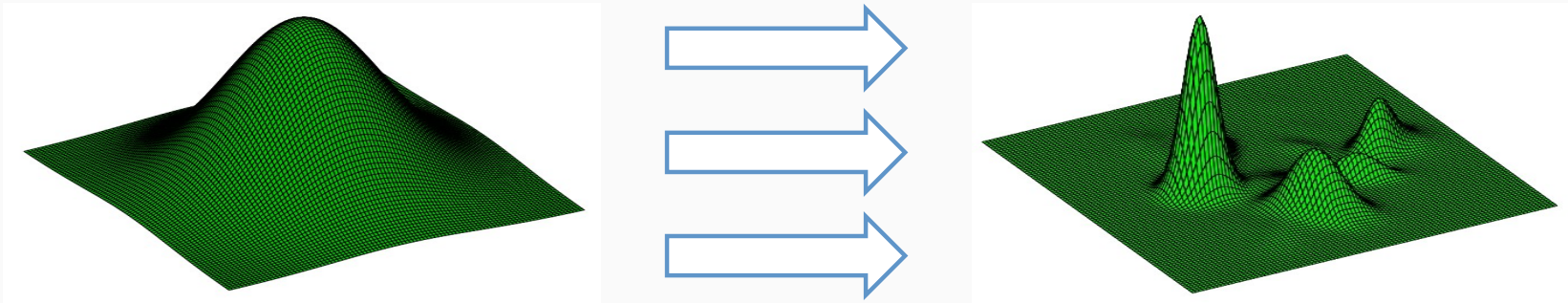


- Minimization of a loss function
- ... equivalent as maximizing likelihood $p(\mathbf{y}|\mathbf{X}, \mathbf{w})$

Bayesian Inference

- Inputs : $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top$
- Labels : $\mathbf{y} = (y_1, \ldots, y_N)^\top$
- Weights : $\mathbf{w} = (w_1, \ldots, w_D)^\top$

$p(\mathbf{w})$ $\qquad$ $p(\mathbf{w}|\mathbf{y}, \mathbf{X})$



$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})}{\int p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})d\mathbf{w}}$$

Introduction

Gaussian Processes

Bayesian Linear Regression

- Prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{S})$ induces a prior over $\mathbf{f} = \mathbf{\Phi w}$:

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K})$$

where $\mathbf{K} = \mathrm{cov}(\mathbf{f}) = \mathrm{E}[\mathbf{\Phi w w}^\top \mathbf{\Phi}^\top] = \mathbf{\Phi S \Phi}^\top$

Bayesian Linear Regression

- Prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{S})$ induces a prior over $\mathbf{f} = \mathbf{\Phi w}$:

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K})$$

  where $\mathbf{K} = \mathrm{cov}(\mathbf{f}) = \mathrm{E}[\mathbf{\Phi w w}^\top \mathbf{\Phi}^\top] = \mathbf{\Phi S \Phi}^\top$

- We can solve linear regression thinking about

$$p(\mathbf{y}|\mathbf{w}, \mathbf{X}, \lambda) = \mathcal{N}(\mathbf{\Phi w}, \lambda \mathbf{I}) \qquad p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{S})$$

  or

$$p(\mathbf{y}|\mathbf{w}, \mathbf{X}, \lambda) = \mathcal{N}(\mathbf{f}, \lambda \mathbf{I}) \qquad p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K})$$

Bayesian Linear Regression

- Prior $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{S})$ induces a prior over $\mathbf{f} = \mathbf{\Phi w}$:

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K})$$

  where $\mathbf{K} = \mathrm{cov}(\mathbf{f}) = \mathrm{E}[\mathbf{\Phi w w}^\top \mathbf{\Phi}^\top] = \mathbf{\Phi S \Phi}^\top$

- We can solve linear regression thinking about

$$p(\mathbf{y}|\mathbf{w}, \mathbf{X}, \lambda) = \mathcal{N}(\mathbf{\Phi w}, \lambda \mathbf{I}) \qquad p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{S})$$

  or

$$p(\mathbf{y}|\mathbf{w}, \mathbf{X}, \lambda) = \mathcal{N}(\mathbf{f}, \lambda \mathbf{I}) \qquad p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K})$$

- Why is this useful??

Gaussian Processes

- Linear models require specifying a set of basis functions
  - Polynomials, Trigonometric, . . .??
- We can use the fact that:

$$\mathbf{K} = \mathbf{\Phi}\mathbf{S}\mathbf{\Phi}^\top$$

to see that $\mathbf{K}$ (actually $k(\cdot, \cdot)$) induces $\mathbf{\Phi}$ (that is $\varphi(\cdot)$)

Gaussian Processes

- Linear models require specifying a set of basis functions
  - Polynomials, Trigonometric, . . .??
- We can use the fact that:

$$\mathbf{K} = \mathbf{\Phi}\mathbf{S}\mathbf{\Phi}^\top$$

  to see that $\mathbf{K}$ (actually $k(\cdot, \cdot)$) induces $\mathbf{\Phi}$ (that is $\varphi(\cdot)$)
- It is possible to choose $k(\cdot, \cdot)$ such that $\varphi(\cdot)$ infinite dimensional!

Kernels

- For example, it is possible to show that for

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2}\right)$$

  there exists a corresponding $\varphi(\cdot)$ that is infinite dimensional!
- There are other kernels satisfying this property

Bayesian Linear Regression

- Posterior over $\mathbf{w}$ **must be** Gaussian (simple exercise)

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \lambda) = \mathcal{N}(\boldsymbol{\mu}_\mathbf{w}, \boldsymbol{\Sigma}_\mathbf{w})$$

- Covariance:

$$\boldsymbol{\Sigma}_\mathbf{w} = \left(\frac{1}{\lambda}\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \mathbf{S}^{-1}\right)^{-1}$$

- Mean:

$$\boldsymbol{\mu}_\mathbf{w} = \frac{1}{\lambda}\boldsymbol{\Sigma}_\mathbf{w}\boldsymbol{\Phi}^\top\mathbf{y}$$

- Predictions – with a similar tedious exercise...

$$p(y_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*, \lambda) = \mathcal{N}\left(\boldsymbol{\varphi}(\mathbf{x}_*)^\top\boldsymbol{\mu}_\mathbf{w}, \lambda + \boldsymbol{\varphi}(\mathbf{x}_*)^\top\boldsymbol{\Sigma}_\mathbf{w}\boldsymbol{\varphi}(\mathbf{x}_*)\right)$$

Bayesian Linear Regression - Thinking about f

- Posterior over $\mathbf{f}$ **must be** Gaussian

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \lambda) = \mathcal{N}(\boldsymbol{\mu}_\mathbf{f}, \boldsymbol{\Sigma}_\mathbf{f})$$

- Covariance:

$$\boldsymbol{\Sigma}_\mathbf{f} = \mathbf{K} - \mathbf{K}(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{K}$$

- Mean:

$$\boldsymbol{\mu}_\mathbf{f} = \mathbf{K}(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}$$

- Predictions – same as those for $\mathbf{w}$... but in terms of $k(\cdot, \cdot)$

$$p(y_*|\mathbf{X}, \mathbf{y}, \mathbf{x}_*, \lambda) = \mathcal{N}\left(\mathbf{k}_*^\top(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}, \lambda + \mathbf{k}_*^\top(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{k}_*\right)$$

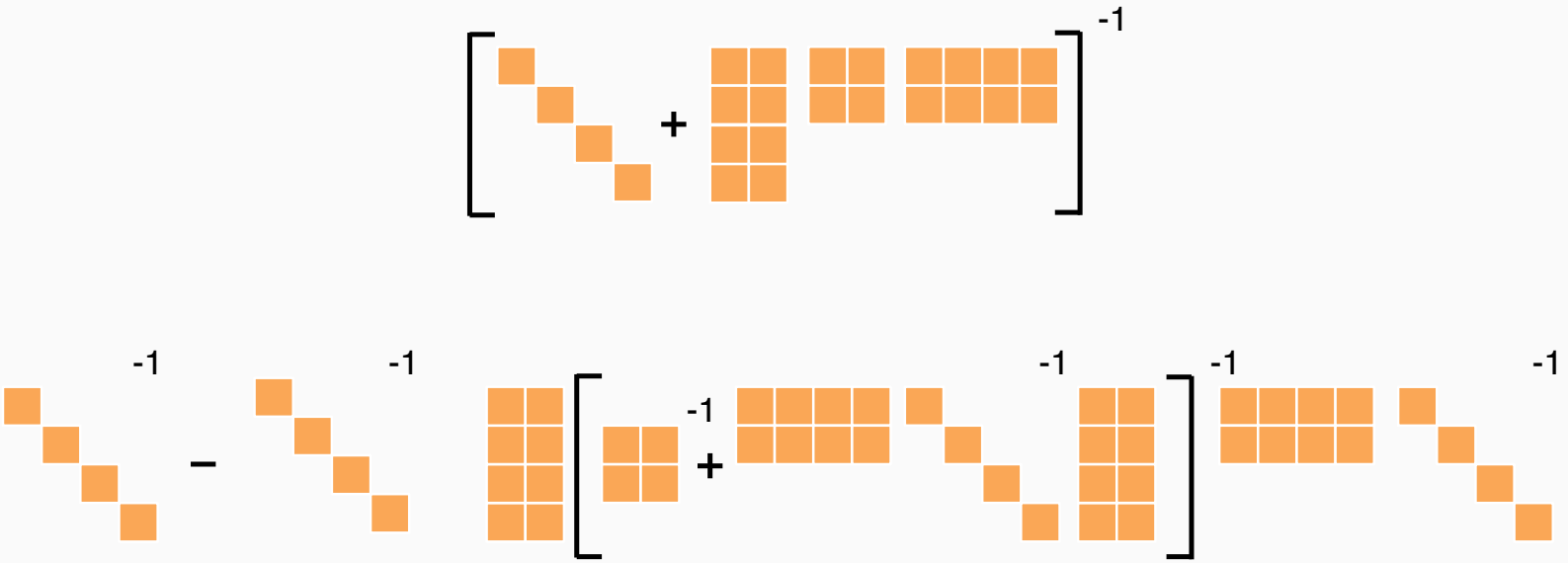
with $(\mathbf{k}_*)_i = k(\mathbf{x}_i, \mathbf{x}_*)$

Bayesian Linear Regression as a Kernel Machine

Proof sketch to help deriving the previous expressions

- Woodbury identity helps in showing that Bayesian linear regression can be formulated as a kernel machine:

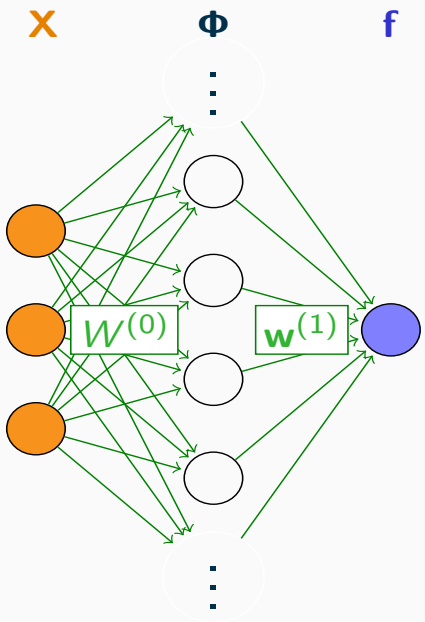
$$(\mathbf{A} + \mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{VA}^{-1}\mathbf{U})^{-1}\mathbf{VA}^{-1}$$



Bayesian Linear Regression as a Kernel Machine

- Working with $\boldsymbol{\varphi}(\cdot)$ costs $O(D^2)$ storage, $O(D^3)$ time
- Working with $k(\cdot, \cdot)$ costs $O(N^2)$ storage, $O(N^3)$ time

Gaussian Processes as Infinitely-Wide Shallow Neural Nets

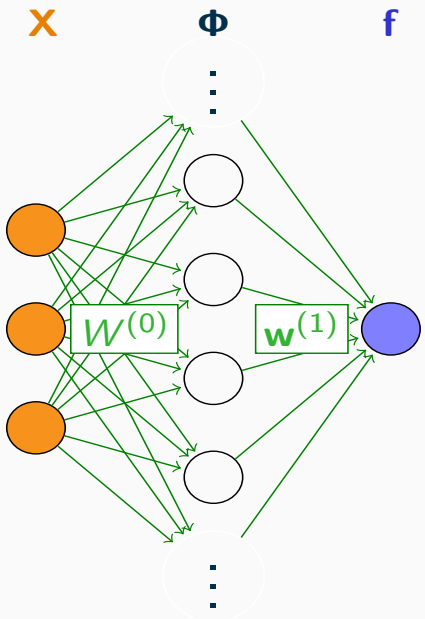


- Take $(\mathbf{W}^{(0)})_{ij} \sim \mathcal{N}(0, \alpha_0)$ and $(\mathbf{w}^{(1)})_i \sim \mathcal{N}(0, \alpha_1)$
- Central Limit Theorem implies that $\mathbf{f}$ is Gaussian

- $\mathbf{f}$ has zero-mean
- $\mathrm{cov}(\mathbf{f}) = \mathrm{E}_{p(\mathbf{W}^{(0)}, \mathbf{w}^{(1)})}[\mathbf{\Phi}(\mathbf{X}\mathbf{W}^{(0)})\mathbf{w}^{(1)}\mathbf{w}^{(1)\top}\mathbf{\Phi}(\mathbf{X}\mathbf{W}^{(0)})^\top]$

Neal, *LNS*, 1996

Gaussian Processes as Infinitely-Wide Shallow Neural Nets



- Take $(\mathbf{W}^{(0)})_{ij} \sim \mathcal{N}(0, \alpha_0)$ and $(\mathbf{w}^{(1)})_i \sim \mathcal{N}(0, \alpha_1)$
- Central Limit Theorem implies that $\mathbf{f}$ is Gaussian

- $\mathbf{f}$ has zero-mean
- $\mathrm{cov}(\mathbf{f}) = \alpha_1 \mathrm{E}_{p(\mathbf{W}^{(0)})}[\mathbf{\Phi}(\mathbf{X}\mathbf{W}^{(0)})\mathbf{\Phi}(\mathbf{X}\mathbf{W}^{(0)})^\top]$
- Some choices of $\mathbf{\Phi}$ lead to analytic expression of known kernels (Gaussian, Matérn, arc-cosine, Brownian motion, ...)

Neal, *LNS*, 1996

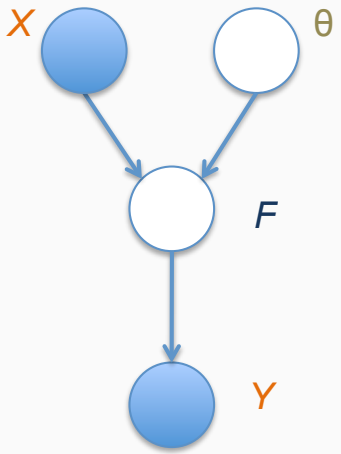
Introduction

Challenges

Challenges and Limitations

- Kernel design $k(\cdot, \cdot|\boldsymbol{\theta})$
- GPs might be too expensive (factorize $\mathbf{K}$)
- GP models might not even be tractable! Marginal likelihood computable under conjugacy only

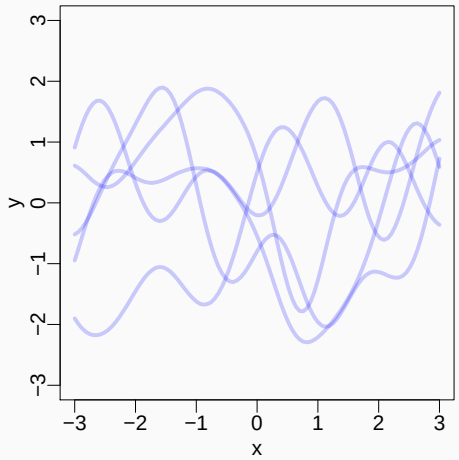
$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta}) d\mathbf{f}$$



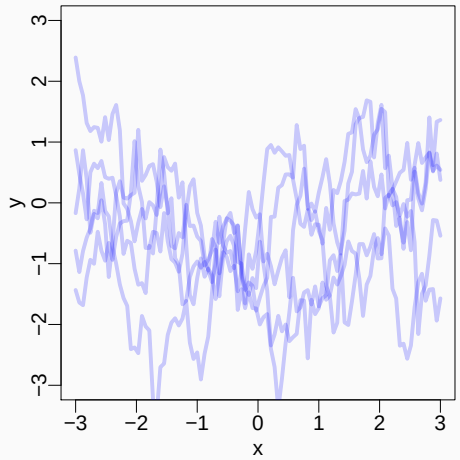
Kernel Design

- The choice of a kernel is critical for good performance
- This encodes any assumptions on the prior over functions

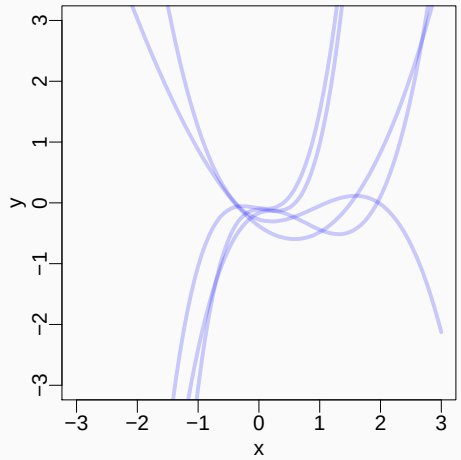
Gaussian



Matérn

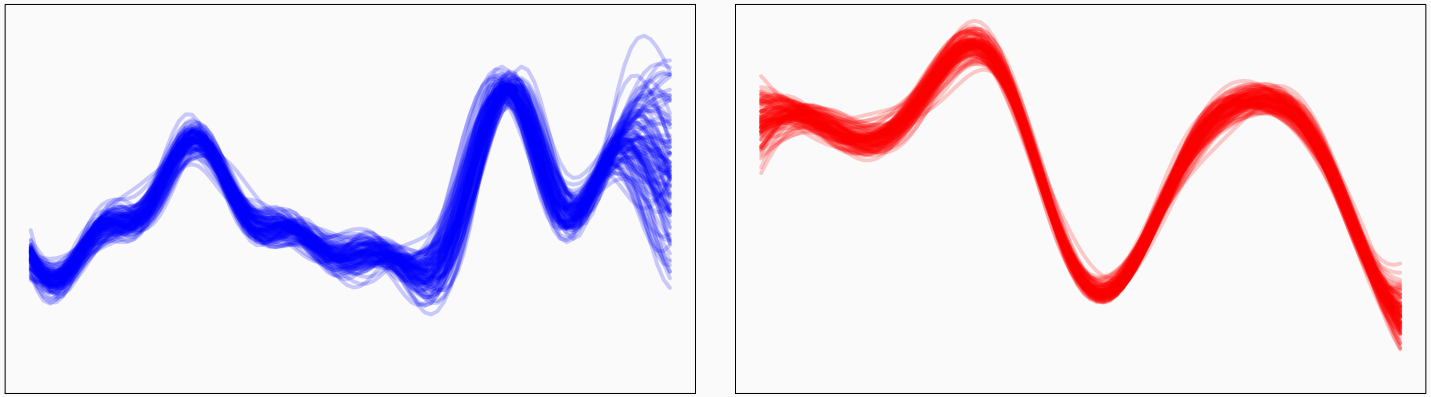


Polynomial



Deep Gaussian Processes for Large Representational Power

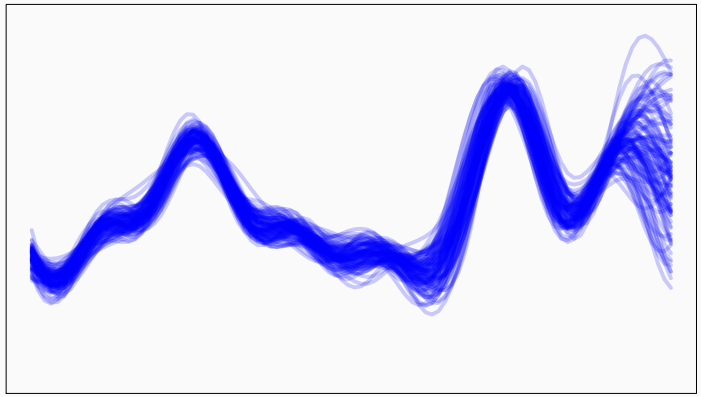
- Bypassing kernel design through **composition** of processes

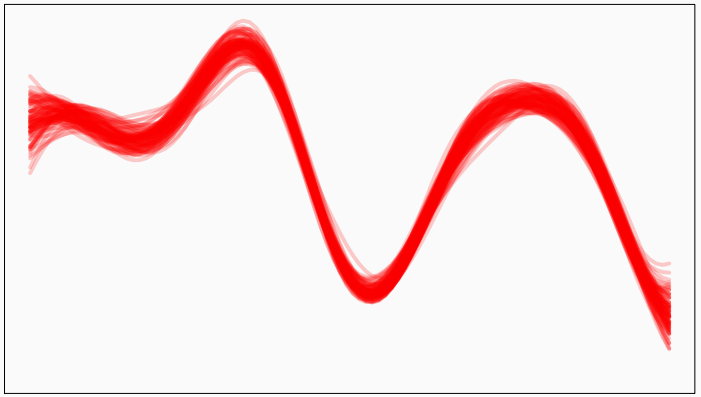


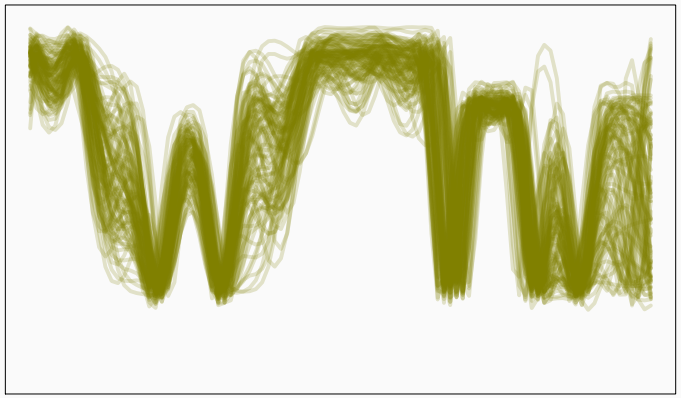
$$(f \circ g)(x)??$$

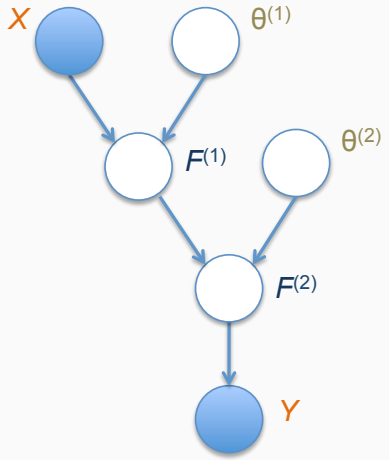
Deep Gaussian Processes for Large Representational Power

- Composition of stationary processes yields something much more complex





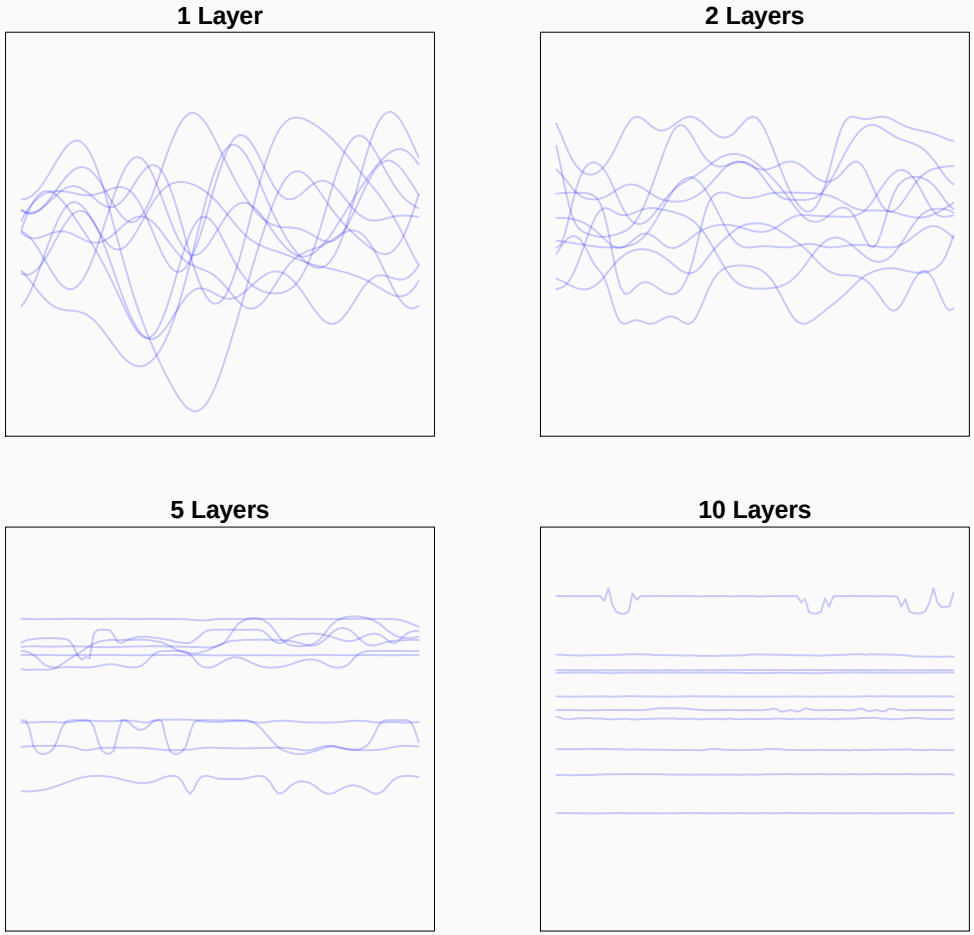


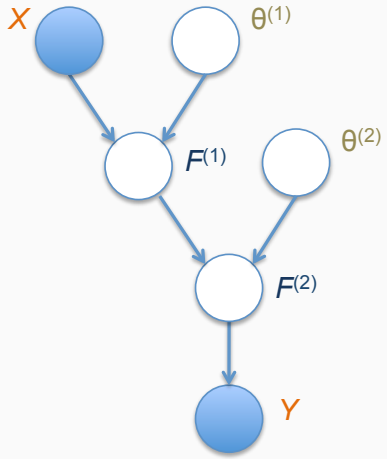


Neal, *LNS*, 1996 – Damianou and Lawrence, *AISTATS*, 2013 – Duvenaud et al., *AISTATS*, 2014

Pathologies of Deep Gaussian Processes

- Deep is not necessarily good!
- Example

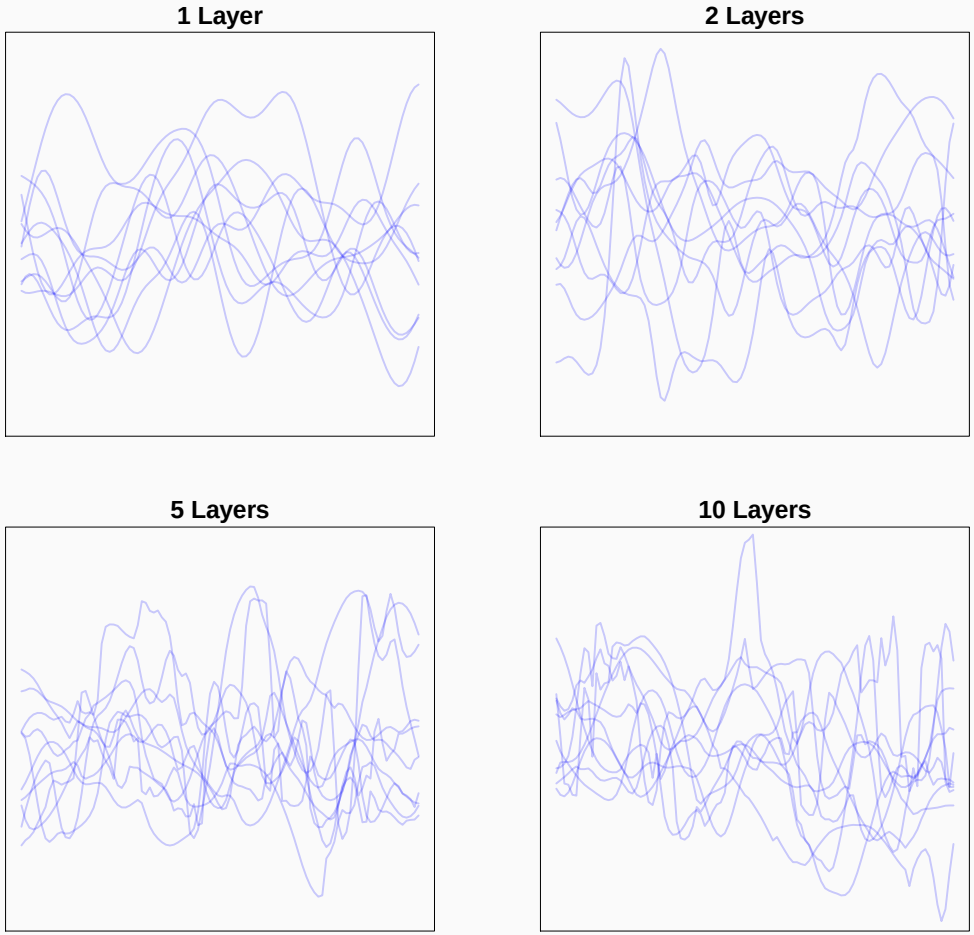


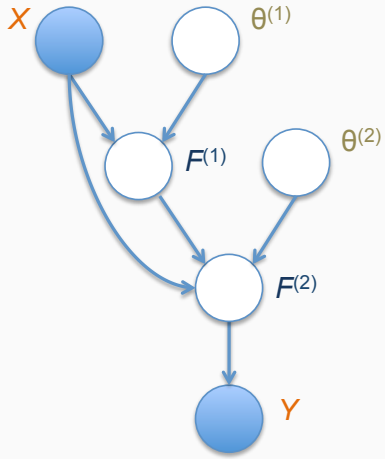


Neal, *LNS*, 1996 – Duvenaud et al., *AISTATS*, 2014 – Matthews et al., *arXiv*, 2018

Pathologies of Deep Gaussian Processes

- Deep is not necessarily good!
- Feeding input to each layer helps...





Neal, *LNS*, 1996 – Duvenaud et al., *AISTATS*, 2014 – Matthews et al., *arXiv*, 2018

Learning Deep Gaussian Processes

- Inference requires calculating integrals of this kind:

$$
\begin{aligned}
p(\mathbf{Y}|\mathbf{X},\boldsymbol{\theta}) \quad = \quad & \int p\left(\mathbf{Y}|\mathbf{F}^{(N_\mathrm{h})},\boldsymbol{\theta}^{(N_\mathrm{h})}\right) \times \\
& p\left(\mathbf{F}^{(N_\mathrm{h})}|\mathbf{F}^{(N_\mathrm{h}-1)},\boldsymbol{\theta}^{(N_\mathrm{h}-1)}\right) \times \ldots \times \\
& p\left(\mathbf{F}^{(1)}|\mathbf{X},\boldsymbol{\theta}^{(0)}\right) d\mathbf{F}^{(N_\mathrm{h})} \ldots d\mathbf{F}^{(1)}
\end{aligned}
$$

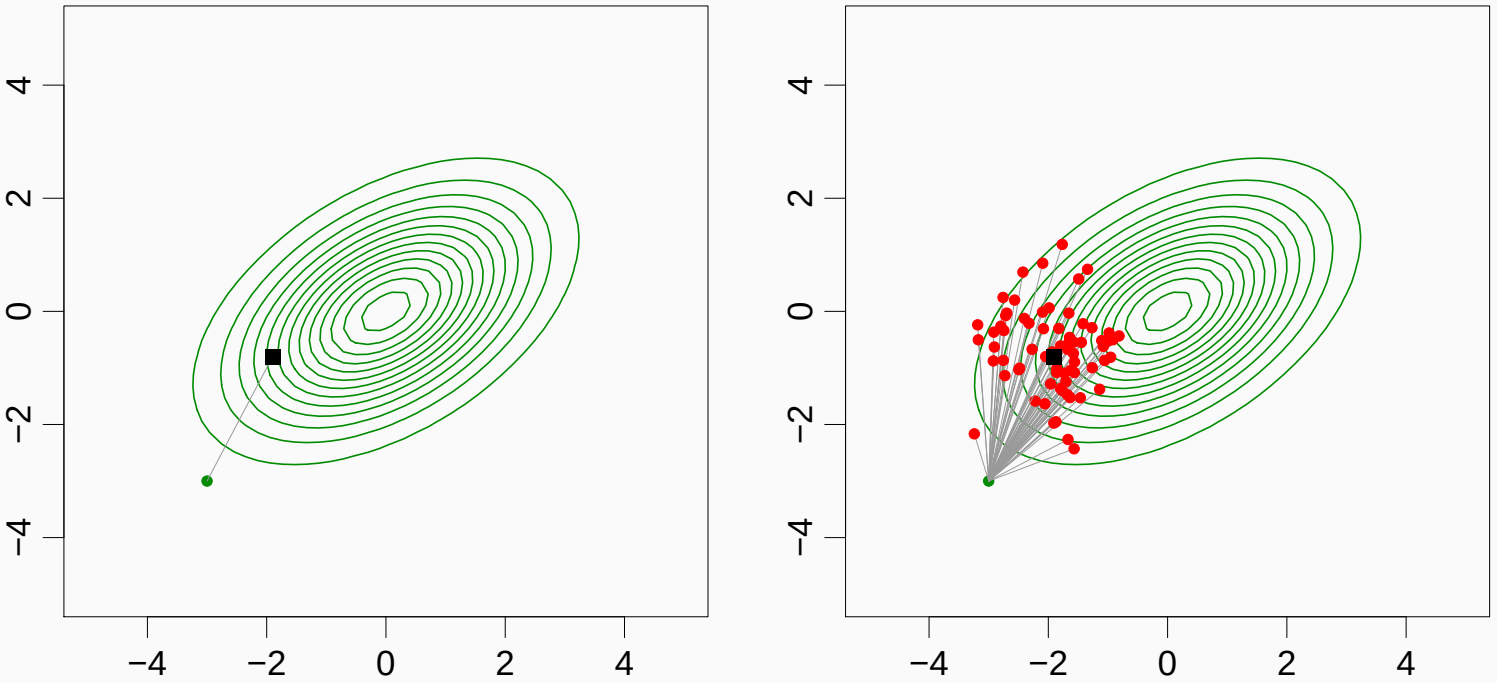
- Extremely challenging!

The Deep Learning Revolution

- Large representational power
- Mini-batch-based learning
- Exploit GPU and distributed computing
- Automatic differentiation
- Mature development of regularization (e.g., dropout)
- Application-specific representations (e.g., convolutional)

Stochastic Gradient Optimization

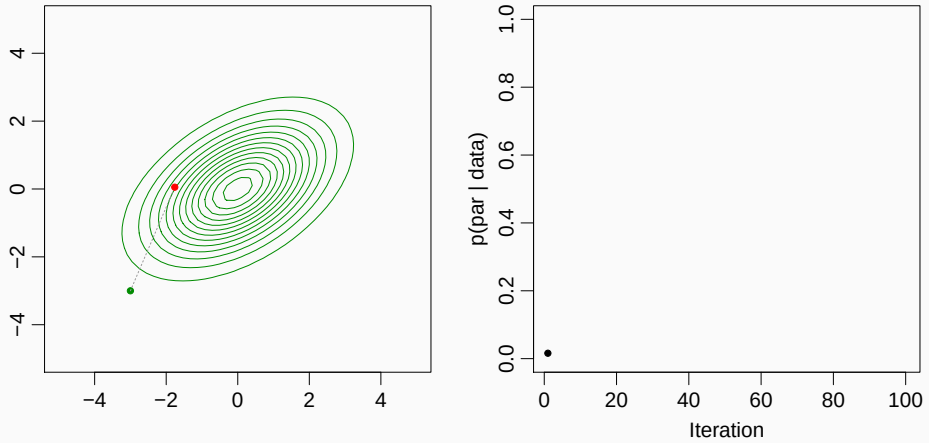
$$\mathrm{E}\left\{\widetilde{\nabla_{\mathrm{vpar}}\mathrm{LowerBound}}\right\} = \nabla_{\mathrm{vpar}}\mathrm{LowerBound}$$

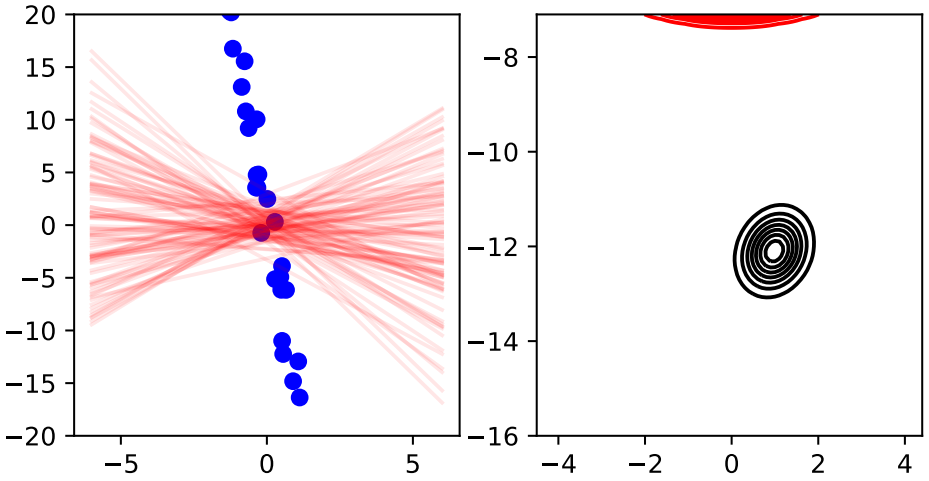


Robbins and Monro, *AoMS*, 1951

Stochastic Variational Inference - Illustration

$$\text{vpar}' = \text{vpar} + \frac{\alpha_t}{2}\widetilde{\nabla_{\text{vpar}}}(\text{LowerBound}) \qquad \alpha_t \to 0$$





Is There Any Hope for GPs and DGPs?

- Mini-batch training is straightforward when objective factorizes over training points

$$\text{objective} = \sum_i g(\mathbf{y}_i, \mathbf{x}_i, \text{par})$$

Is There Any Hope for GPs and DGPs?

- Mini-batch training is straightforward when objective factorizes over training points

$$\text{objective} = \sum_i g(\mathbf{y}_i, \mathbf{x}_i, \text{par})$$

- In GPs latent variables are fully correlated

$$p(\mathbf{f}|\mathbf{X}, \theta) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K}(\mathbf{X}, \theta)) \propto \exp\left(-\frac{1}{2}\mathbf{f}^\top \mathbf{K}^{-1}\mathbf{f}\right)$$

- Naïve mini-batch approaches would still require $\mathbf{K}^{-1}$

Can we exploit what made Deep Learning successful for practical and scalable learning of (Deep) Gaussian processes?

Deep Gaussian Processes

Deep Gaussian Processes

Inference and Approximations

Main Ingredients for Inference for DGPs

Gaussian posterior approximations for GPs:

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \theta) \approx q(\mathbf{f})$$

- Laplace Approximation – Barber and Williams, *IEEE TPAMI*, 1998
- Expectation Propagation – Seeger, *Tech. Rep.*, 2002

$$p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \theta) \propto p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{X}, \theta) \approx p(\mathbf{f}|\mathbf{X}, \theta) \prod_i q(y_i|f_i)$$

- Variational Bayes – Opper and Archambeau, *Naural Comput.*, 2009

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

- MCMC – Filippone et al., *Mach. Learn.*, 2013

Main Ingredients for Inference for DGPs

- Sparse GP (Nyström) approximations - introduce $M < N$ new latent variables $\mathbf{u}$ at locations $\mathbf{Z}$.

$$\mathbf{K} \approx \mathbf{K}_{\mathbf{xz}}\mathbf{K}_{\mathbf{zz}}^{-1}\mathbf{K}_{\mathbf{zx}}$$

  - Fully Independent Training Conditionals (FITC)
  - Partially Independent Training Conditionals (PITC)
  - Titsias, *AISTATS*, 2009 – Variationally Sparse GPs

$$q(\mathbf{f}, \mathbf{u}) \approx p(\mathbf{f}|\mathbf{u})q(\mathbf{u})$$

- Random Feature Expansions
- Kronecker/Toepliz/Tensor structured GPs

Candela and Rasmussen, *JMLR*, 2005 – Titsias, *AISTATS*, 2009 – Lázaro-Gredilla et al., *JMLR*, 2010 – Wilson and Nickisch., *ICML*, 2015 – Izmailov et al., *AISTATS*, 2018

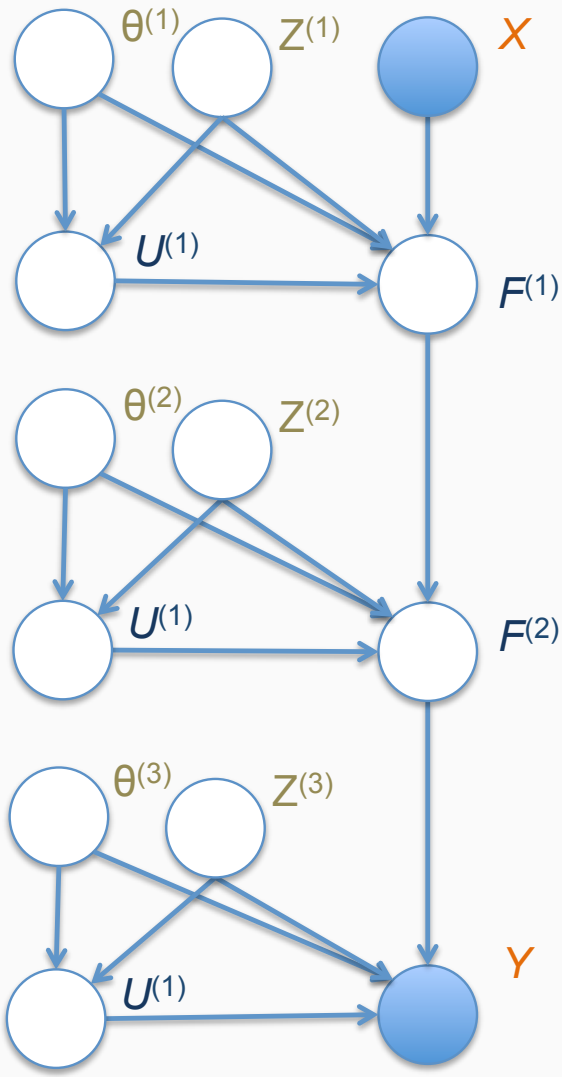
Inference for DGPs

- Inducing points-based approximations
  - VI+Titsias *AISTATS* 2009 Sparse GP
    - Damianou and Lawrence, *AISTATS*, 2013
    - Hensman and Lawrence, *arXiv*, 2014
    - Salimbeni and Deisenroth, *NIPS*, 2017
  - EP+FITC - Bui et al. *ICML*, 2016
  - MCMC+Titsias *AISTATS* 2009 Sparse GP
    - Havasi et al., *NIPS*, 2018
- VI+Random feature-based approximations
  - Gal and Ghahramani, ICML 2016
  - Cutajar et al., ICML 2017

Scalable Expectation Propagation for DGPs

- Pseudo-inputs $Z^{(i)}$
- Inducing variables $U^{(i)}$
- VI targets

$$q\left(U^{(i)}\right)$$



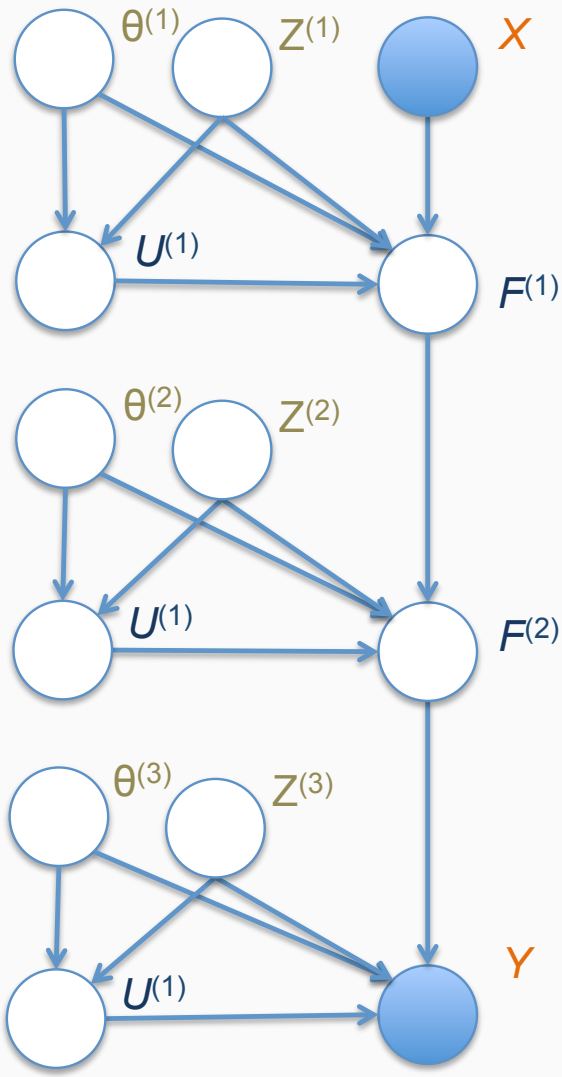
Bui et al., *ICML*, 2016

Scalable Expectation Propagation for DGPs

- Pseudo-inputs $Z^{(i)}$
- Inducing variables $U^{(i)}$
- VI targets

$$q\left(U^{(i)}\right)$$

- Assuming $q\left(U^{(i)}\right) \propto p\left(U^{(i)}\right) g\left(U^{(i)}\right)^N$ learn g as an average data factor
- Reduces memory and allows for factorization of the objective (output of each layer made Gaussian)

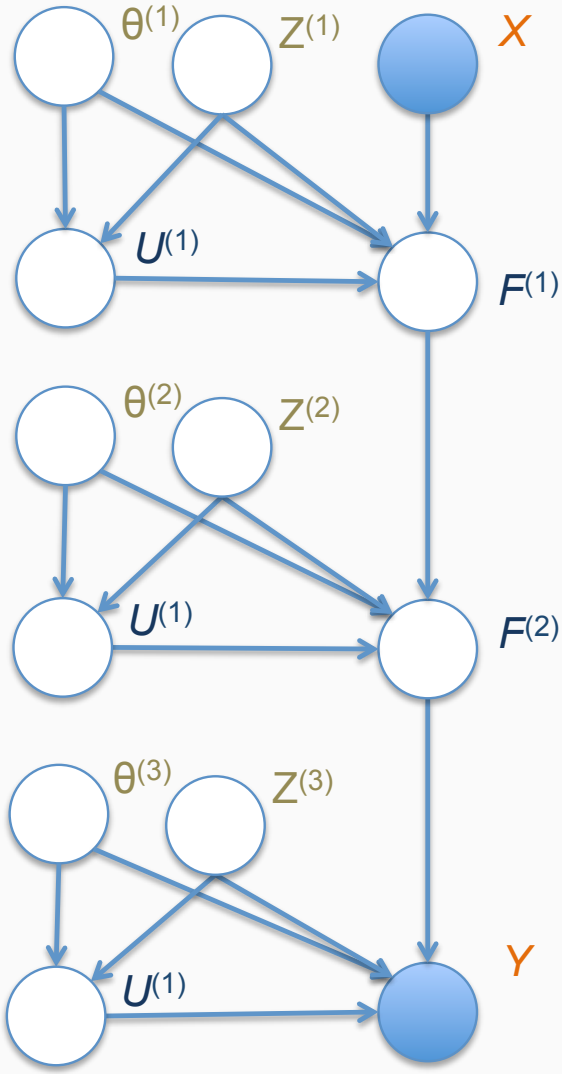


Bui et al., *ICML*, 2016

Inducing Points for DGPs extending Titsias, AISTATS, 2009

- Pseudo-inputs $Z^{(i)}$
- Inducing variables $U^{(i)}$
- VI targets $q\left(F^{(i)}, U^{(i)} | F^{(i-1)}\right)$

$$p\left(F^{(i)} | U^{(i)}, F^{(i-1)}\right) q\left(U^{(i)}\right)$$



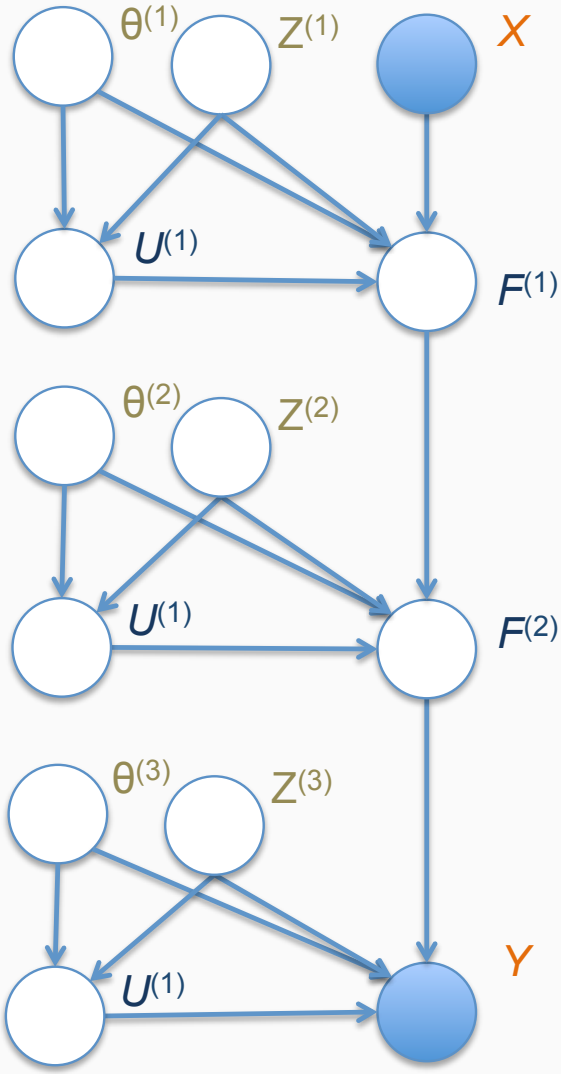
Hensman and Lawrence, *arXiv*, 2014 – Salimbeni and Deisenroth, *NIPS*, 2017

Inducing Points for DGPs extending Titsias, AISTATS, 2009

- Pseudo-inputs $Z^{(i)}$
- Inducing variables $U^{(i)}$
- VI targets $q\left(F^{(i)}, U^{(i)} | F^{(i-1)}\right)$

$$p\left(F^{(i)} | U^{(i)}, F^{(i-1)}\right) q\left(U^{(i)}\right)$$

- Lower bound factorizes across training points...
- ... and the ith marginal of the final layer depends only on the ith marginals of all layers



Hensman and Lawrence, *arXiv*, 2014 – Salimbeni and Deisenroth, *NIPS*, 2017

Random Feature Expansions for DGPs - Bochner's theorem

- Continuous shift-invariant covariance function

$$k(\mathbf{x}_i - \mathbf{x}_j|\boldsymbol{\theta}) = \sigma^2 \int p(\boldsymbol{\omega}|\boldsymbol{\theta}) \exp\left(\iota(\mathbf{x}_i - \mathbf{x}_j)^\top \boldsymbol{\omega}\right) d\boldsymbol{\omega}$$

Random Feature Expansions for DGPs - Bochner's theorem

- Continuous shift-invariant covariance function

$$k(\mathbf{x}_i - \mathbf{x}_j|\boldsymbol{\theta}) = \sigma^2 \int p(\boldsymbol{\omega}|\boldsymbol{\theta}) \exp\left(\iota(\mathbf{x}_i - \mathbf{x}_j)^\top \boldsymbol{\omega}\right) d\boldsymbol{\omega}$$

- Monte Carlo estimate

$$k(\mathbf{x}_i - \mathbf{x}_j|\boldsymbol{\theta}) \approx \frac{\sigma^2}{N_{\mathrm{RF}}} \sum_{r=1}^{N_{\mathrm{RF}}} \mathbf{z}(\mathbf{x}_i|\tilde{\boldsymbol{\omega}}_r)^\top \mathbf{z}(\mathbf{x}_j|\tilde{\boldsymbol{\omega}}_r)$$

with

$$\tilde{\boldsymbol{\omega}}_r \sim p(\boldsymbol{\omega}|\boldsymbol{\theta})$$

$$\mathbf{z}(\mathbf{x}|\boldsymbol{\omega}) = [\cos(\mathbf{x}^\top \boldsymbol{\omega}), \sin(\mathbf{x}^\top \boldsymbol{\omega})]^\top$$

Rahimi and Recht, *NIPS*, 2008 - Lázaro-Gredilla et al., *JMLR*, 2010

Random Feature Expansions for DGPs

- Define

$$\boldsymbol{\Phi}^{(l)} = \sqrt{\frac{\sigma^2}{N_{\mathrm{RF}}^{(l)}}} \left[\cos\left(\mathbf{F}^{(l)}\boldsymbol{\Omega}^{(l)}\right), \sin\left(\mathbf{F}^{(l)}\boldsymbol{\Omega}^{(l)}\right)\right]$$

  and

$$\mathbf{F}^{(l+1)} = \boldsymbol{\Phi}^{(l)} W^{(l)}$$

- We are stacking Bayesian linear models with

$$p\left(W_{\cdot i}^{(l)}\right) = \mathcal{N}\left(\mathbf{0}, I\right)$$

Random Feature Expansions for DGPs

- Define

$$\mathbf{\Phi}^{(l)} = \sqrt{\frac{\sigma^2}{N_{\mathrm{RF}}^{(l)}}} \left[\cos\left(\mathbf{F}^{(l)}\mathbf{\Omega}^{(l)}\right), \sin\left(\mathbf{F}^{(l)}\mathbf{\Omega}^{(l)}\right)\right]$$

  and

$$\mathbf{F}^{(l+1)} = \mathbf{\Phi}^{(l)} W^{(l)}$$

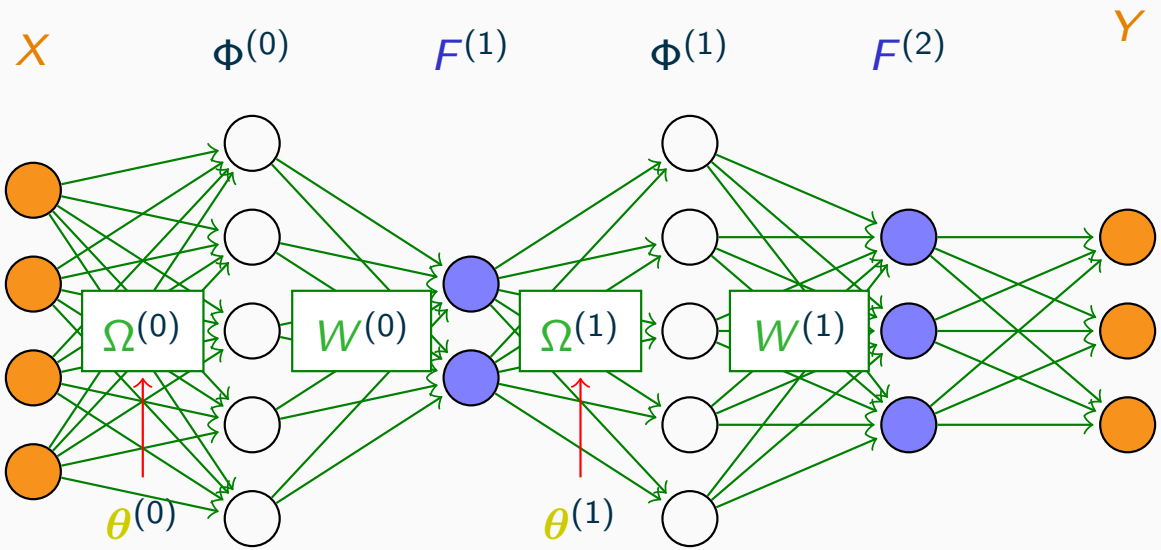
- We are stacking Bayesian linear models with

$$p\left(W_{\cdot i}^{(l)}\right) = \mathcal{N}\left(\mathbf{0}, I\right)$$

- Expansion of arc-cosine kernel yields ReLU activations!

Cutajar, Bonilla, Michiardi, Filippone, *ICML*, 2017

DGPs with random features become DNNs



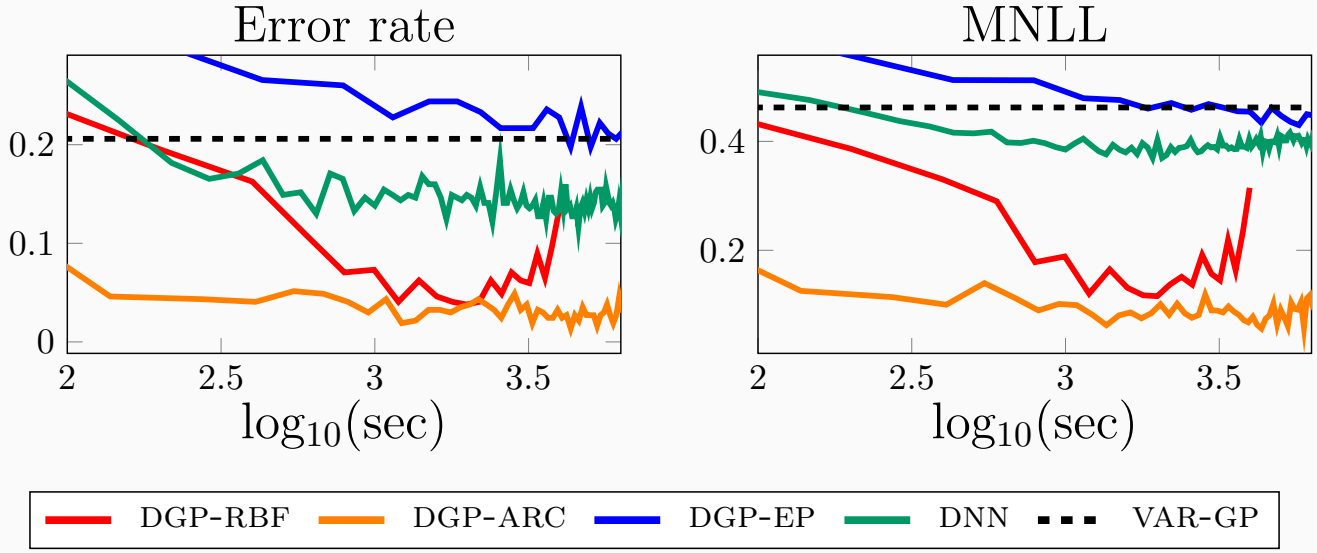
We can learn the model using Stochastic Variational Inference for Bayesian DNNs!

Cutajar, Bonilla, Michiardi, Filippone, *ICML*, 2017

Results - Classification

EEG dataset

$(n = 14979,\ d = 14)$

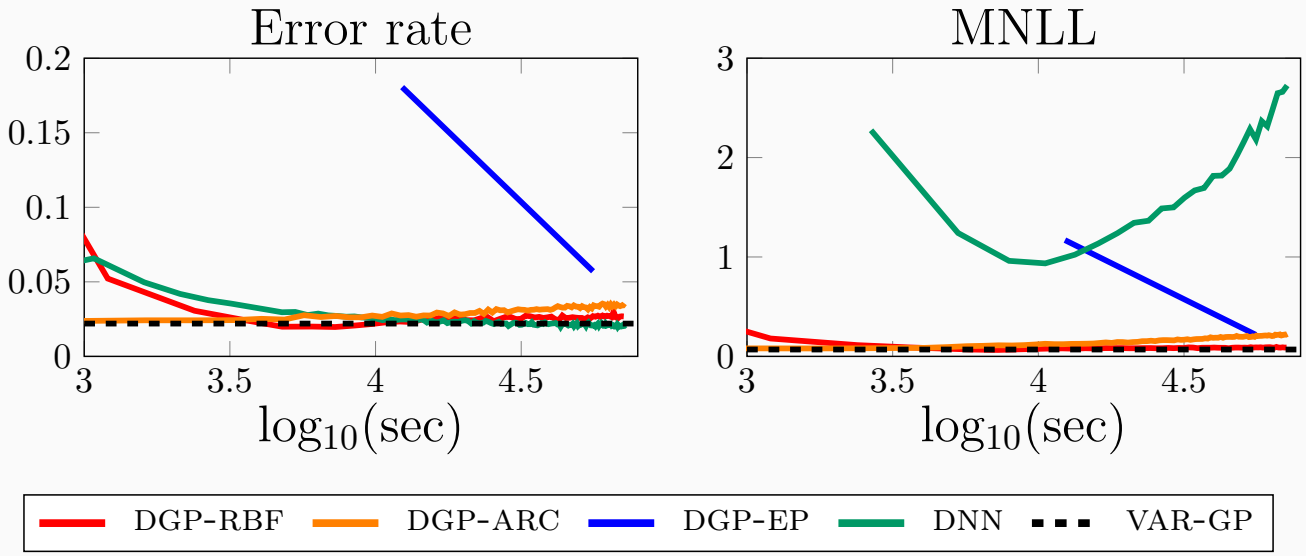


Cutajar, Bonilla, Michiardi, Filippone, *ICML*, 2017

Results - Multiclass Classification

MNIST dataset

$(n = 60000,\ d = 784)$



Cutajar, Bonilla, Michiardi, Filippone, *ICML*, 2017

Results - MNIST-8M

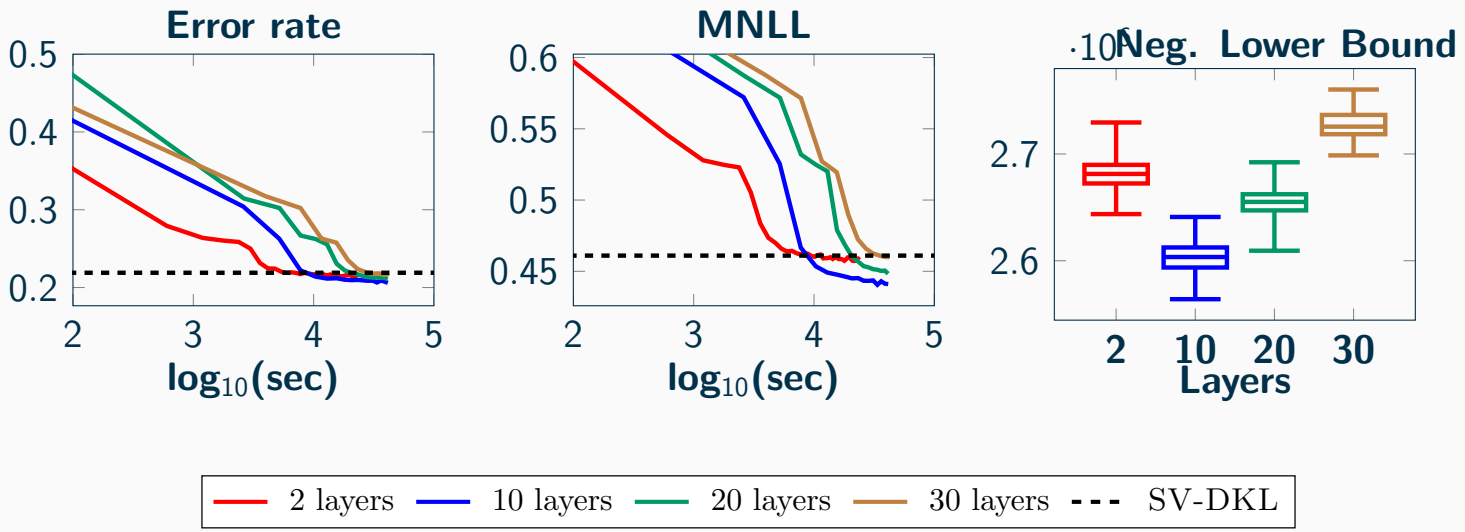
- Variant of MNIST with $8.1\mathrm{M}$ images
- $99+\%$ accuracy!
- Also, check out Krauth et al., UAI 2017

Cutajar, Bonilla, Michiardi, Filippone, *ICML*, 2017 – Krauth, Cutajar, Bonilla, Filippone, *UAI*, 2017

Results - Model (Depth) Selection

Airline dataset

$(n = 5\text{M}+,\ d = 8)$



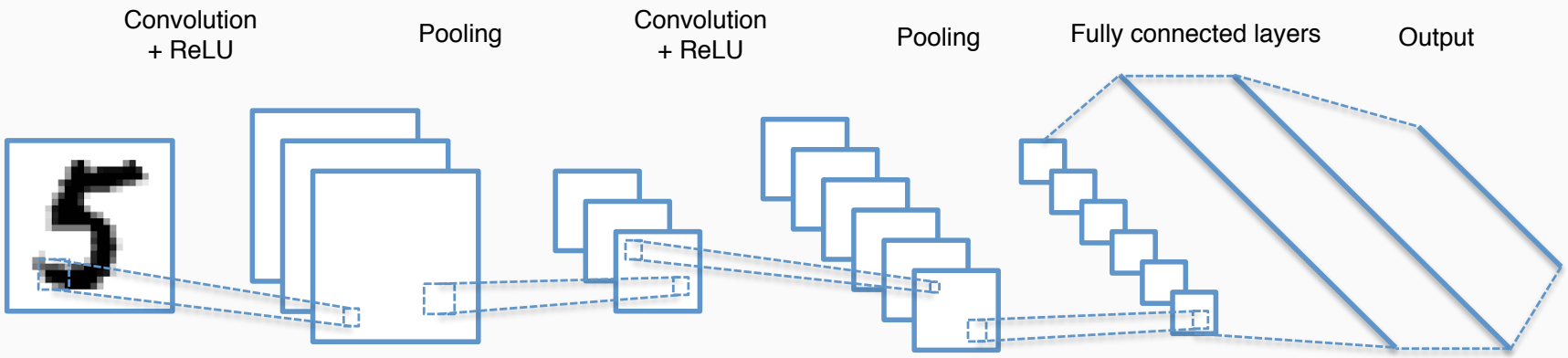
Cutajar, Bonilla, Michiardi, Filippone, *ICML*, 2017

Deep Gaussian Processes

Convolutional Deep Gaussian Processes

Convolutional Nets

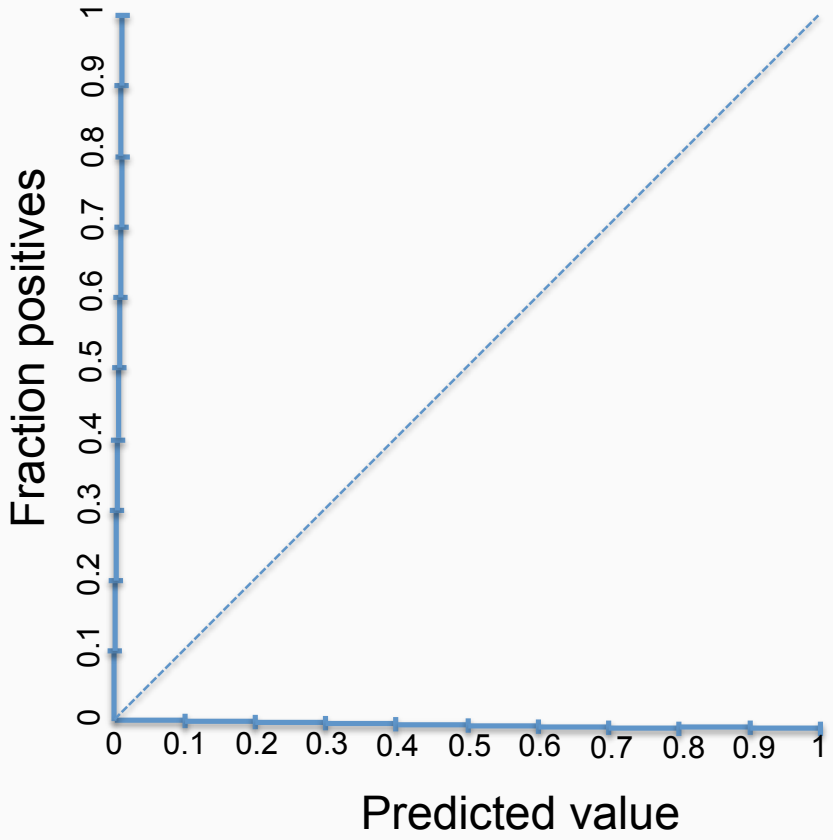
- Convolutional nets are widely used. . .
- . . .but they are known to be overconfident!



Guo et al., *ICML*, 2017

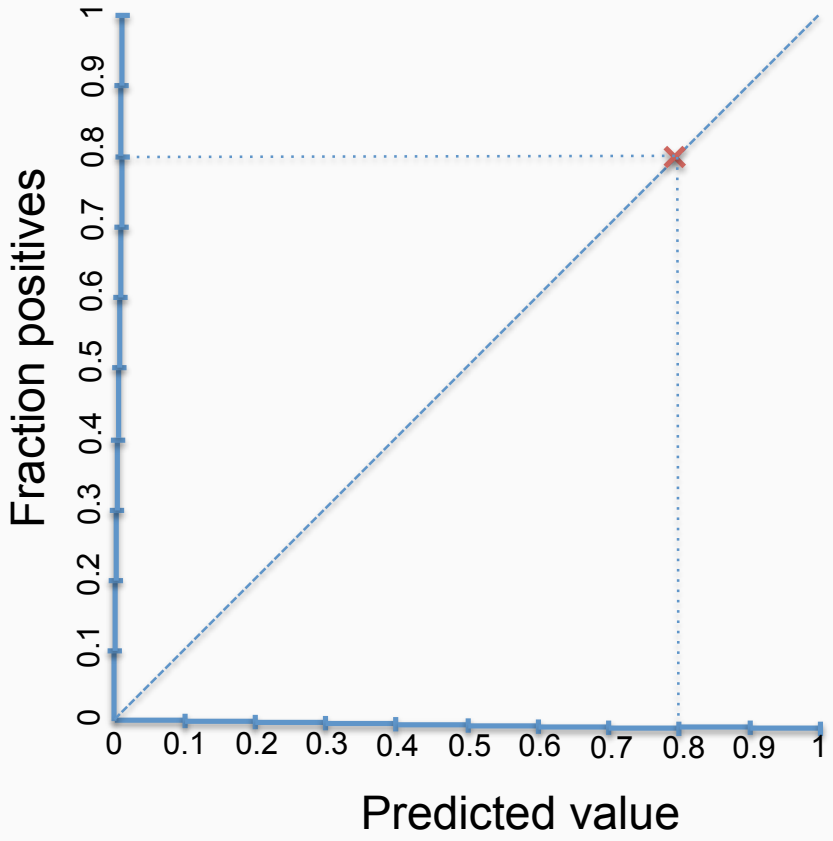
Calibration as a Measure of Quantification of Uncertainty

- Reliability diagrams



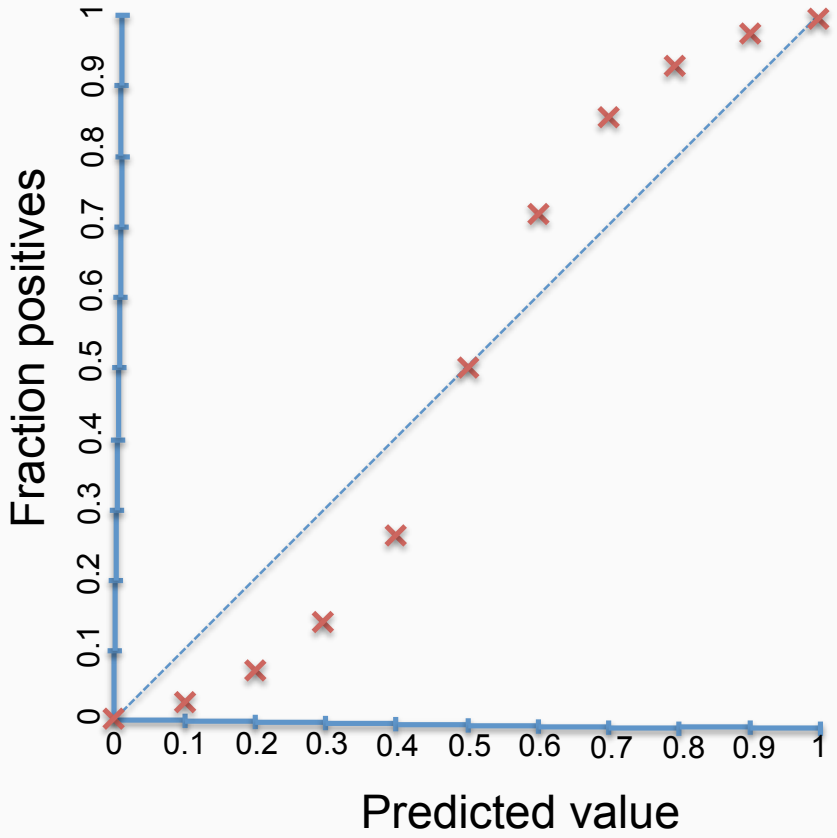
Calibration as a Measure of Quantification of Uncertainty

- Reliability diagrams



Calibration as a Measure of Quantification of Uncertainty

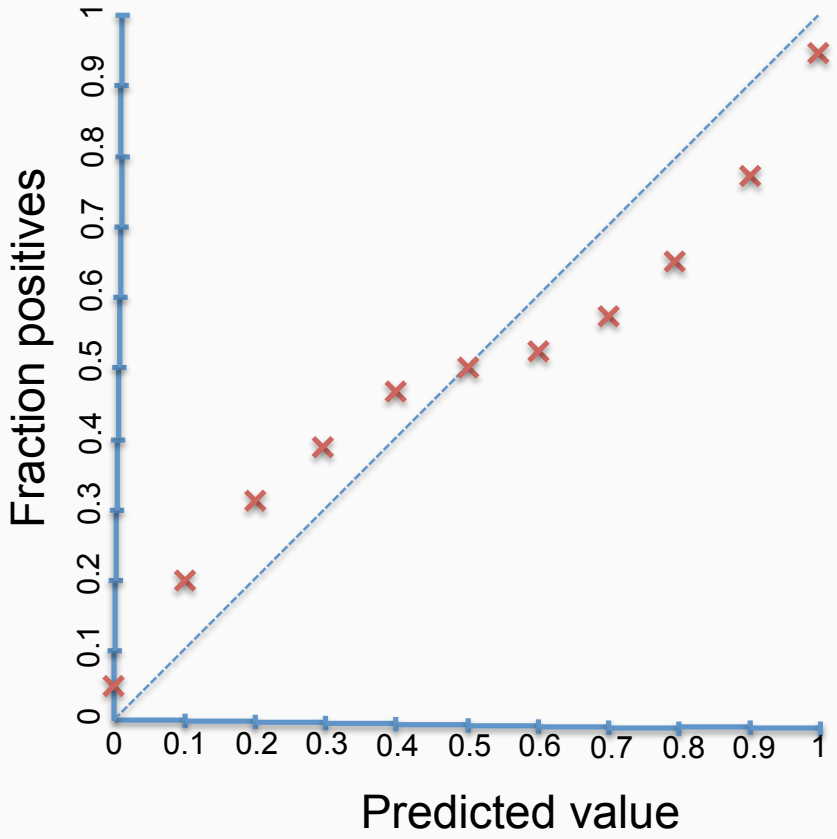
- Reliability diagrams - Under-confident predictions



- We can extract the Expected Calibration Error (ECE) score
- The BRIER score is another measure of calibration

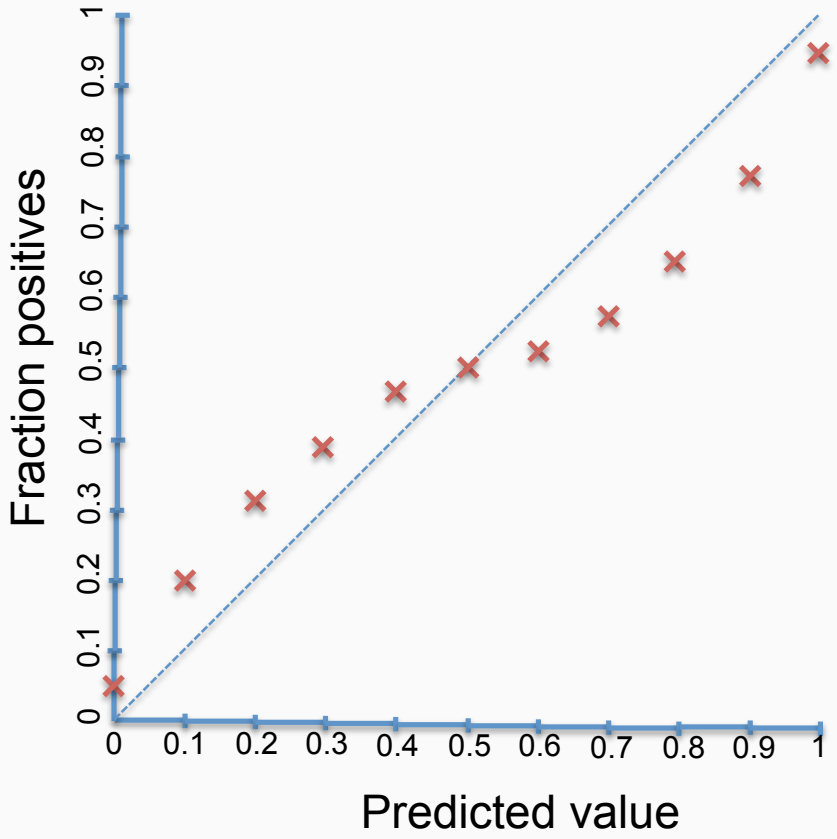
Calibration as a Measure of Quantification of Uncertainty

- Reliability diagrams - Overconfident predictions



Calibration as a Measure of Quantification of Uncertainty

- Reliability diagrams - Overconfident predictions

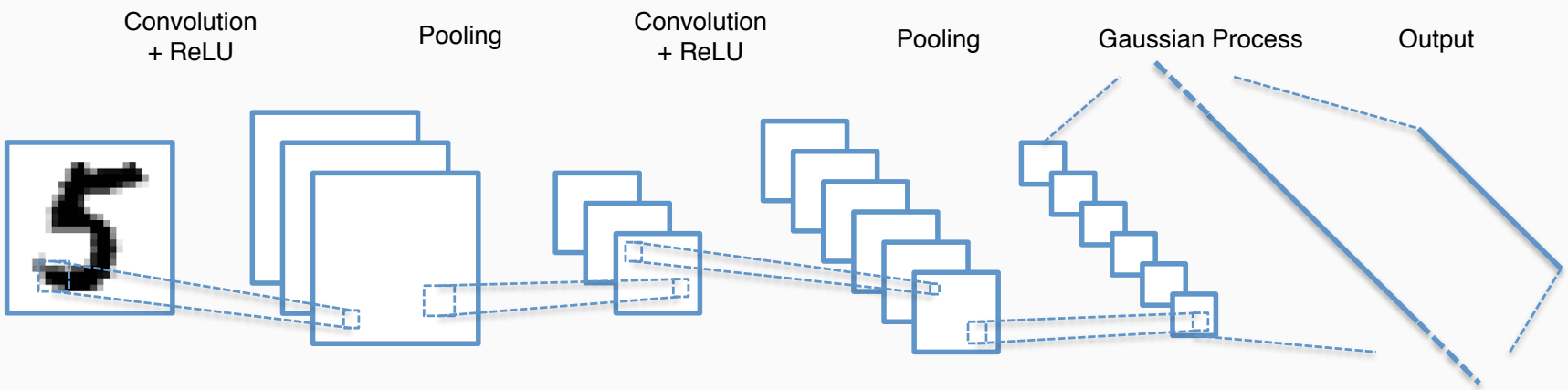


Reliability diagrams of modern Deep CNNs look like this!

Post-calibration fixes it

Combining Convolutional Nets with GPs

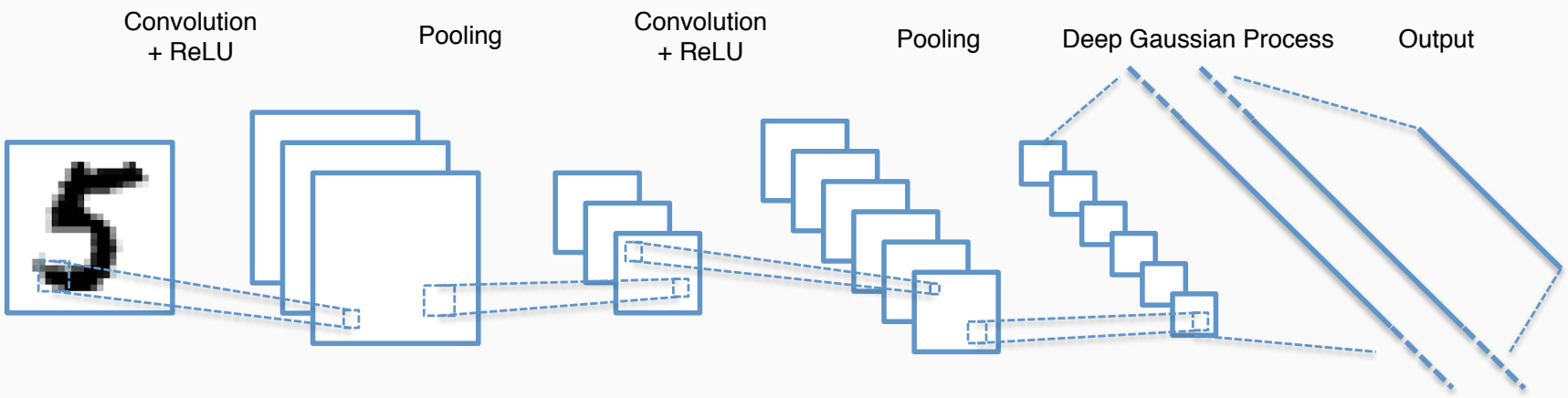
- There have been attempts to combine CNNs with GPs
- Most popular ones replace fully connected layers with GPs



Wilson et al., *NIPS*, 2016 – Bradshaw et al., *arXiv*, 2017 – Tran et al., *AISTATS*, 2019

Combining Convolutional Nets with GPs

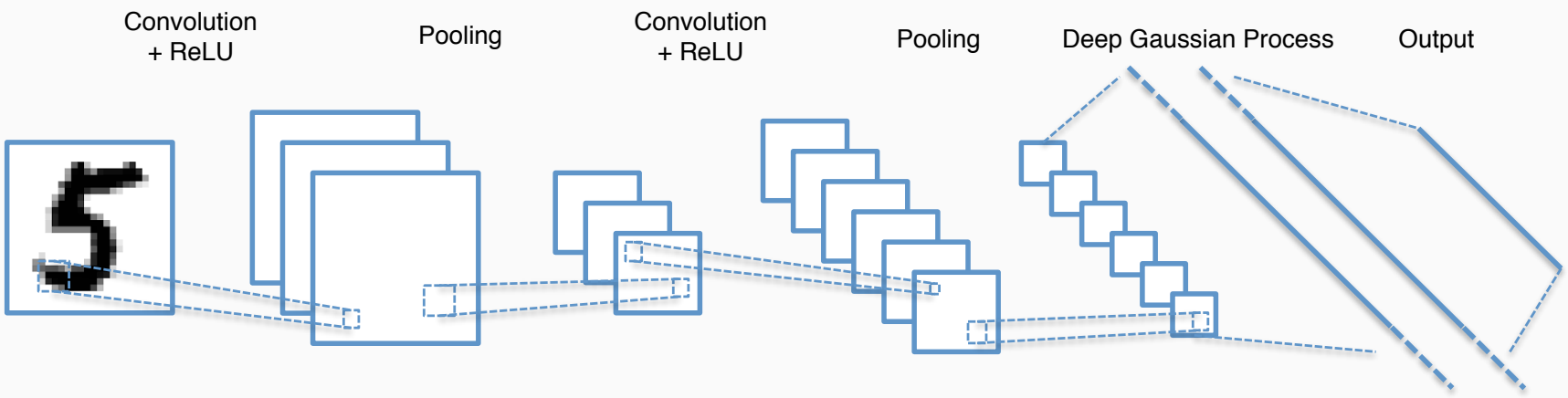
- There have been attempts to combine CNNs with GPs
- Most popular ones replace fully connected layers with GPs



- Better quantification of uncertainty??

Combining Convolutional Nets with GPs

- There have been attempts to combine CNNs with GPs
- Most popular ones replace fully connected layers with GPs

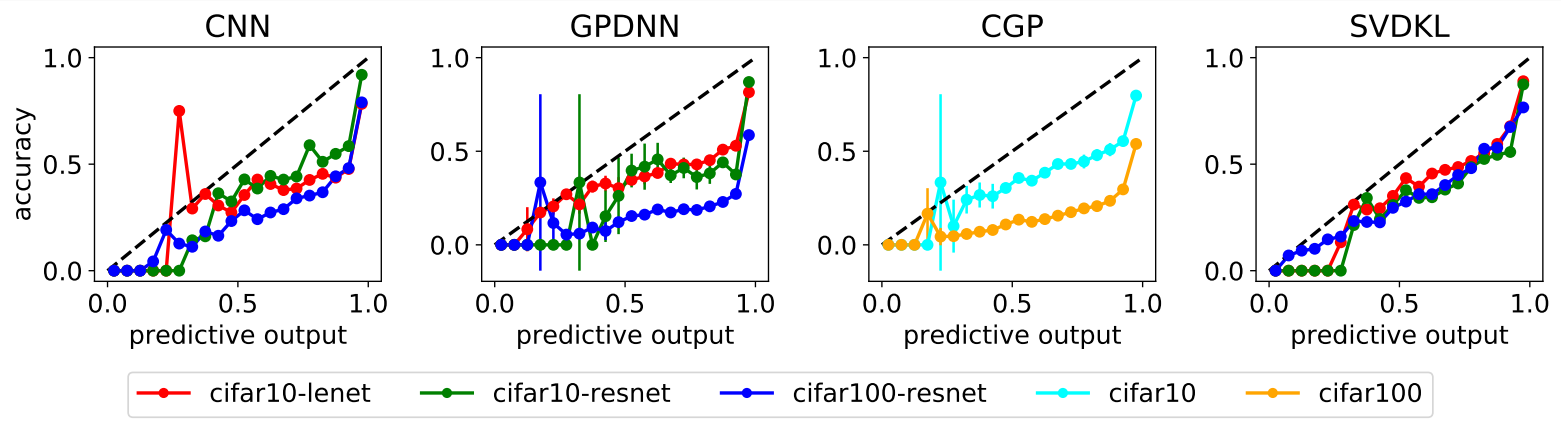


- Better quantification of uncertainty?? **NO!**

Tran et al., *AISTATS*, 2019

Existing Combinations of CNNs and GPs

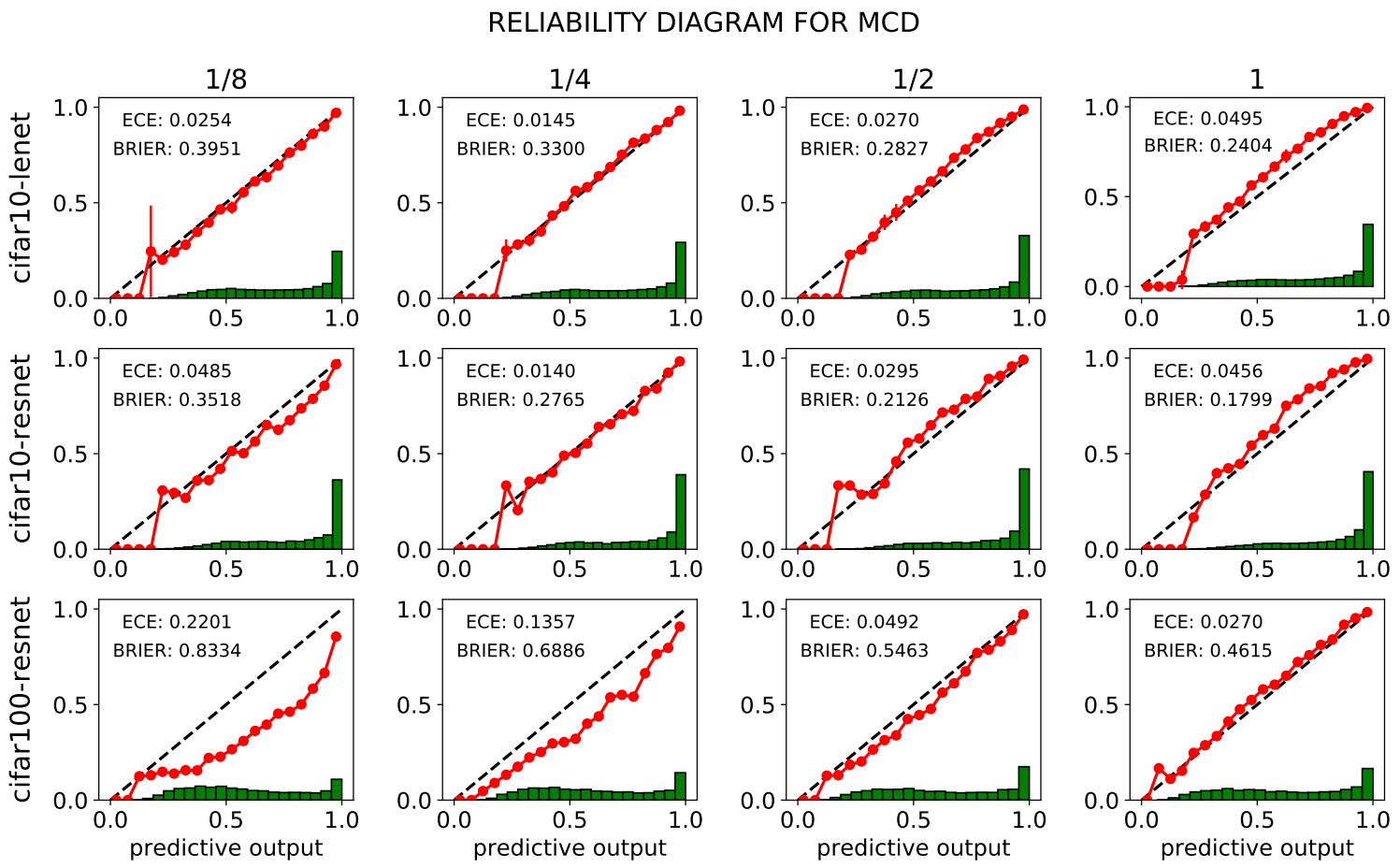
- Convolutional Neural Nets - CNN
- Hybrid GPs and DNNs - GPDNN
- Stochastic Variational Deep Kernel Learning - SVDKL
- Convolutional GP - CGP



Bradshaw et al., *arXiv*, 2017 – Wilson et al., *NIPS*, 2016 – van der Wilk et al., *NIPS*, 2017

Bayesian CNNs are calibrated

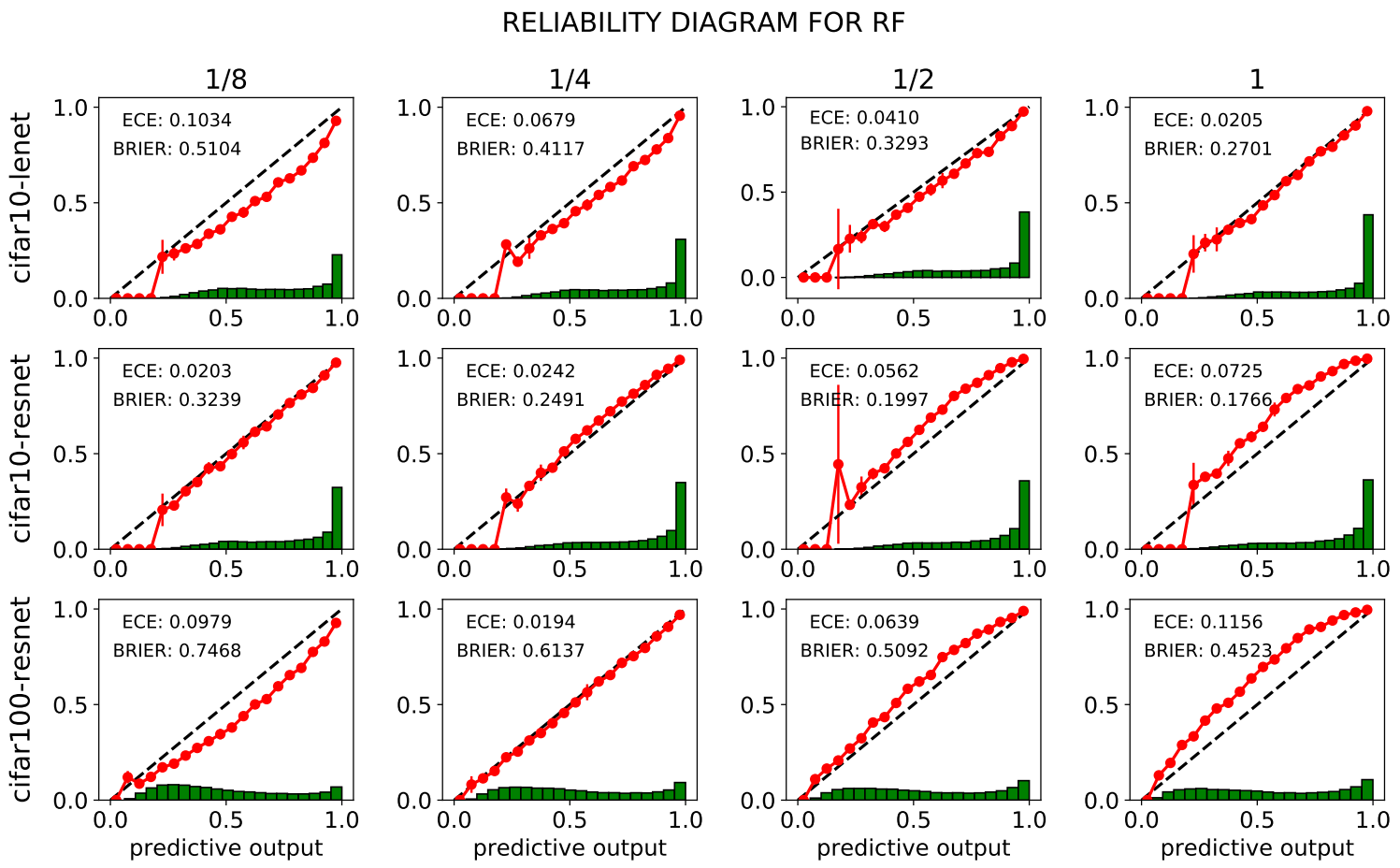
- Inferring parameters of convolutional filter recovers calibration
- Example with Monte Carlo Dropout



Tran et al., *AISTATS*, 2019

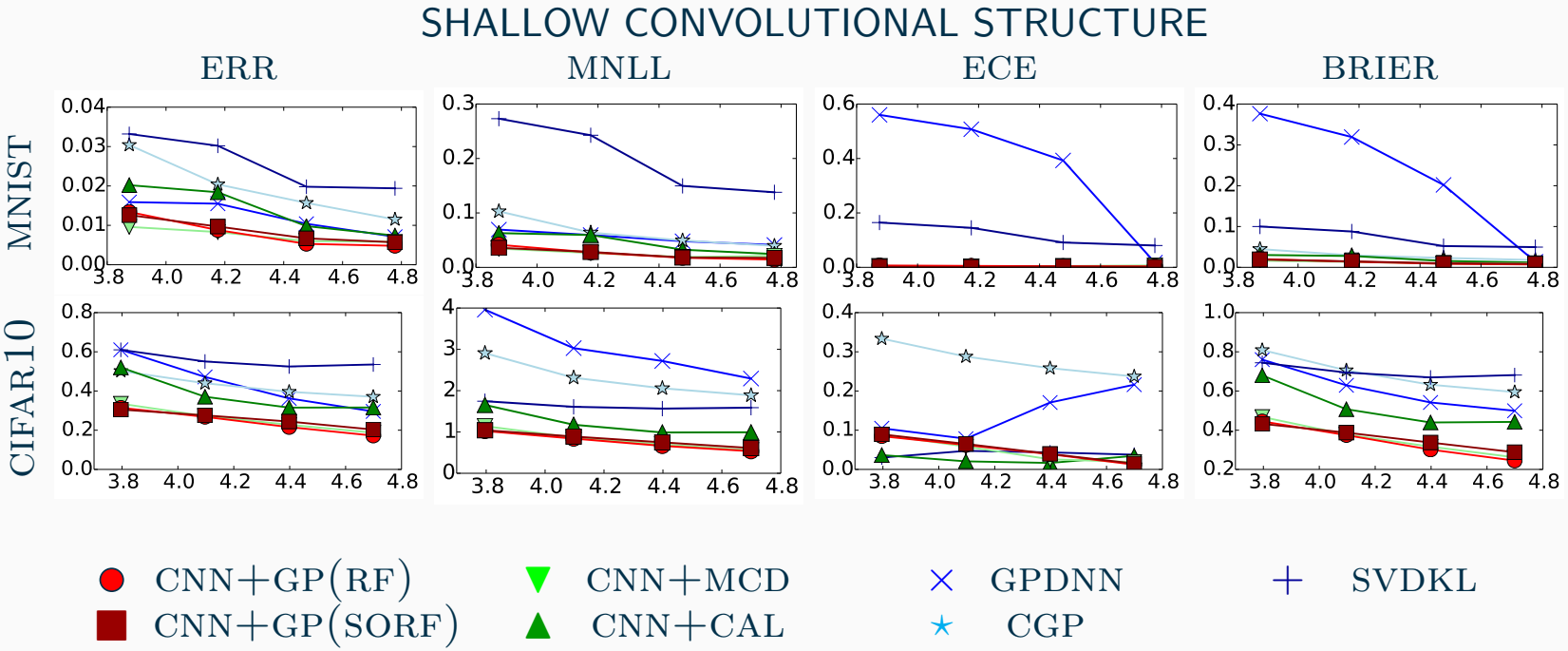
Bayesian CNNs with DGPs with Random Features

- We extended our work on Random Feature Expansions for DGPs to replace fully connected layers



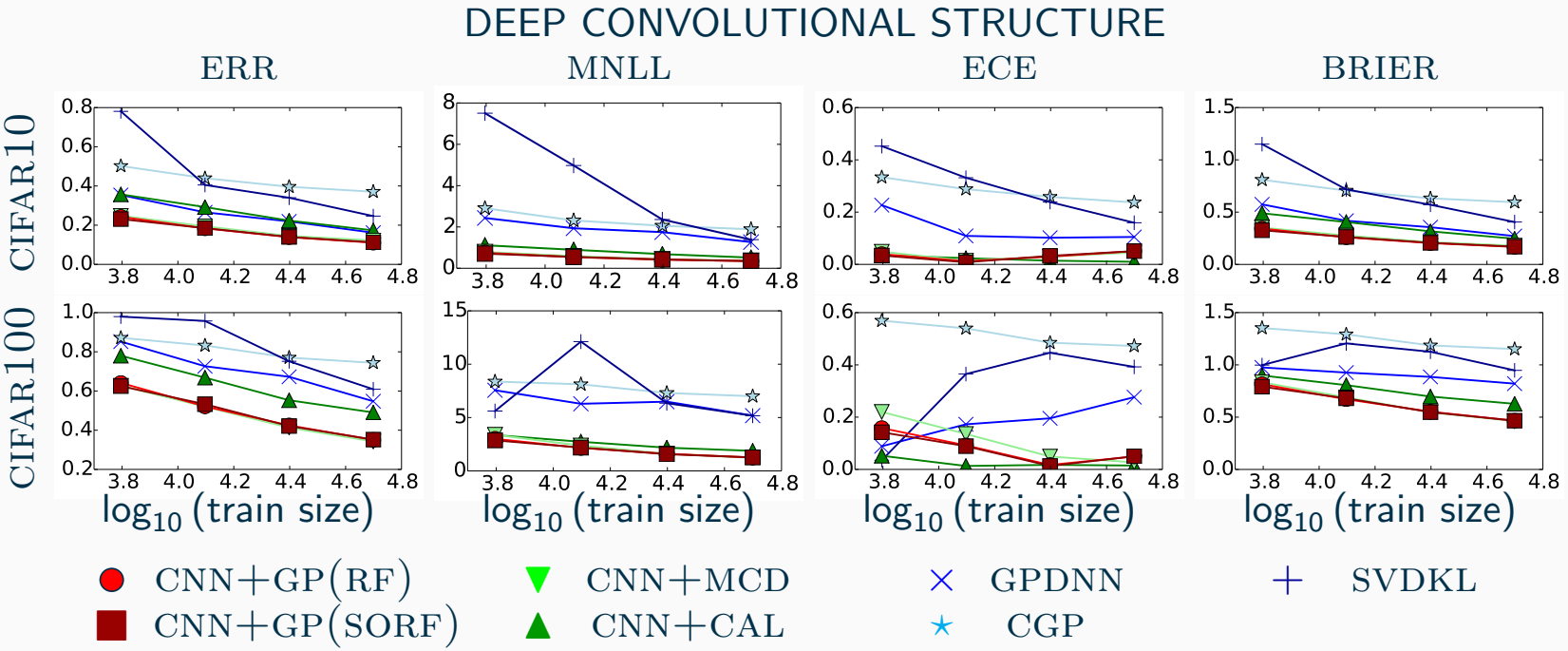
Tran et al., *AISTATS*, 2019

Comparison with competitors



Tran et al., *AISTATS*, 2019

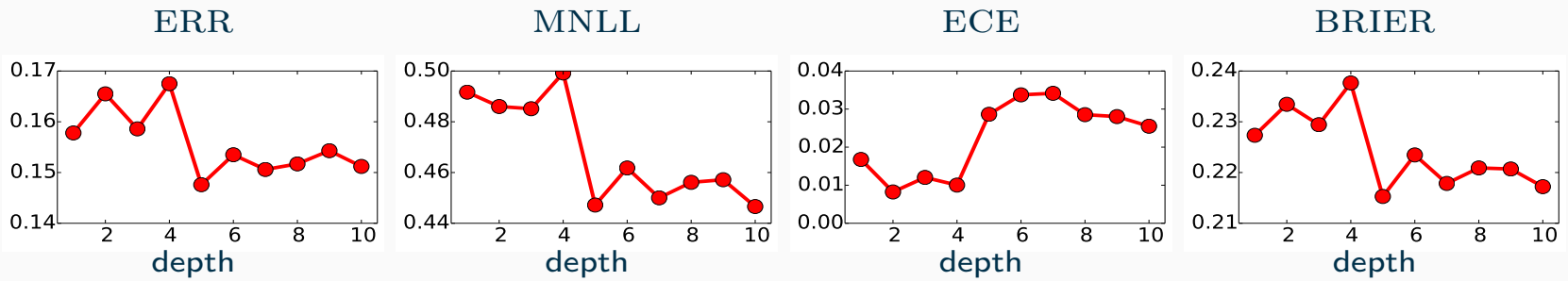
Comparison with competitors



Tran et al., *AISTATS*, 2019

Analysis of Depth of DGP

- Increasing depth of DGP slightly improves error rate...
- ... and slightly worsen calibration



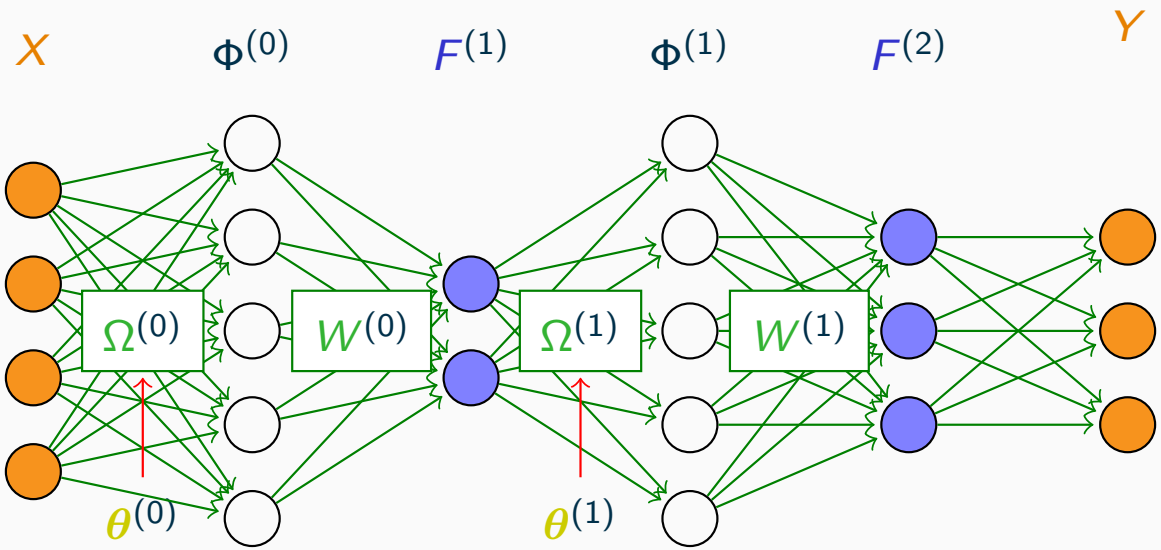
Tran et al., *AISTATS*, 2019

Deep Gaussian Processes

Recent Trends

Use Kernel Methods to Improve Bayesian Deep Learning

DGPs with random feature expansions



- Intermediate layers are based on multiplications with random iid normal matrices

$$\mathbf{\Phi}^{(l)} = g\left(\mathbf{F}^{(l)}\mathbf{\Omega}^{(l)}\right)$$

Use Kernel Methods to Improve Bayesian Deep Learning

- Replace $\mathbf{F}^{(l)}\boldsymbol{\Omega}^{(l)}$ with multiplication with pseudo-random matrices

$$\boldsymbol{\Omega}^{(l)} \approx \mathbf{H}\mathbf{D}_1^{(l)}\mathbf{H}\mathbf{D}_2^{(l)}\mathbf{H}\mathbf{D}_3^{(l)}$$

- This reduces:
  - time complexity from $\mathcal{O}(D^2)$ to $\mathcal{O}(D \log D)$
  - space complexity from $\mathcal{O}(D^2)$ to $\mathcal{O}(D)$

What about a Bayesian DNN where the posterior over matrices of weights is parameterized in this way?

Yu et al., *NIPS*, 2016 – Rossi et al., *arXiv*, 2019

Other Interesting Recent DGP Works

- Autoencoders Dai et al. *ICLR*, 2015 – Domingues et al., *Mach. Learn.*, 2018
- DGPs with constrained dynamics Lorenzi and Filippone, *ICML*, 2018
- Deep Convolutional GPs Blomqvist et al., *arXiv*, 2018
- Importance Weighted VI for DGPs Salimbeni et al., *ICML*, 2019

Conclusions

Conclusions

- Why DGPs?
  - Sensible priors for Bayesian deep learning
  - Improve our understanding of Bayesian deep learning

Conclusions

- Why DGPs?
  - Sensible priors for Bayesian deep learning
  - Improve our understanding of Bayesian deep learning
- Inference for DGPs is hard
  - Model approximations
  - Approximate inference
- Difficult to assess the impact of these approximations

Conclusions

- We are borrowing/mixing ideas from GPs and deep learning
  - Stochastic-based approximate inference
  - Low-rank process decompositions
  - Algebraic/computational tricks

Conclusions

- Combinations of GPs with CNNs slightly disappointing
  - Quantification of uncertainty not for free...
  - ... regularization of filters is necessary
  - Performance gains are small compared to plain CNNs

Acknowledgments

Thank you!