Imperial College London

PROWLER.io™
the principled AI company

Doubly Stochastic Variational Inference for Deep Gaussian Processes

Hugh Salimbeni
CSML 20th October 2017

Outline

- Part 1: Priors
- Part 2: Inference

Why Bayesian?

- Well-calibrated uncertainty
- Robust to overfitting
- Incorporate prior knowledge explicitly
- Cox's axioms etc

We assume that we:

- believe the prior (to some extent)
- can do inference (to some extent)

Example: Bayesian Neural Network

$$\mathbf{y} = \sigma(\sigma(\mathbf{X}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2)\mathbf{W}_3 + \mathbf{b}_3$$

$$\mathbf{W}_i, \mathbf{b}_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1)$$

When would we believe in this prior?

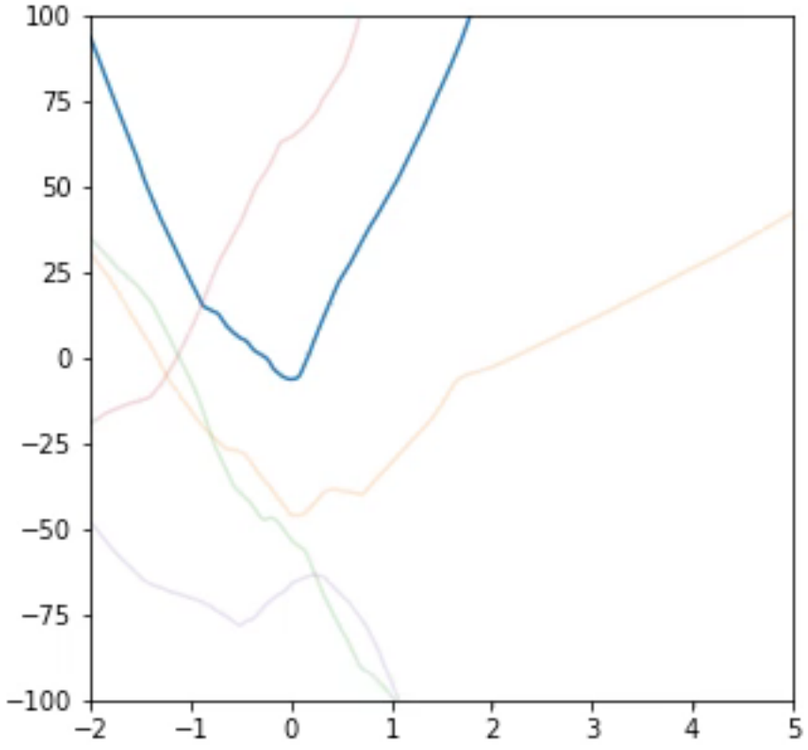
Do we even understand this prior?

Understanding the prior (1): take samples

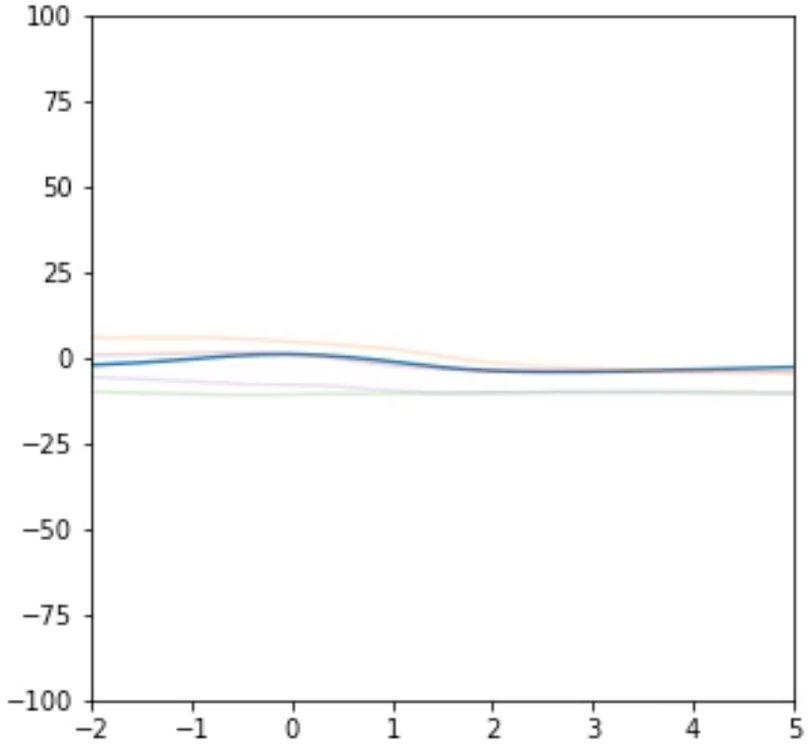
$$\mathbf{y} = \sigma(\sigma(\mathbf{X}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2)\mathbf{W}_3 + \mathbf{b}_3$$

$$\mathbf{W}_i, \mathbf{b}_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1)$$

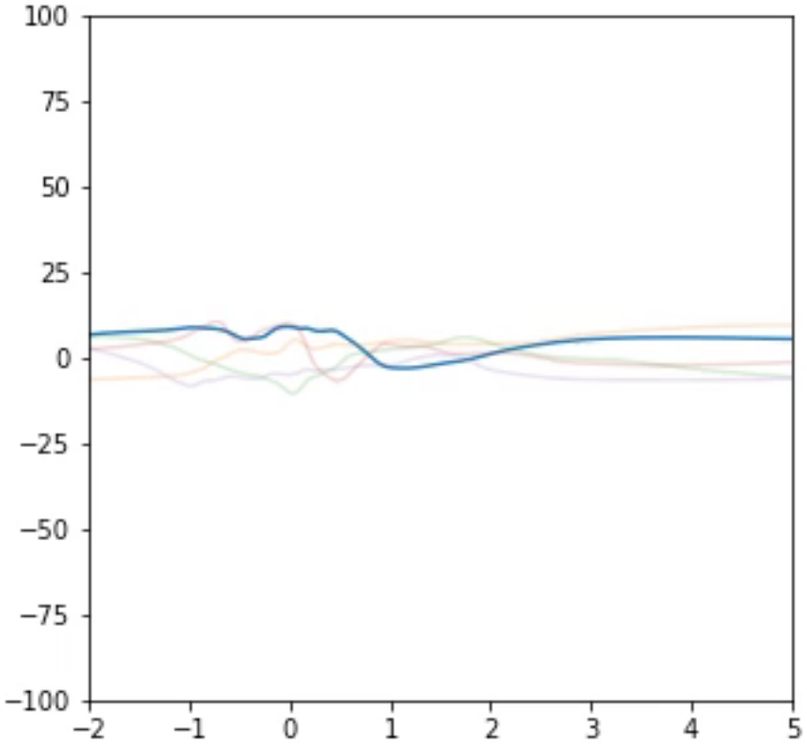
relu



sigmoid

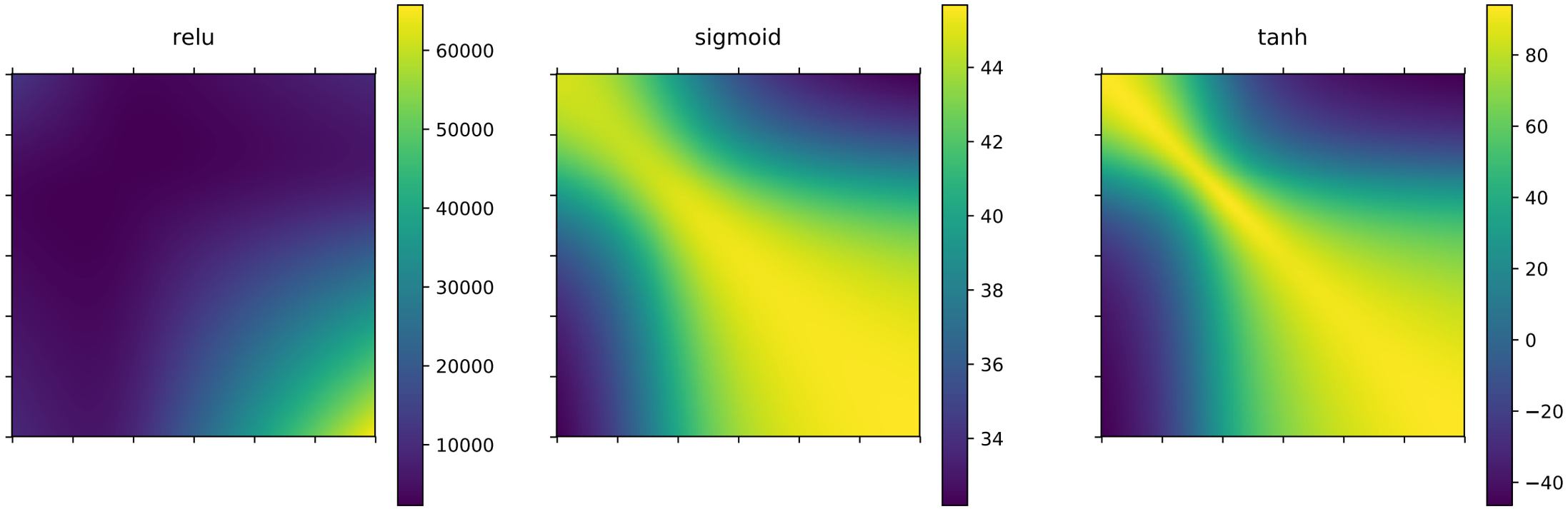


tanh



Understanding the prior (2): look at moments

- Mean (all zero)
- Covariance:



Higher moments??

Two possibilities

Option 1)

Interpret the prior in terms of things we understand

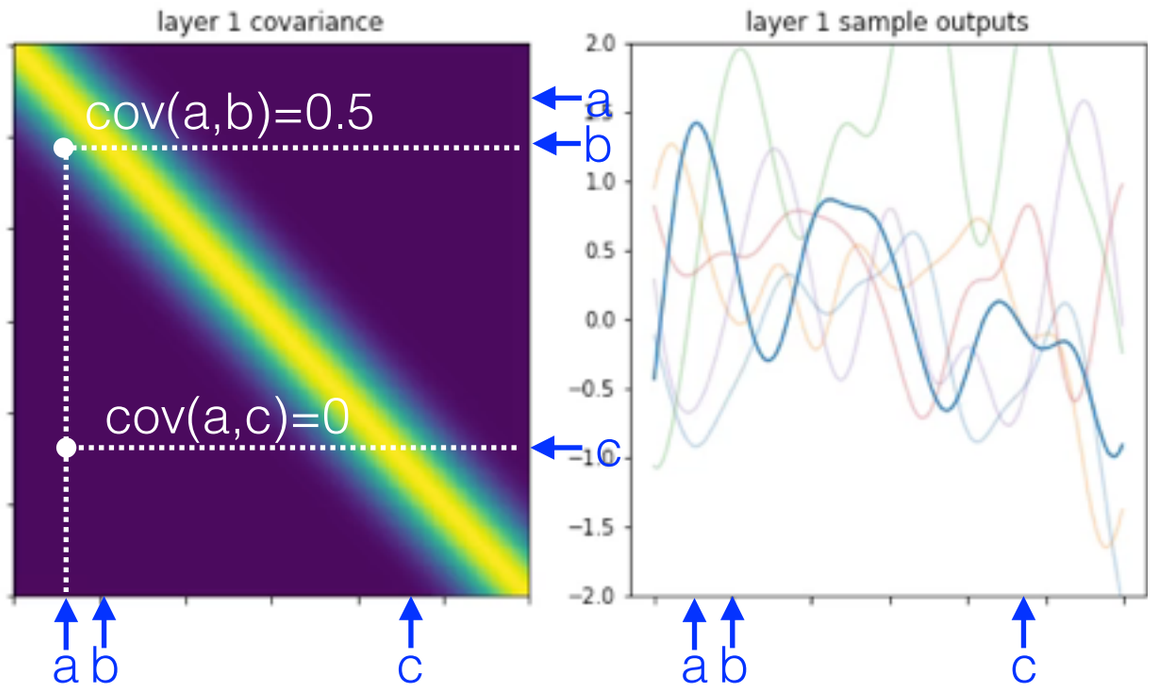
Option 2)

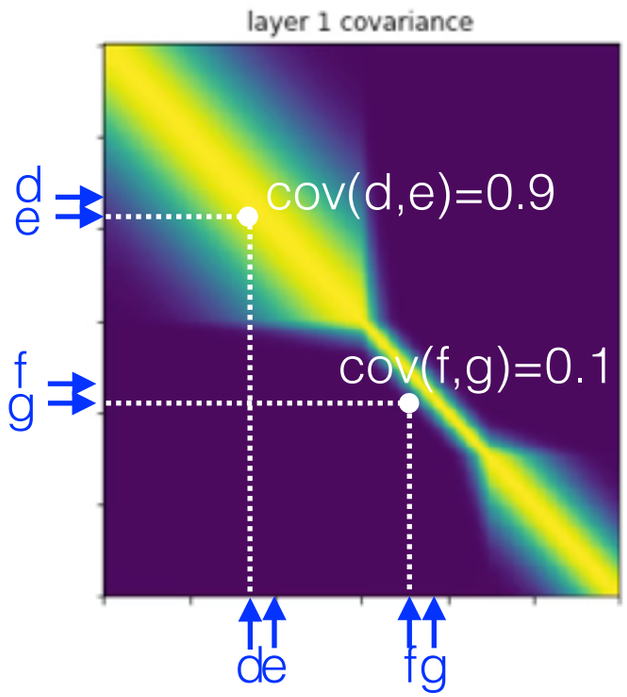
Define the prior directly over the things we understand

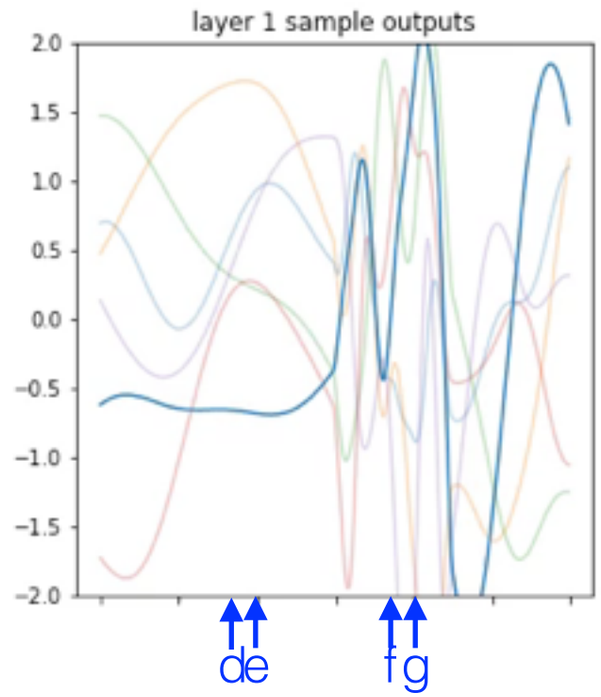
Gaussian processes

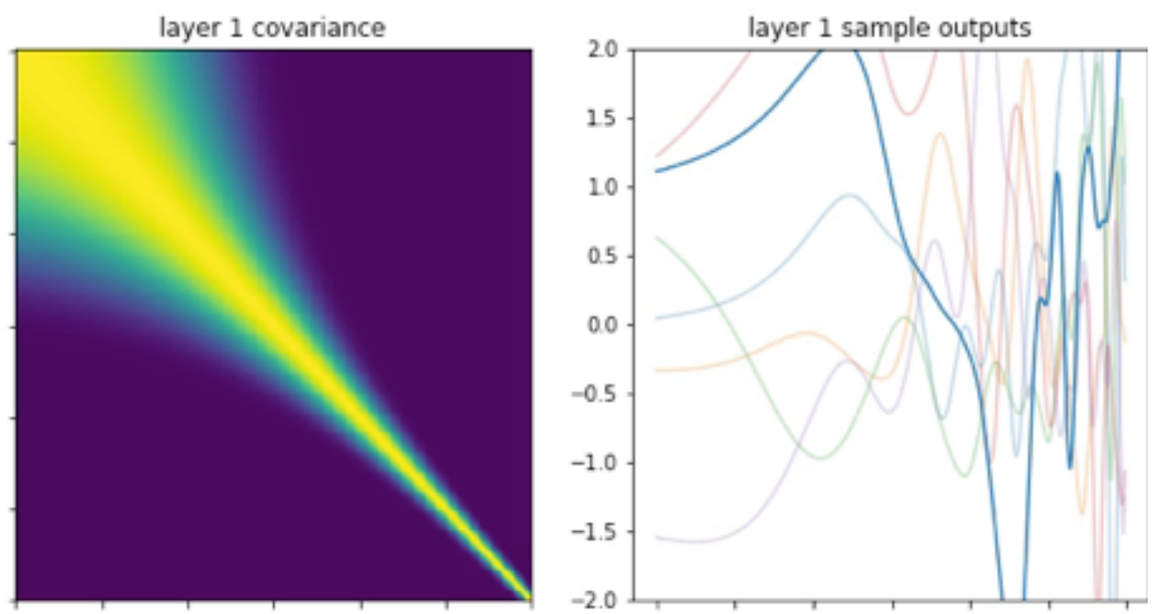
- We can use a Gaussian to model the outputs **y** directly
- Rather than specify a mapping (**X** to **y**), we instead specify a mean and covariance.
- Mean and covariance are interpretable quantities: we can understand this model.
- Also, we can do inference exactly (or nearly exactly)
- All we need is to choose a covariance…

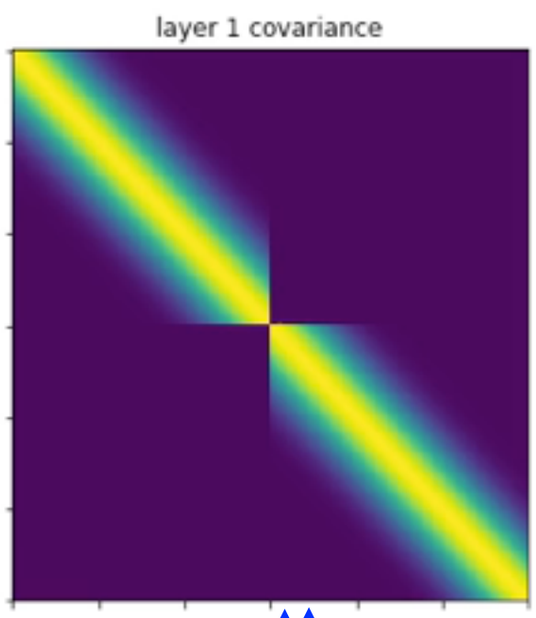
Example covariances

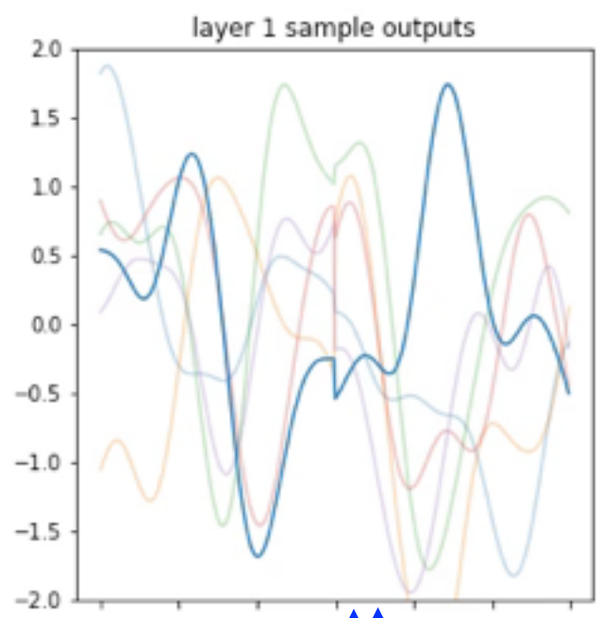




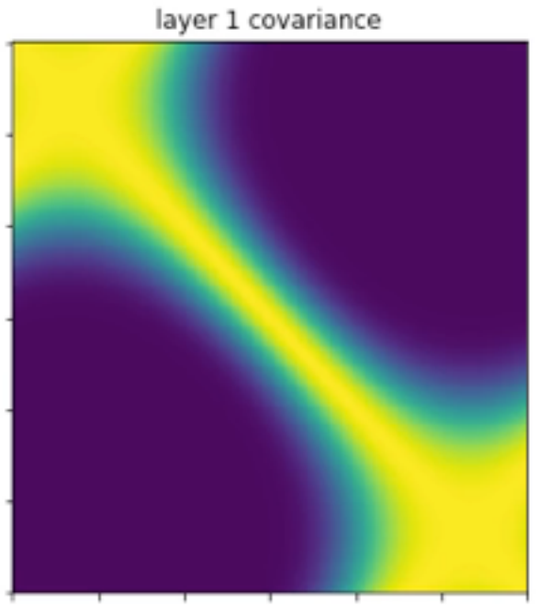


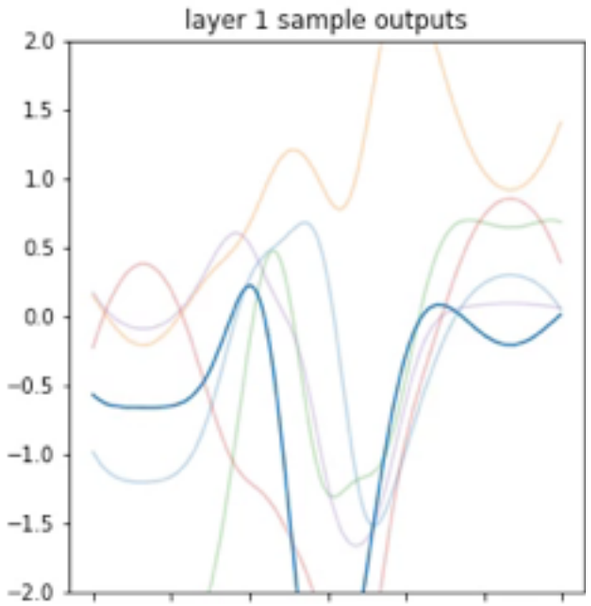


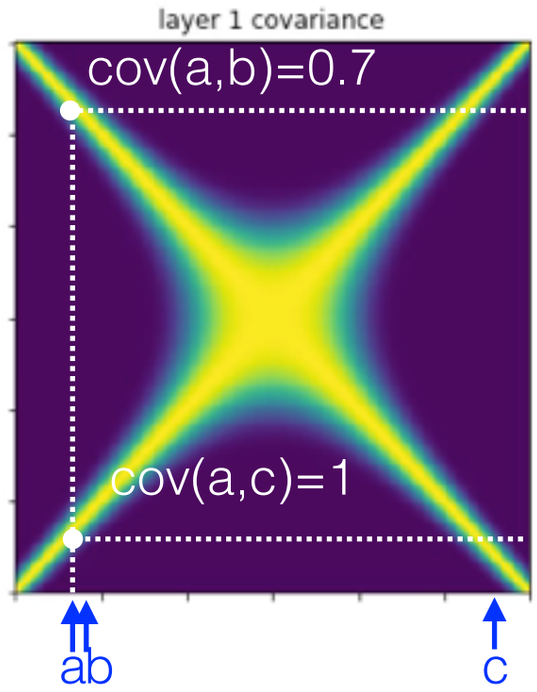


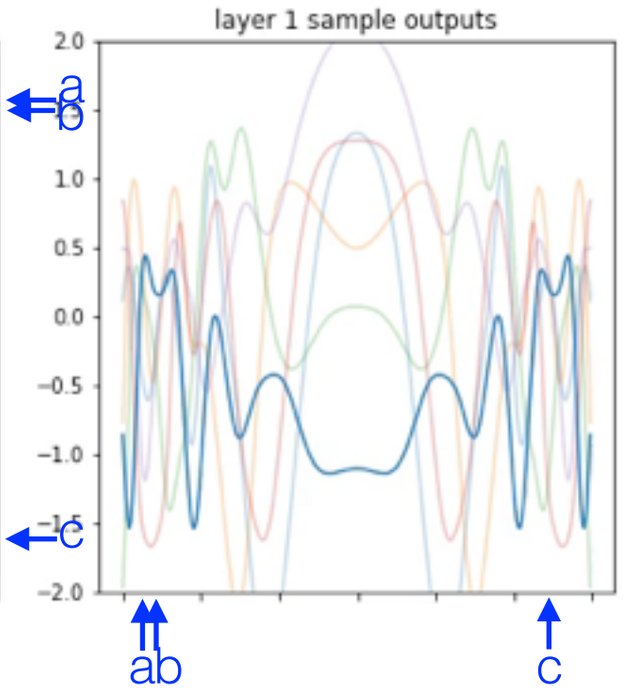


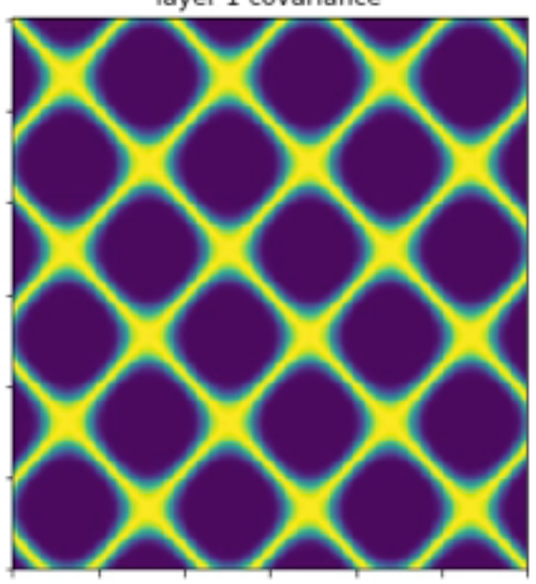
More covariances

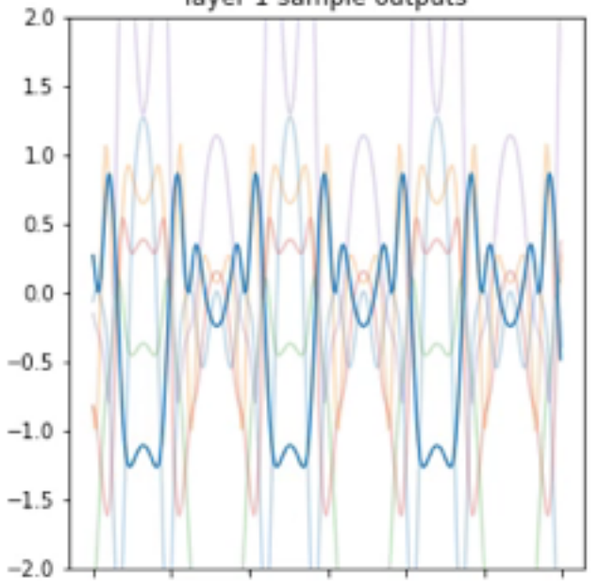


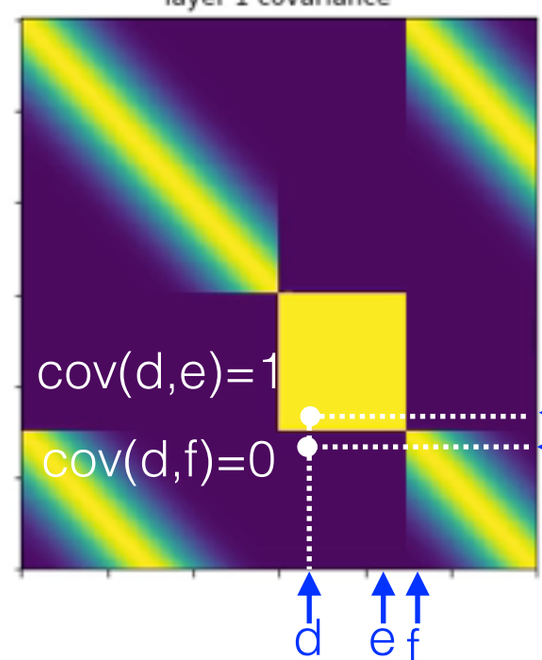


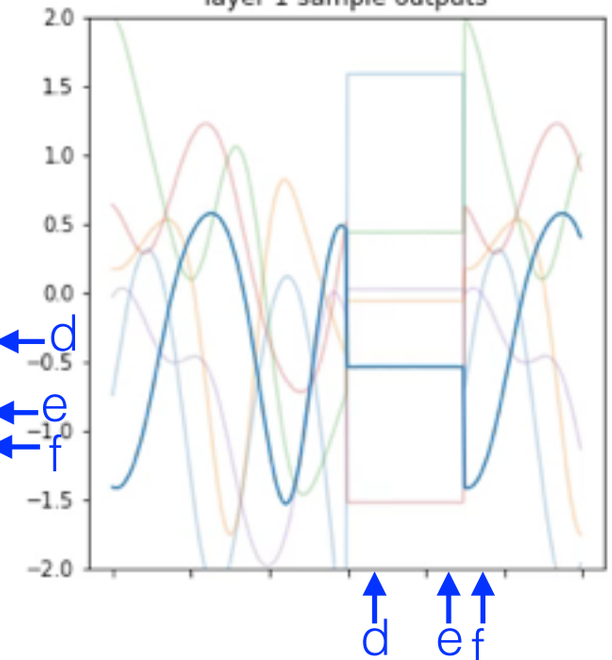










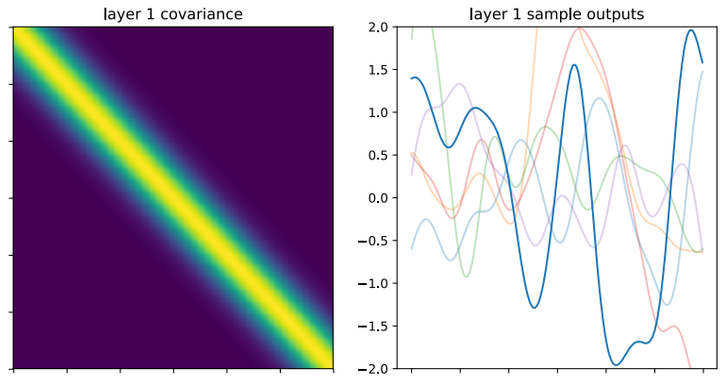


How to create covariances

Anything positive definite will do

- Most common: RBF (radial basis function):

$$k(x, x') = \exp\left(-\frac{(x-x')^2}{l^2}\right)$$



In practice

- It may be difficult to specify our prior beliefs as a covariance

- The standard covariance function toolkit is limited

- Simple kernels (e.g. RBF, Matern) give uninteresting models (smoothers), and cannot capture interesting structure.

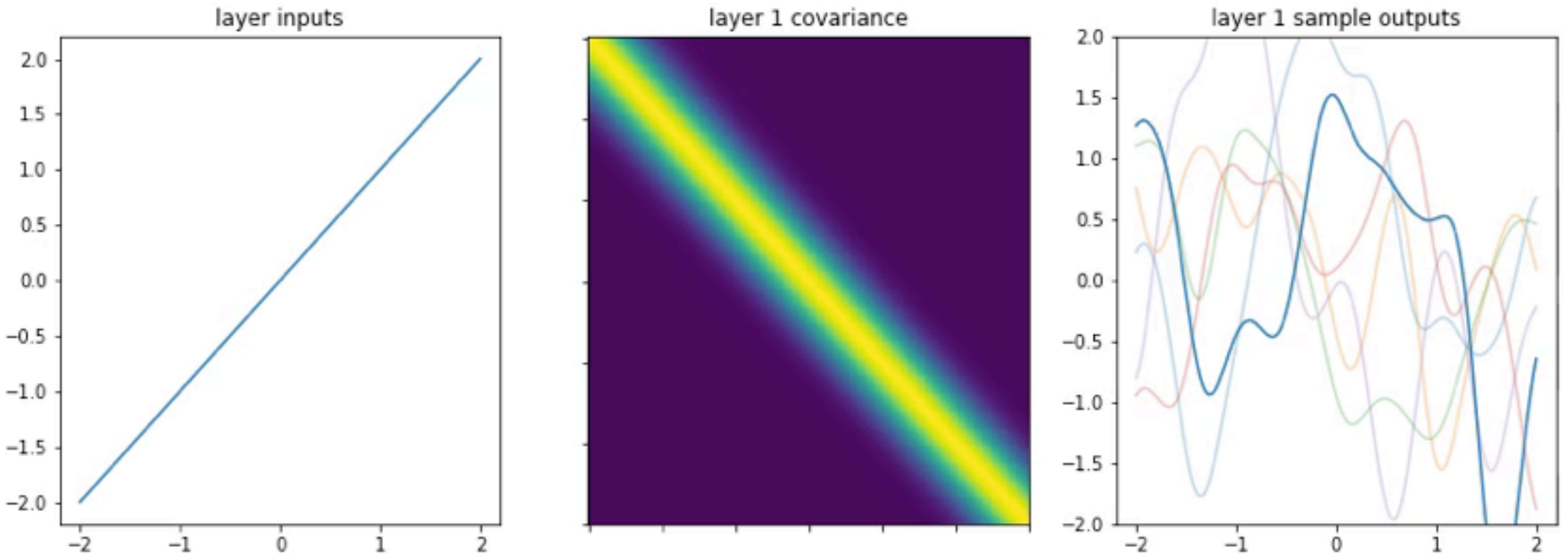
- We need a way of making interesting covariances

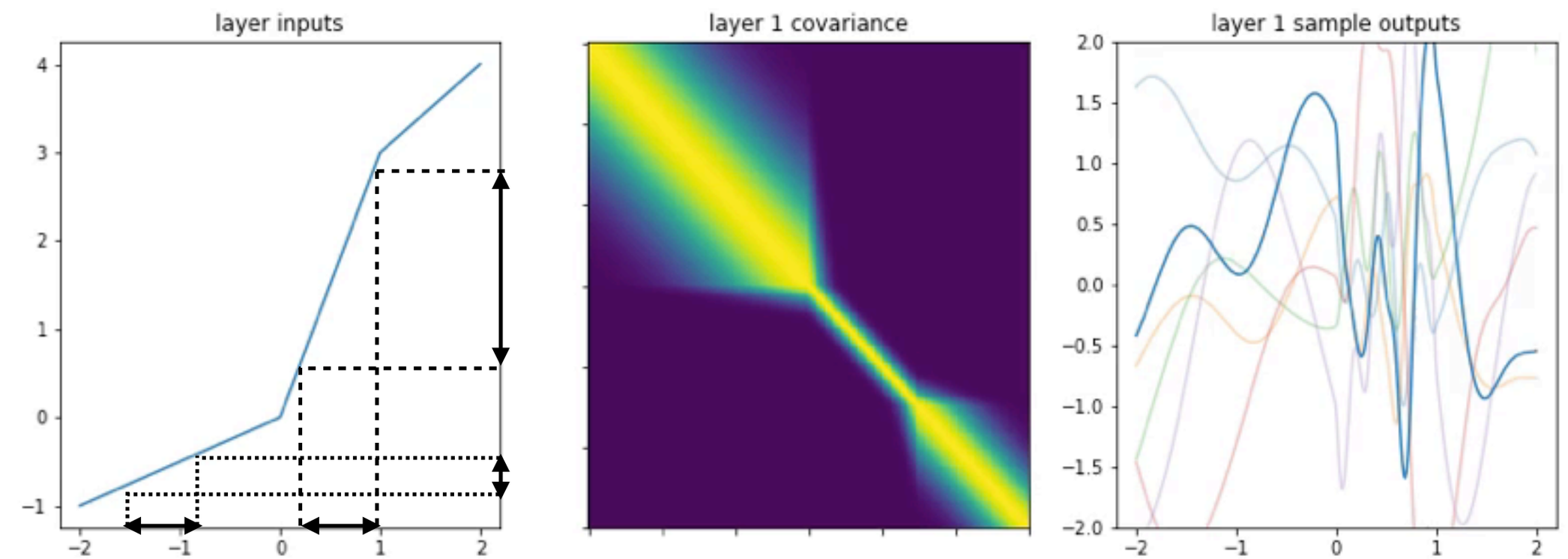
Key idea

- Simple covariance functions can produce complicated covariances when combined with input warping

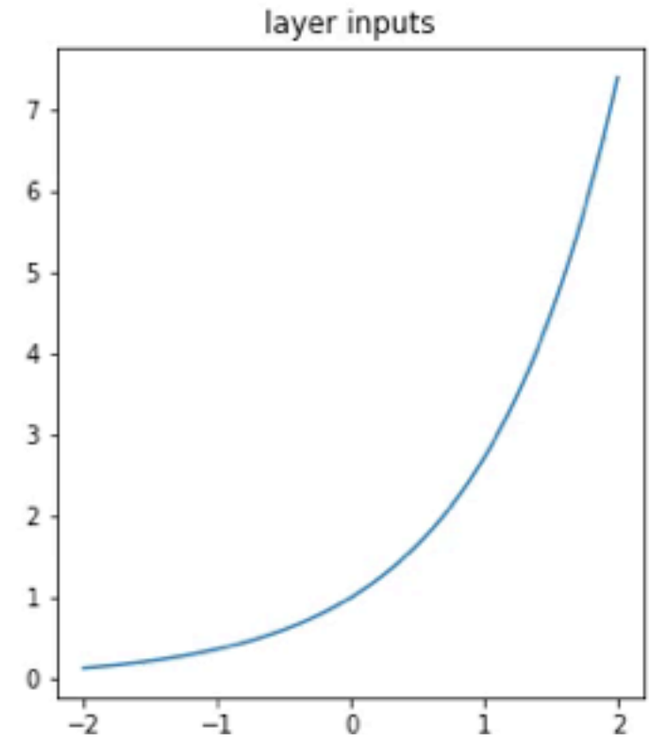
$$k_f(x, x') = k_{\mathrm{RBF}}(f(x), f(x'))) = \exp\left(-\frac{(f(x)-f(x'))^2}{l^2}\right)$$

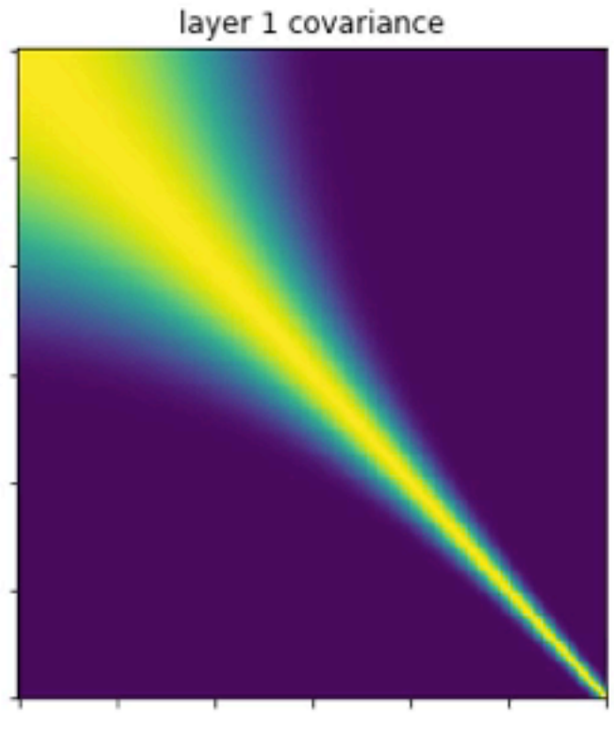
How the covariances were created

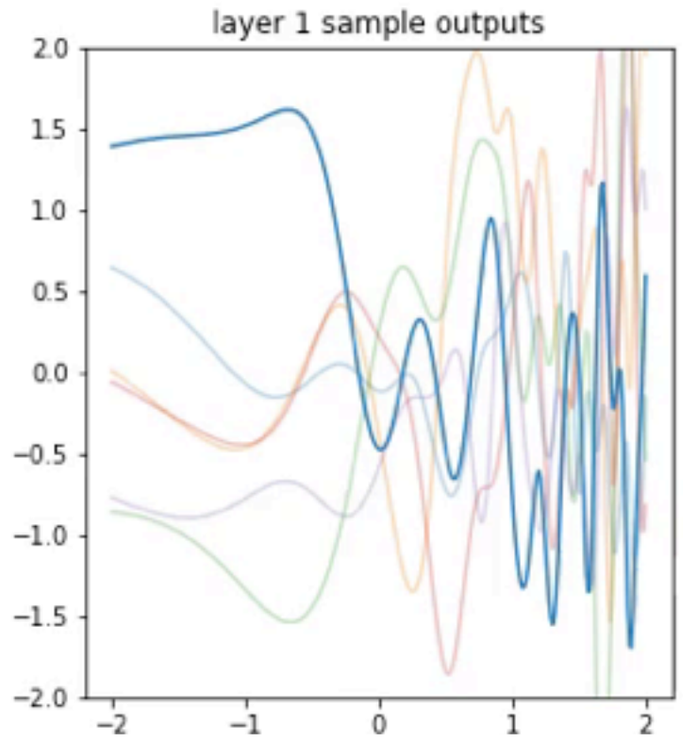


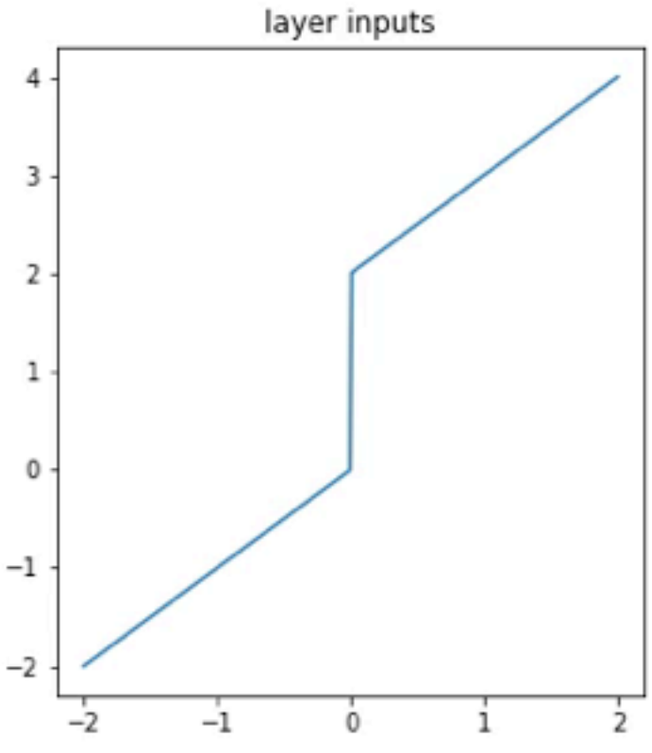


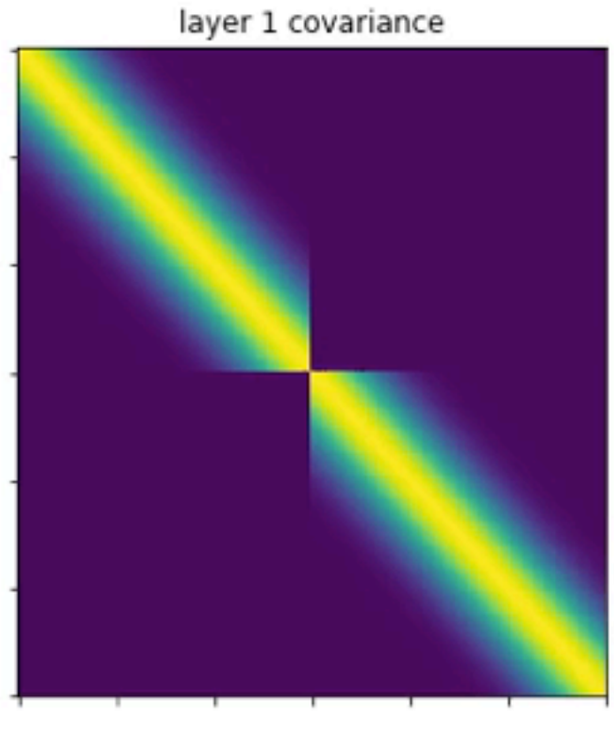
How the covariances were created

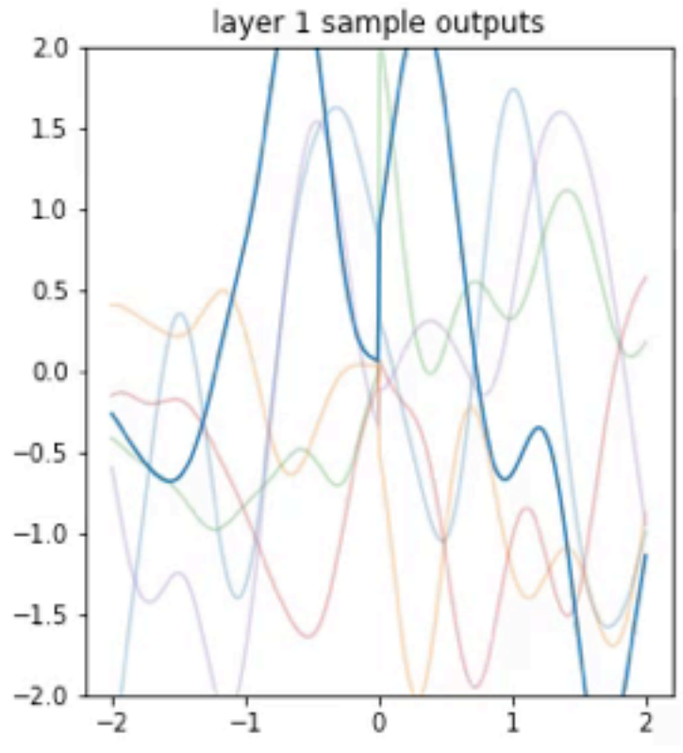




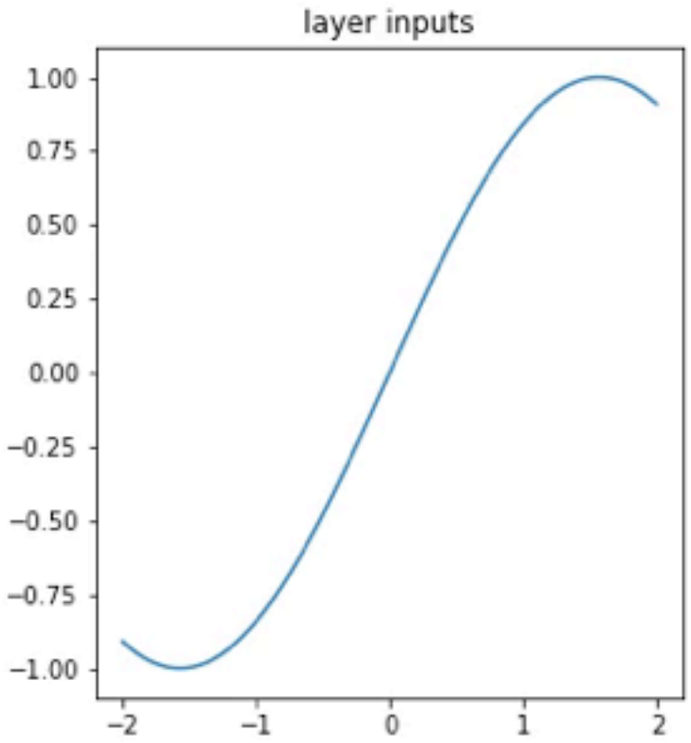


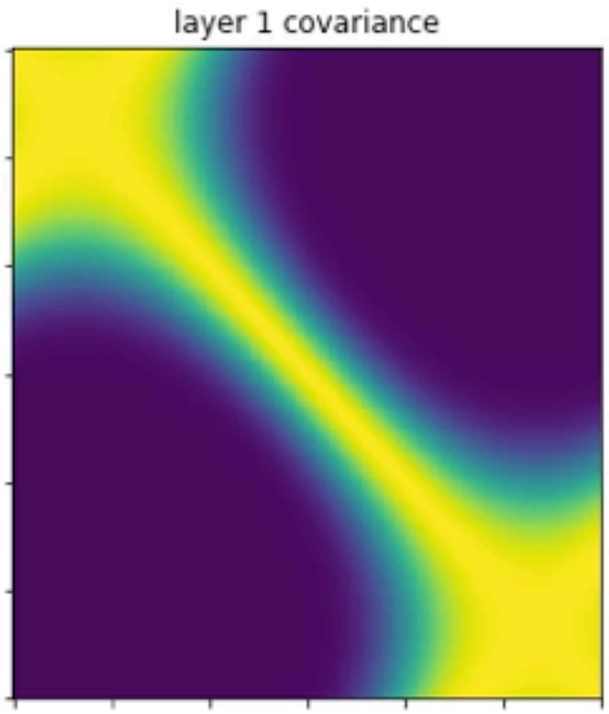


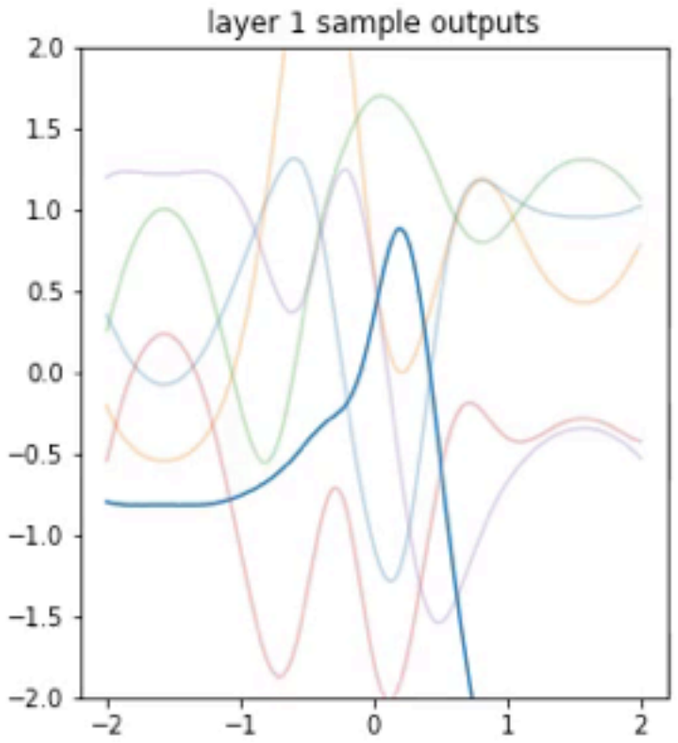


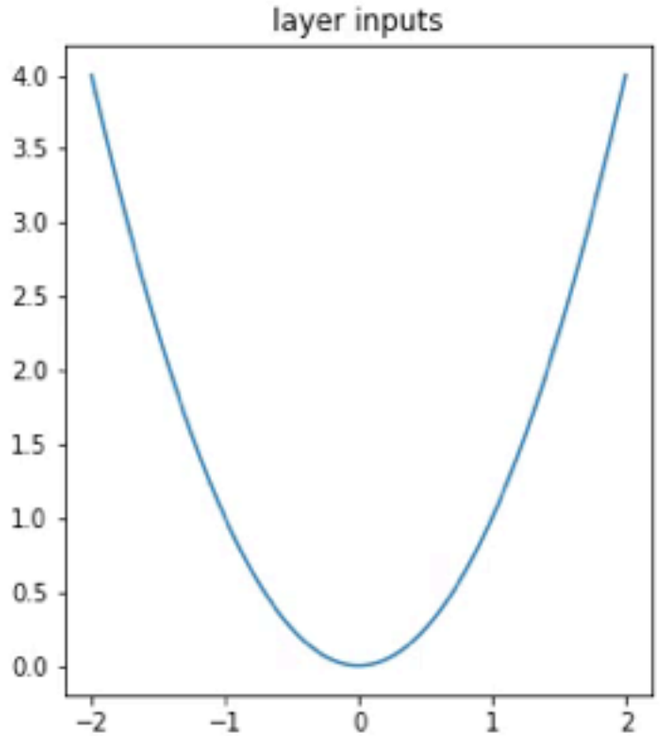


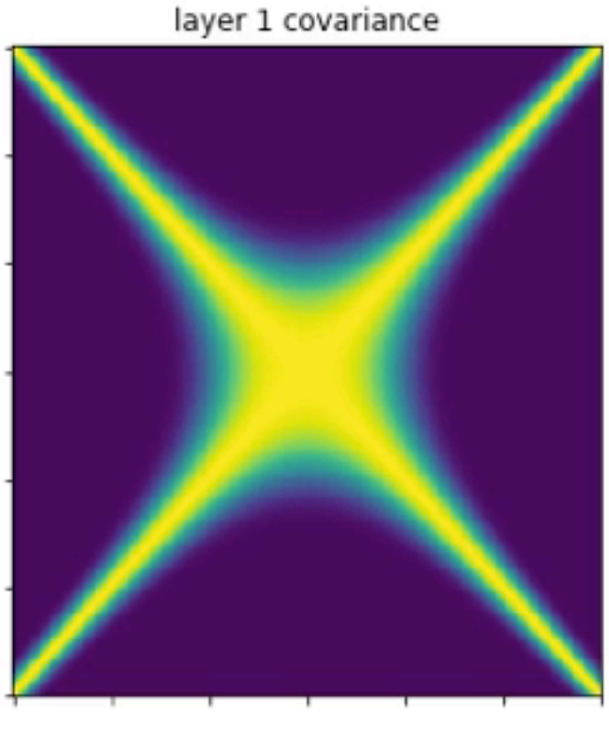
How the covariances were created

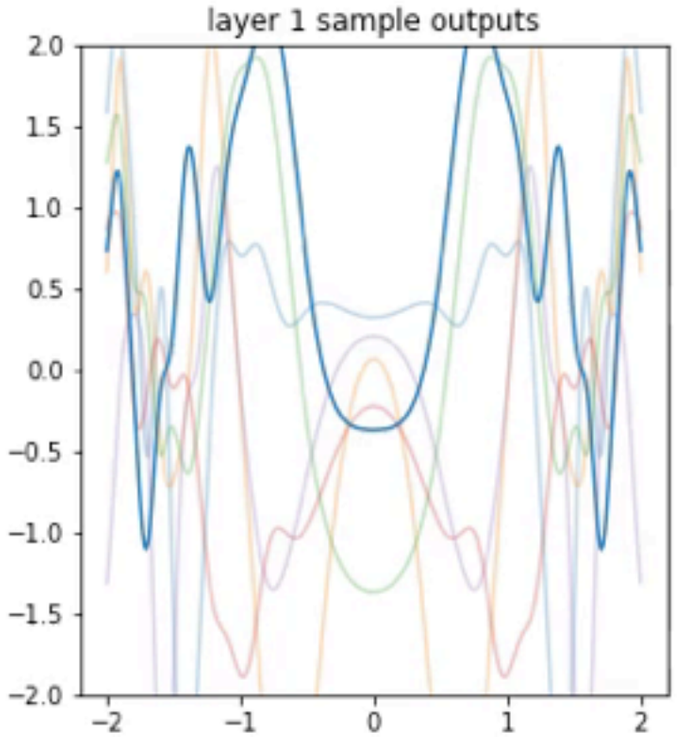




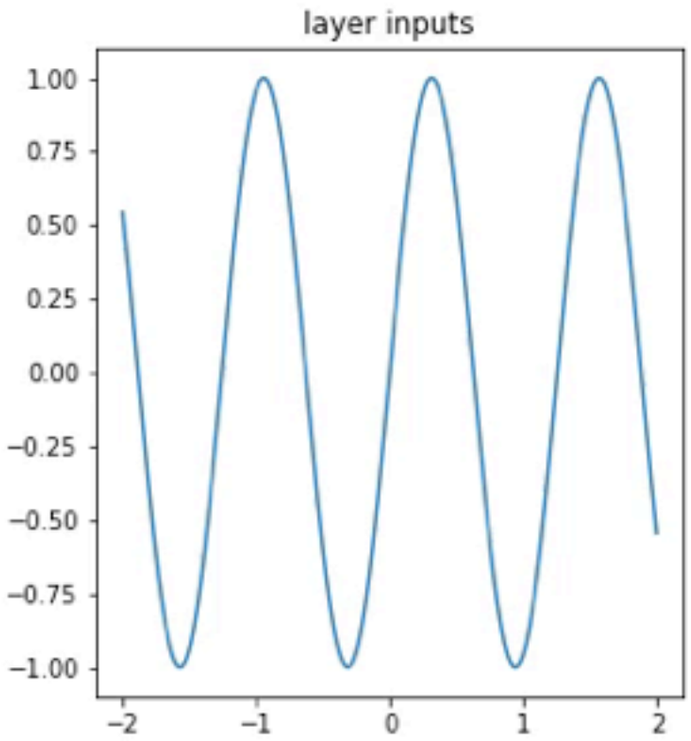


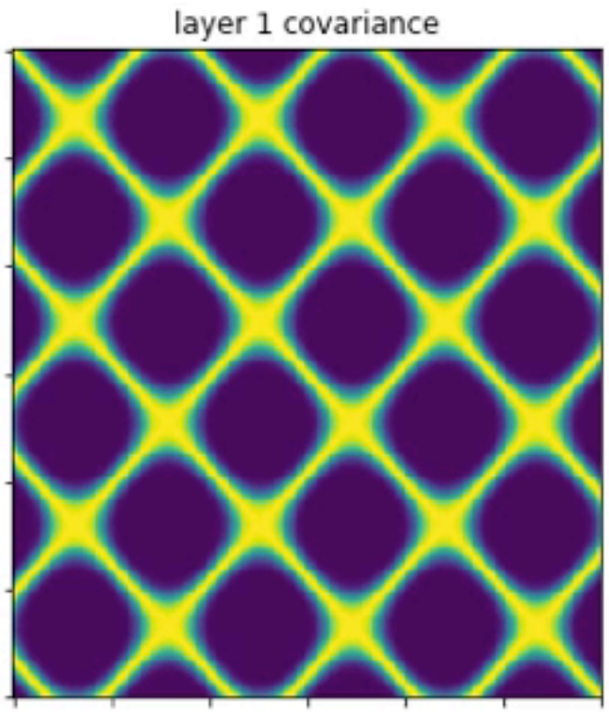


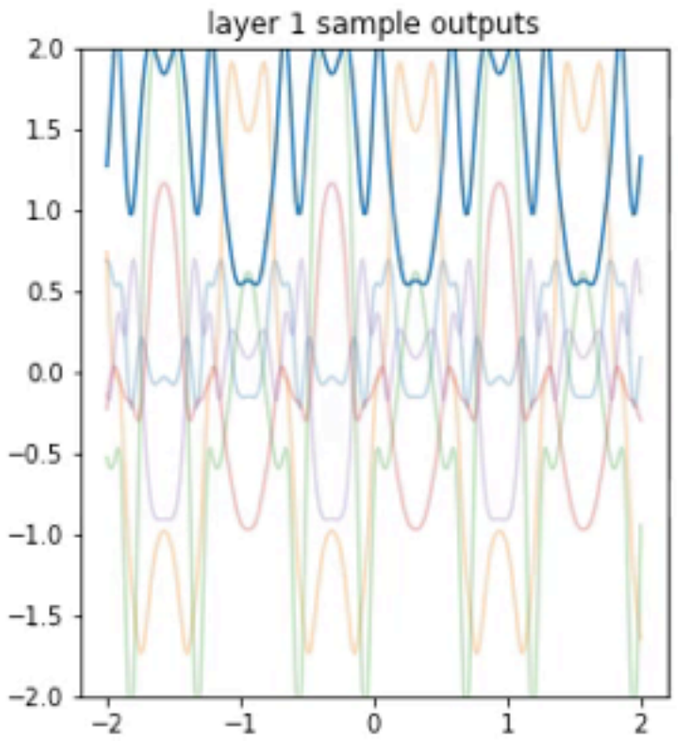


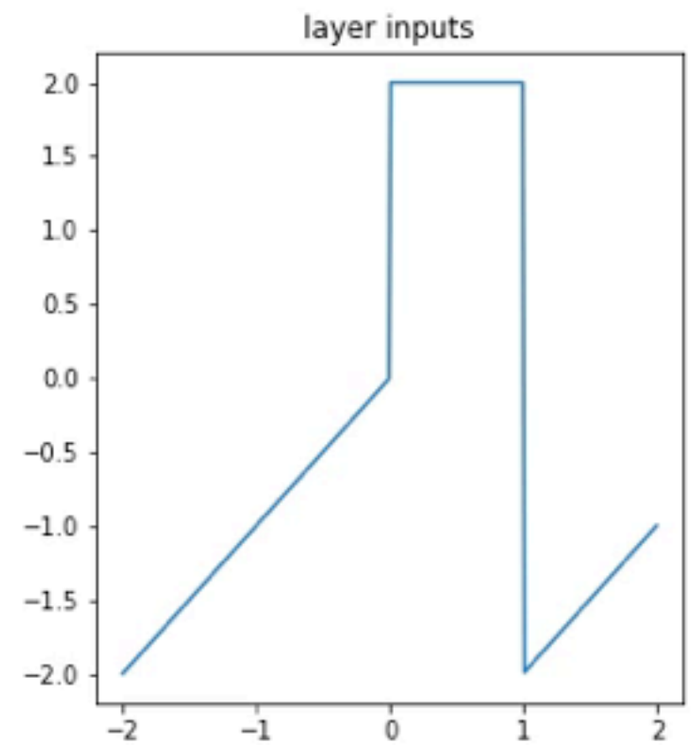


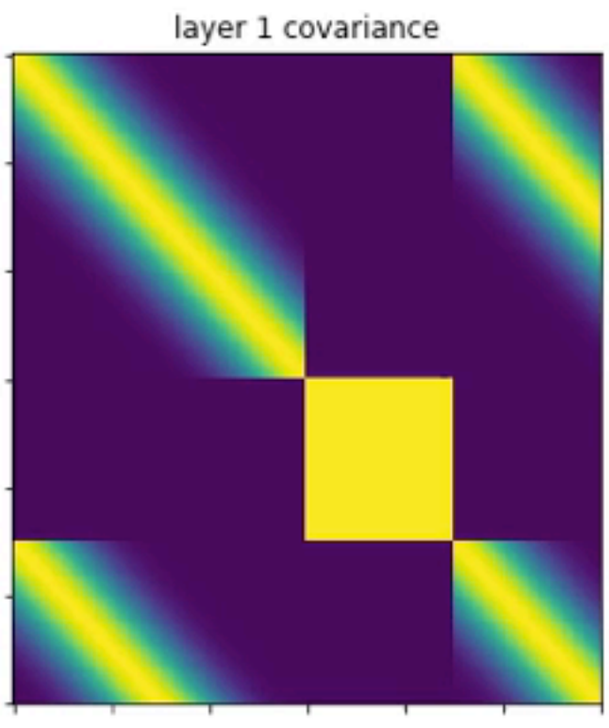
How the covariances were created

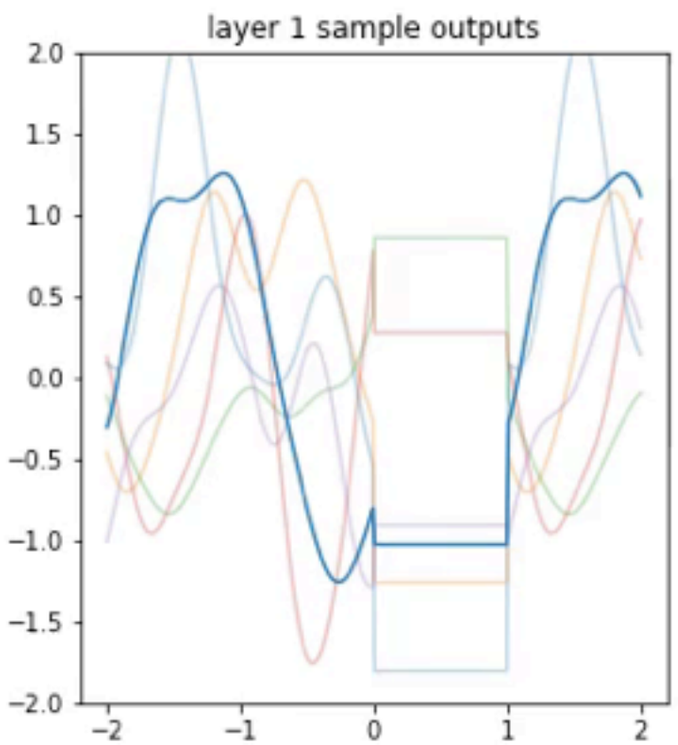




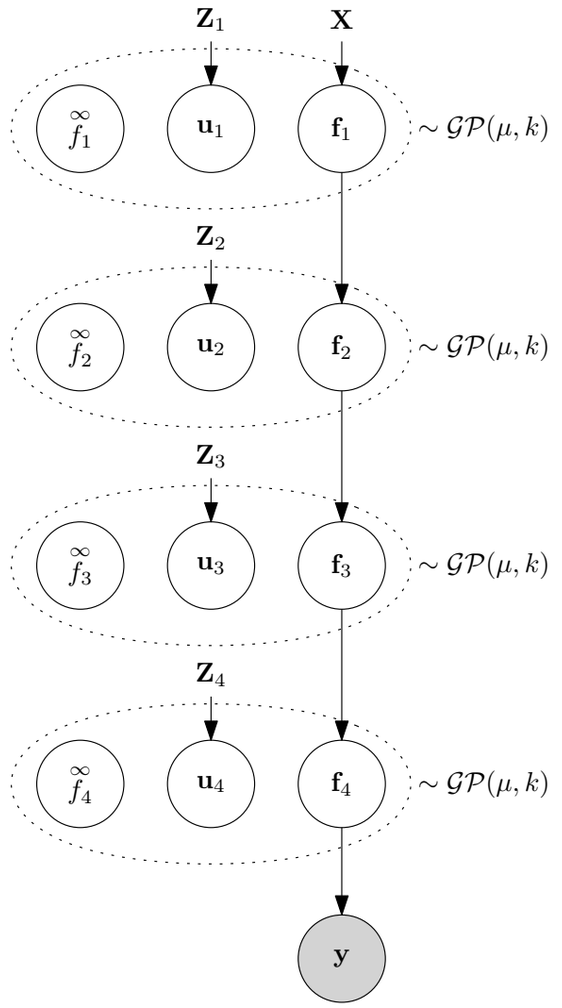








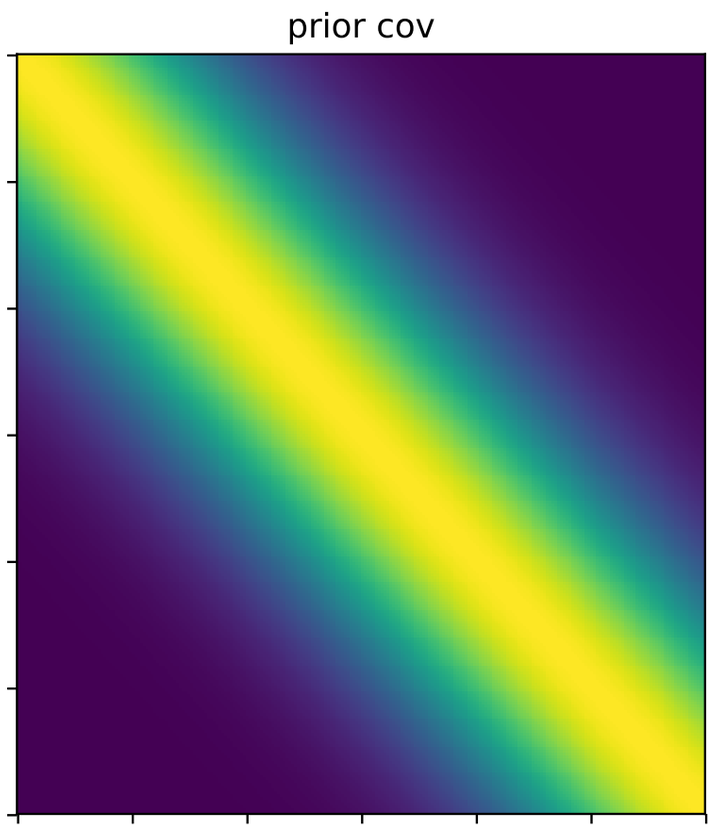
Why Deep?

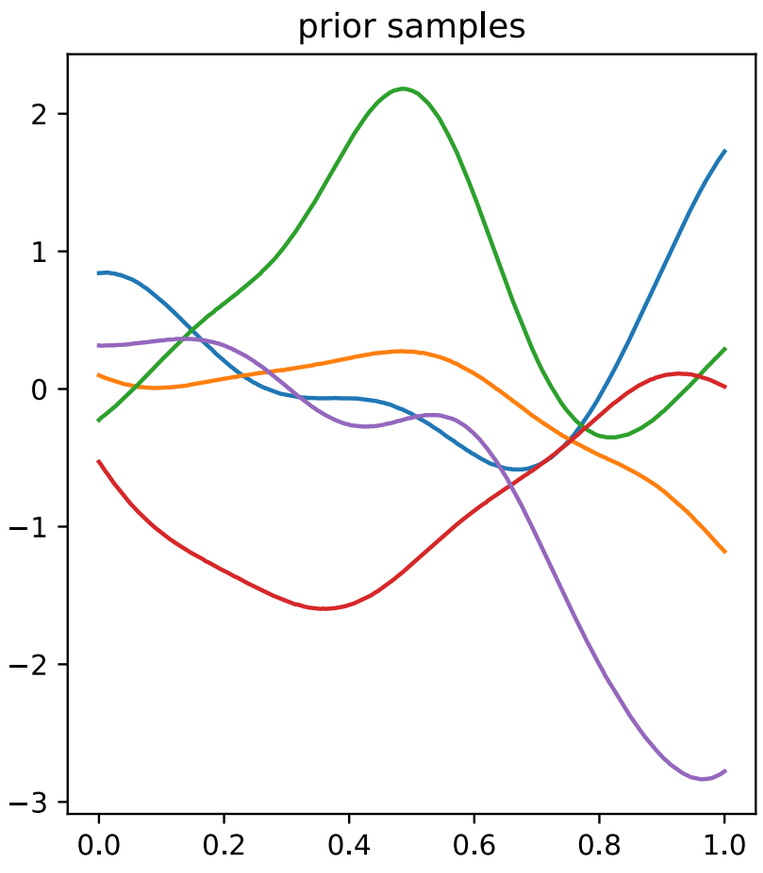


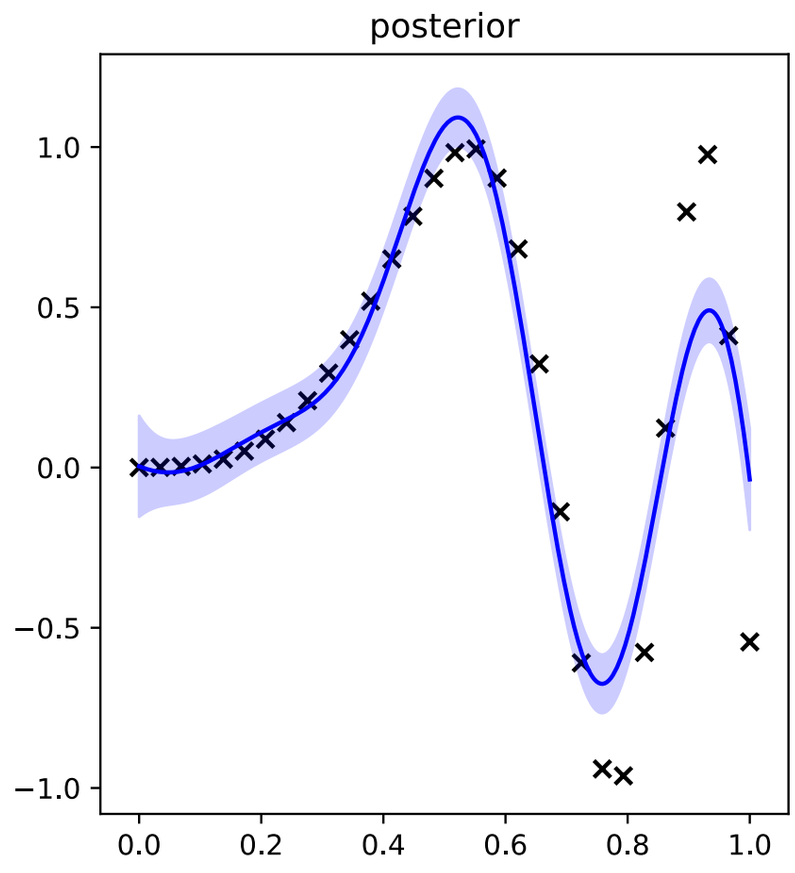
A GP is not the answer to all problems:

- We might not know the covariance
- Specifying a wrong covariance leads to bad results
- Using a GP we might inadvertently use a prior that is too concentrated
- The data might have structure that we cannot easily express as a covariance

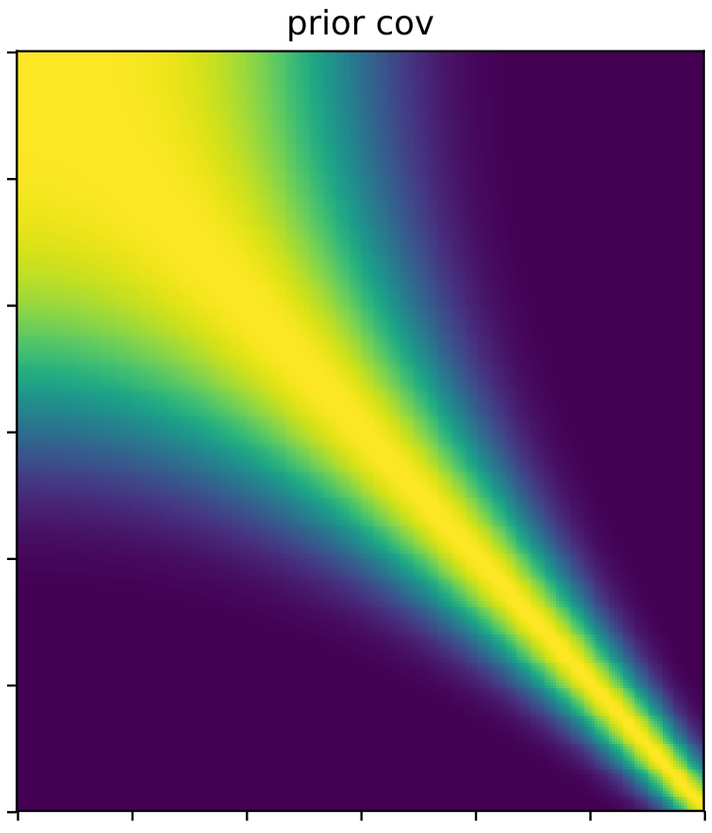
Example: too simple covariance

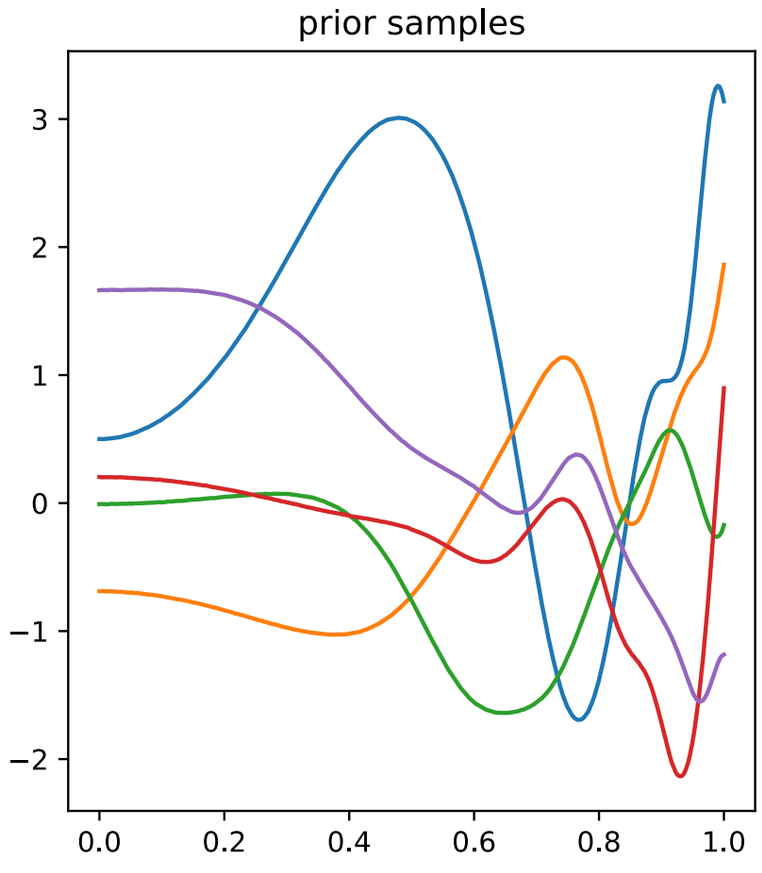


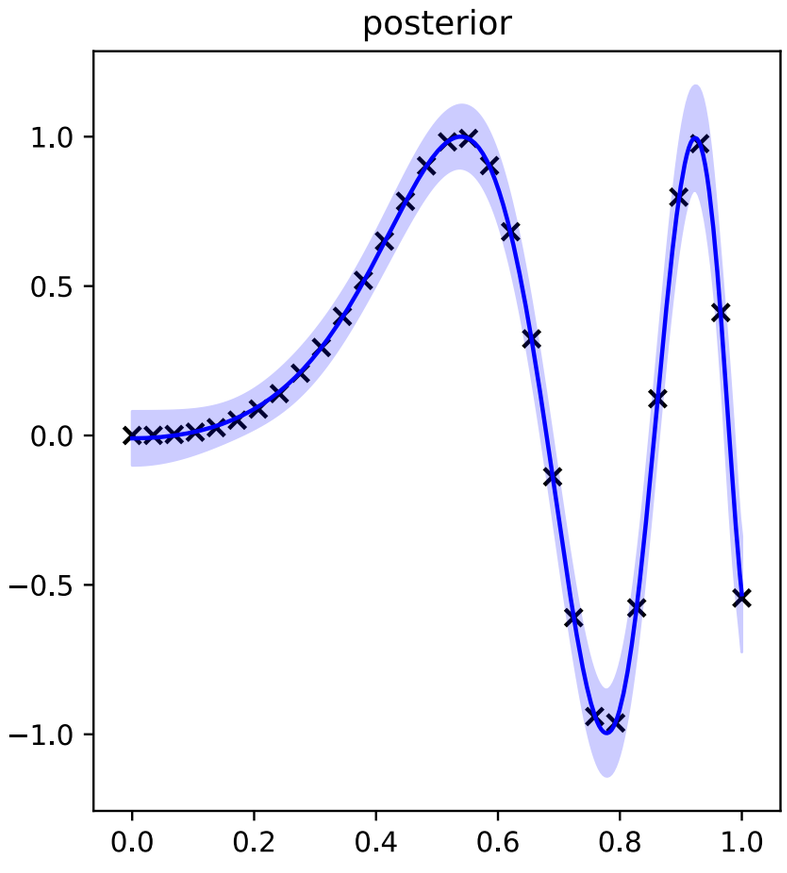




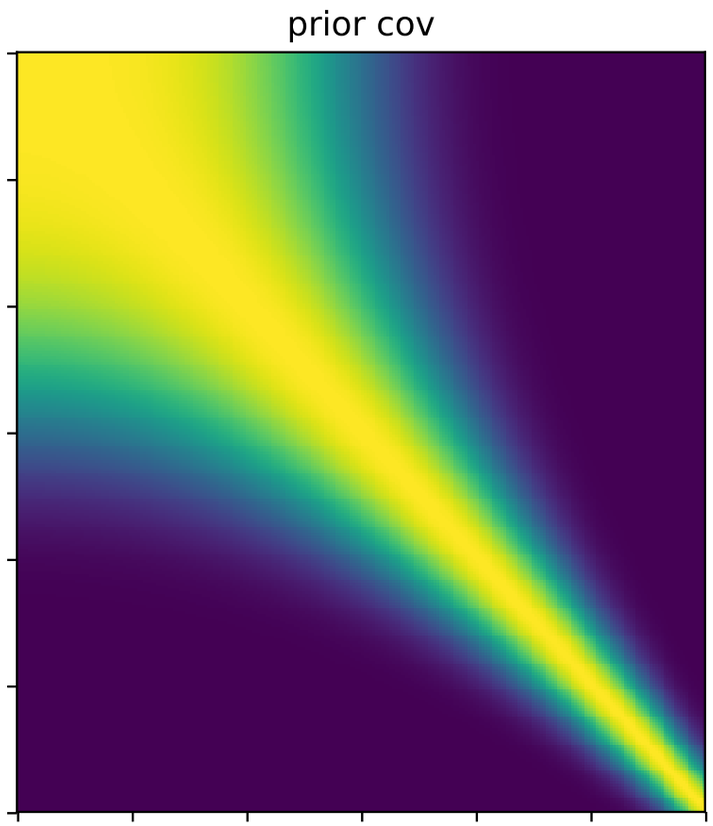
Example: correctly specified covariance

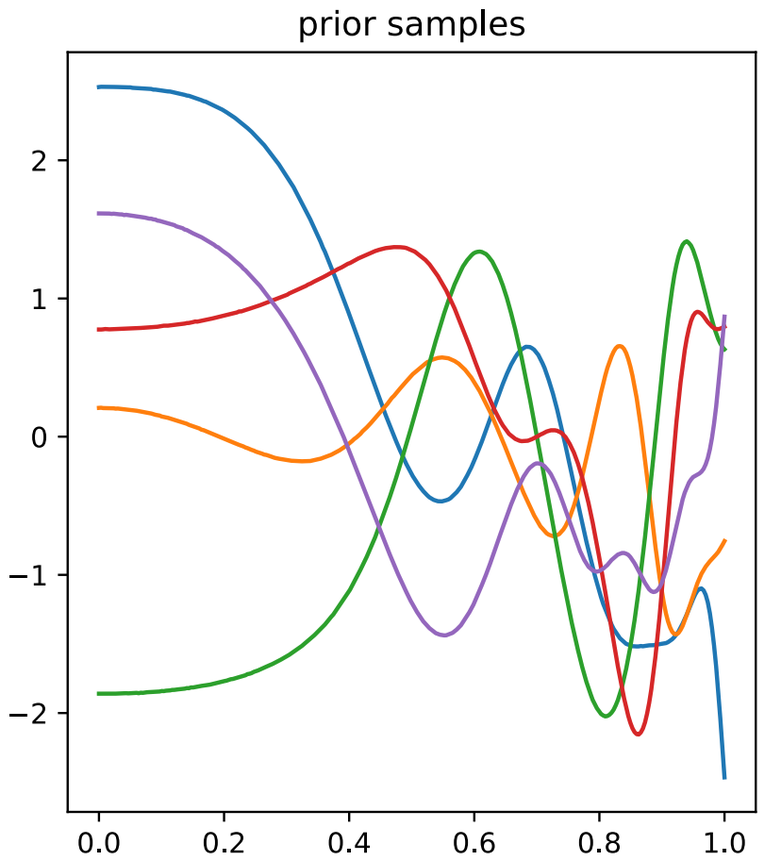


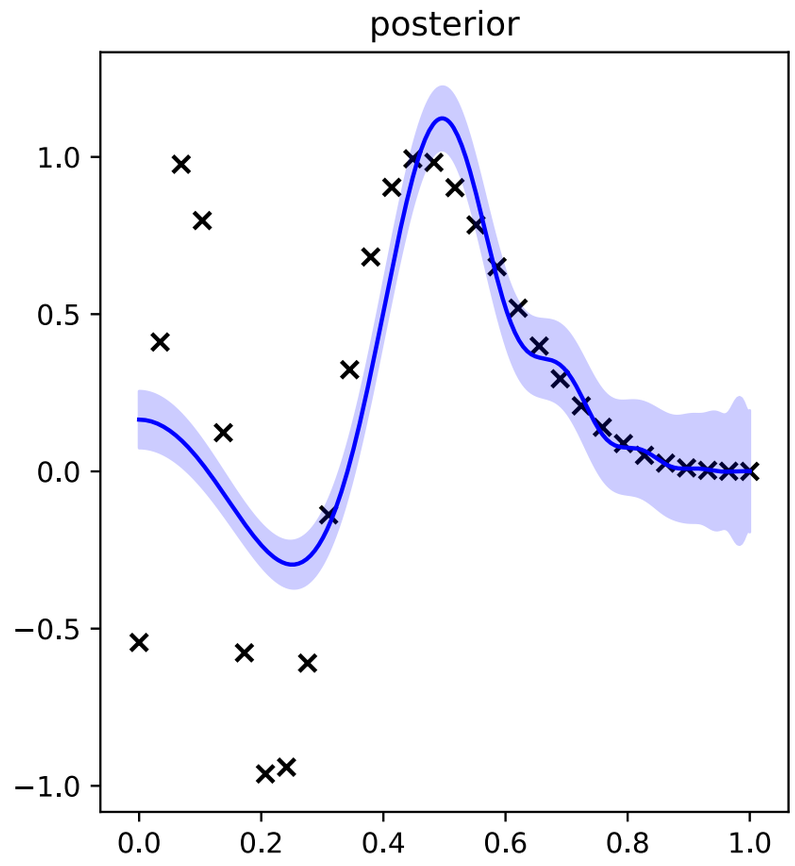




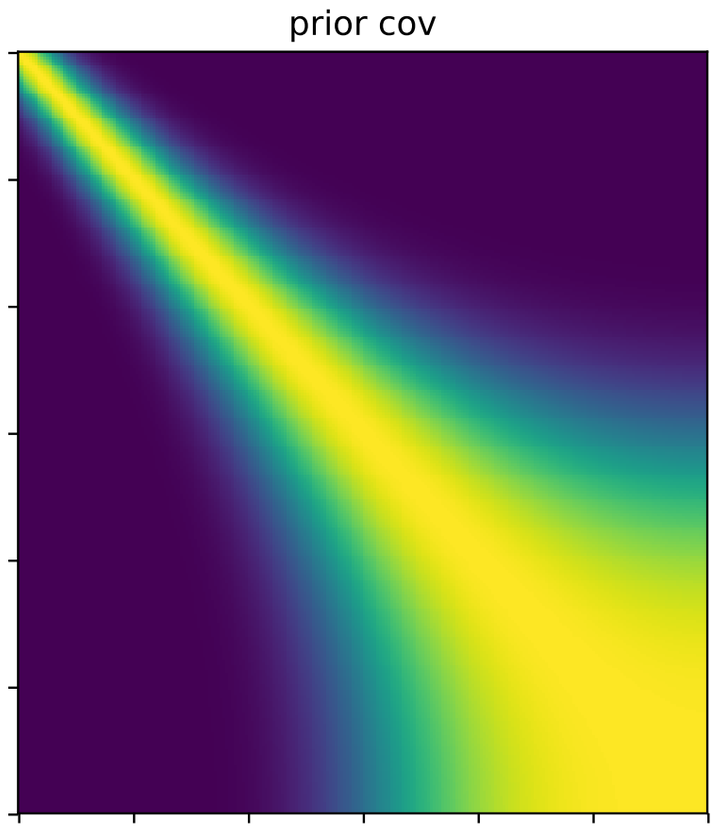
Example: incorrectly specified covariance

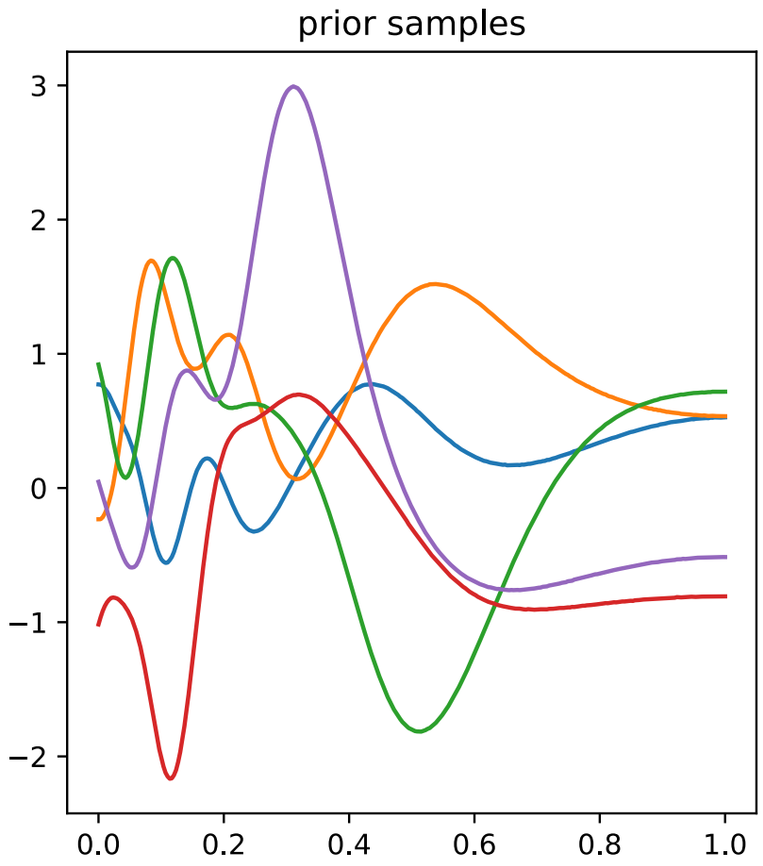


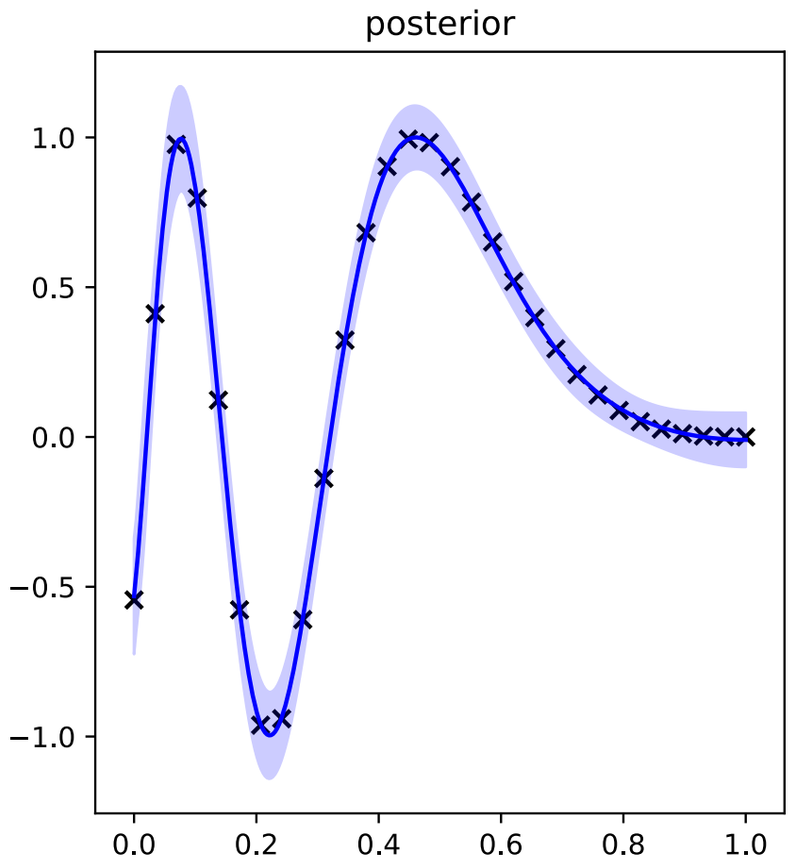




Example: correctly specified covariance



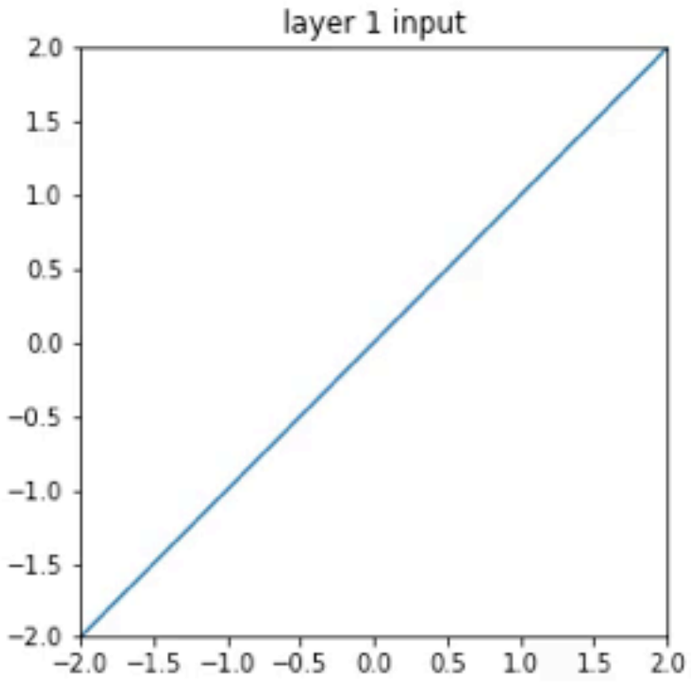


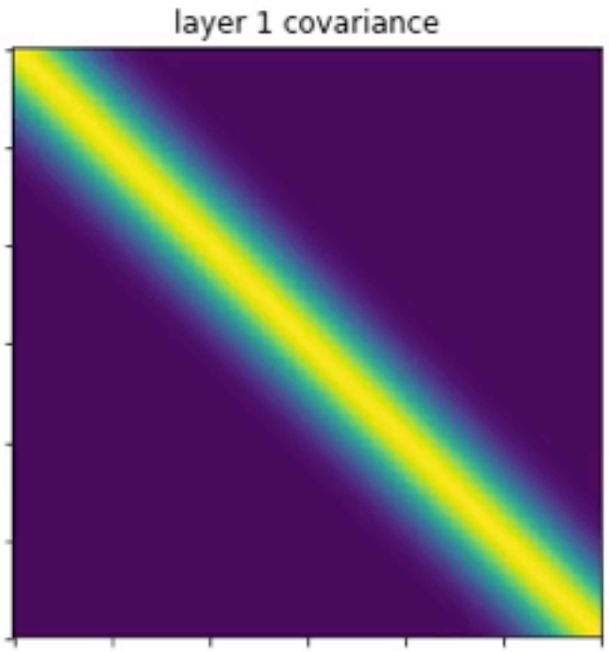


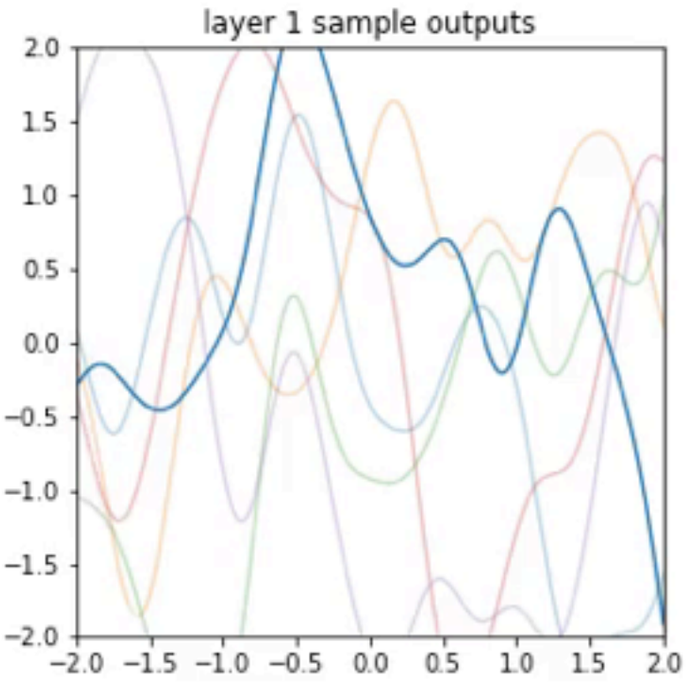
A prior on the warping function

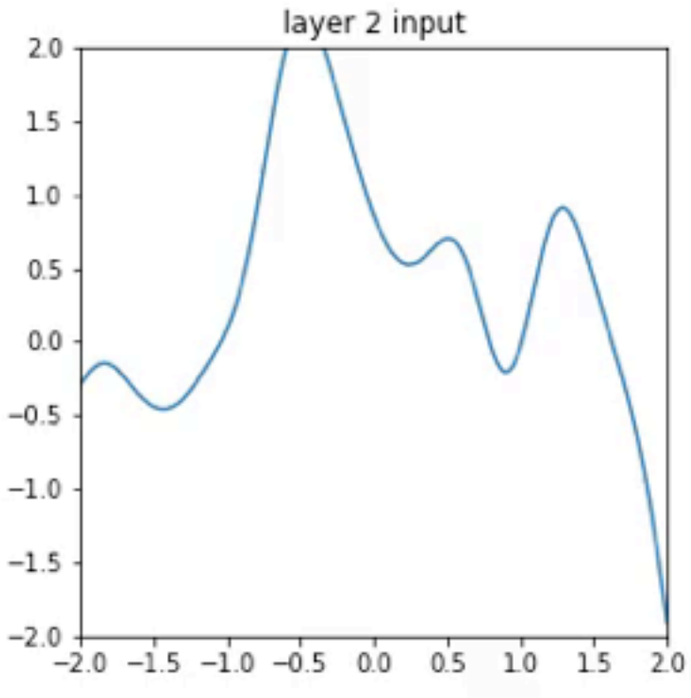
- A Gaussian process is a natural way to define a non-linear prior
- We can use a GP for the warping function
- The model is no longer a Gaussian process

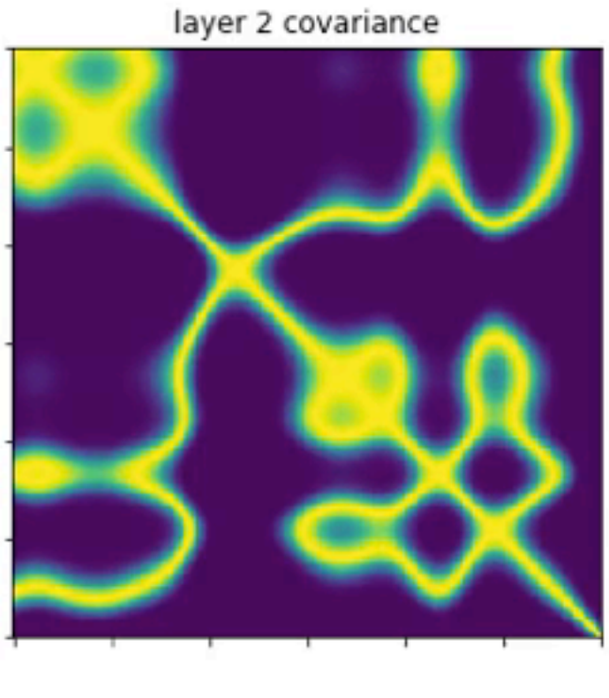
A 2 layer example:

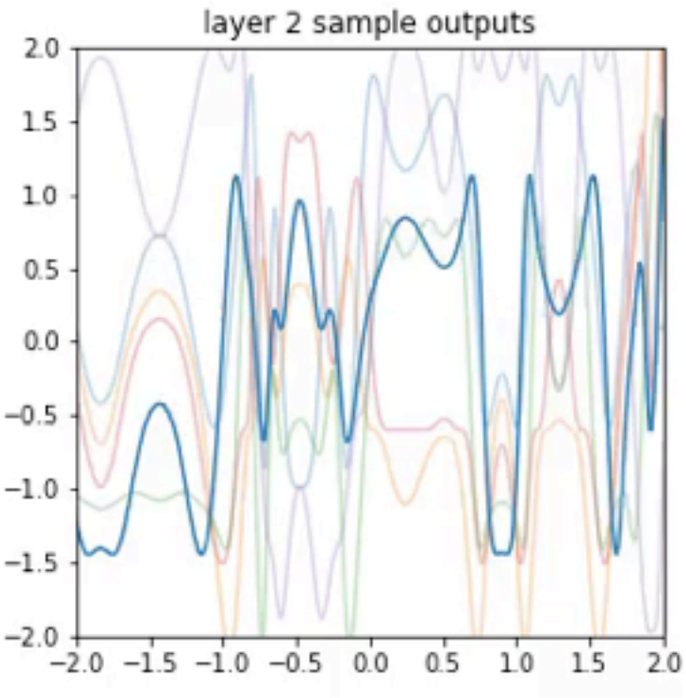


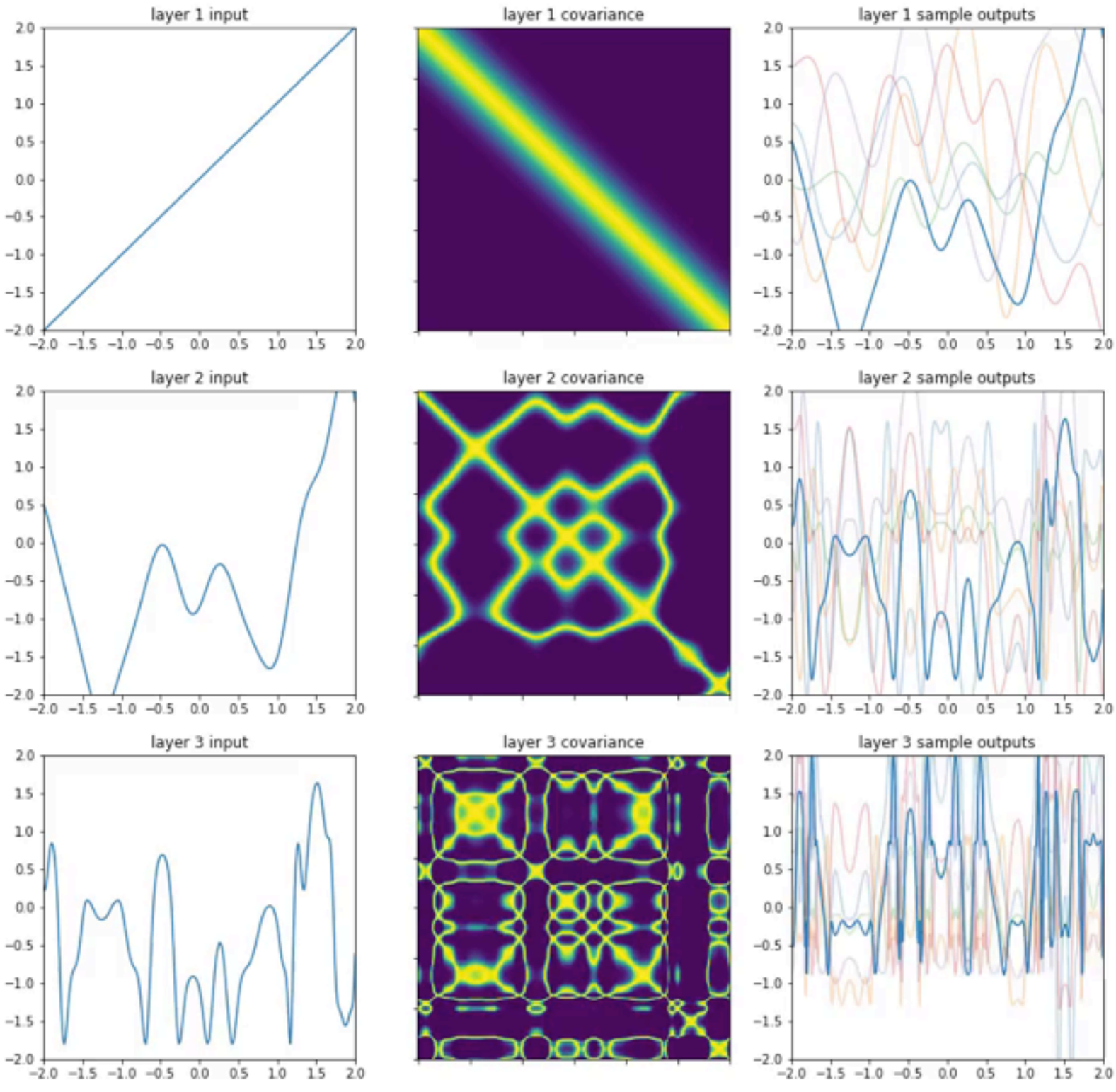


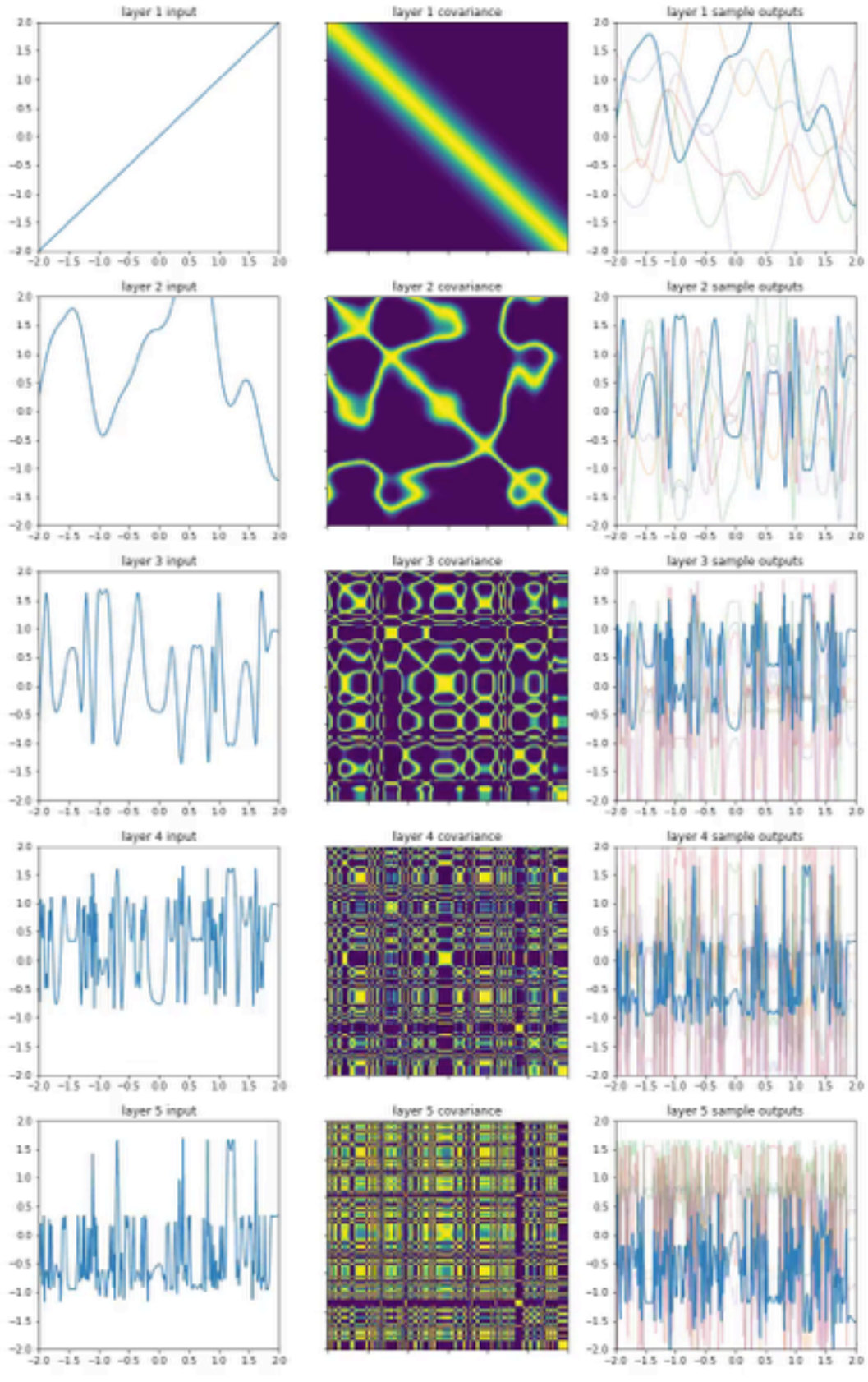








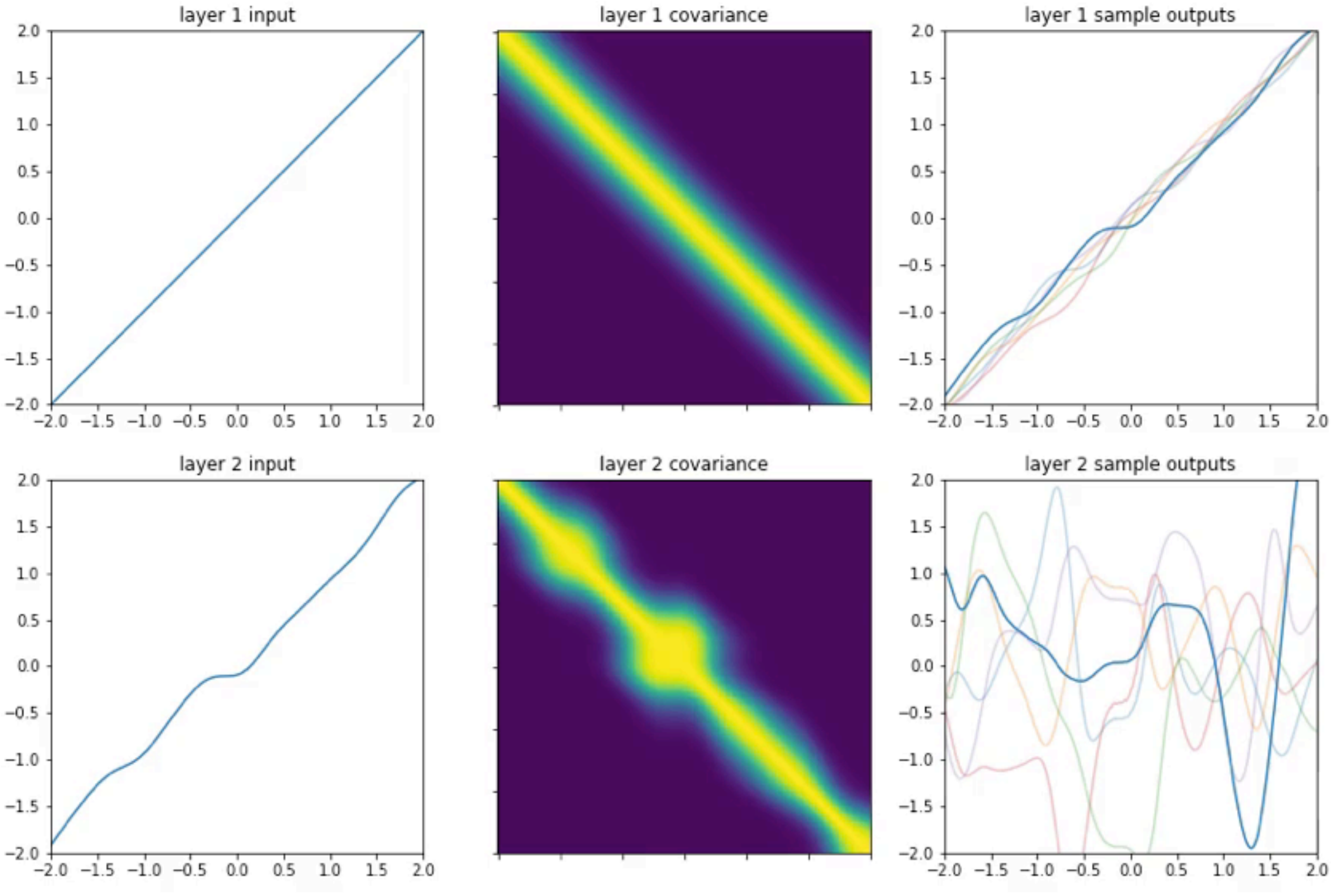

layer 1 input
layer 1 covariance
layer 1 sample outputs
layer 2 input
layer 2 covariance
layer 2 sample outputs
layer 3 input
layer 3 covariance
layer 3 sample outputs

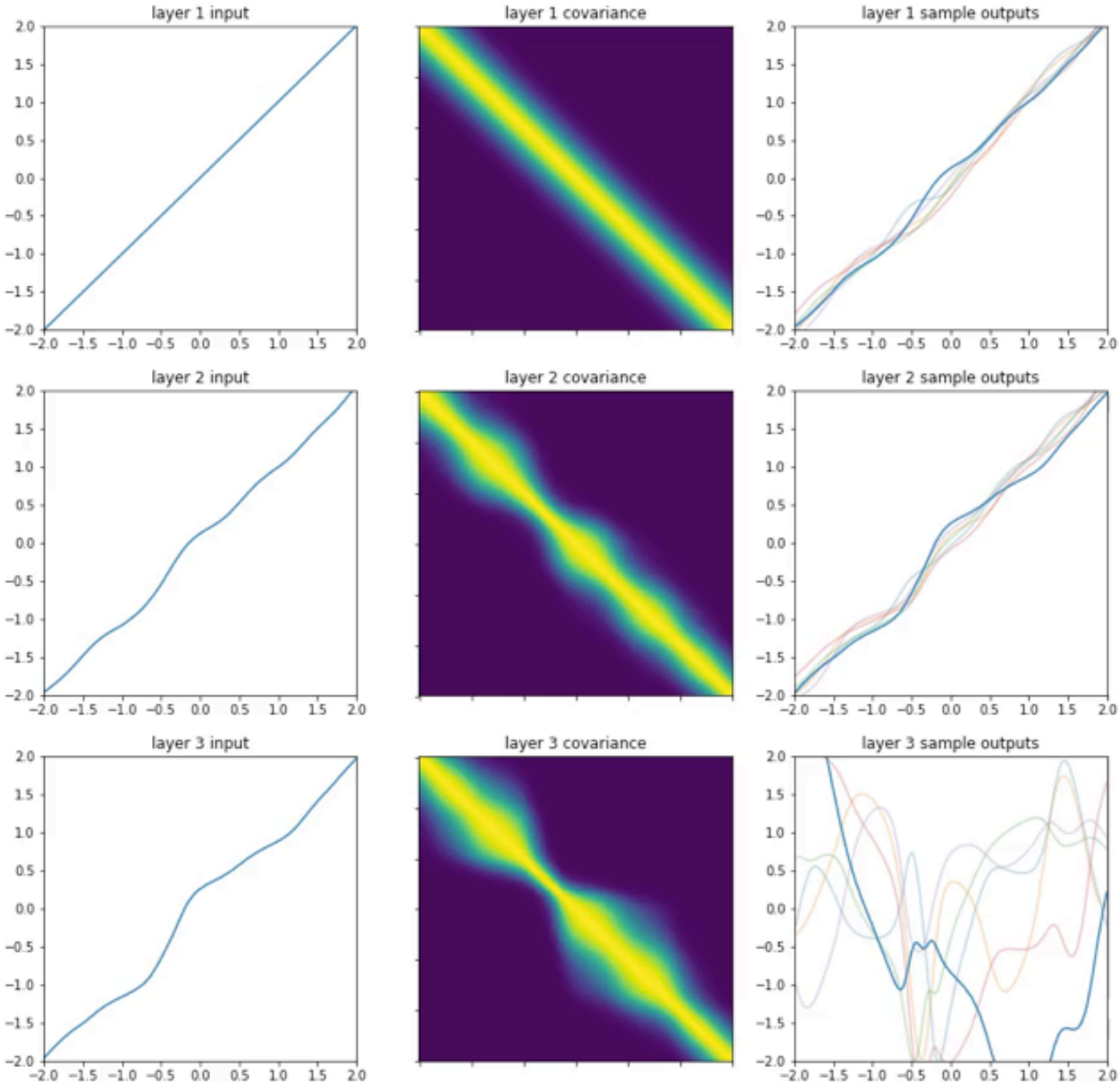

layer 1 input
layer 1 covariance
layer 1 sample outputs
layer 2 input
layer 2 covariance
layer 2 sample outputs
layer 3 input
layer 3 covariance
layer 3 sample outputs
layer 4 input
layer 4 covariance
layer 4 sample outputs
layer 5 input
layer 5 covariance
layer 5 sample outputs

Something is not quite right…

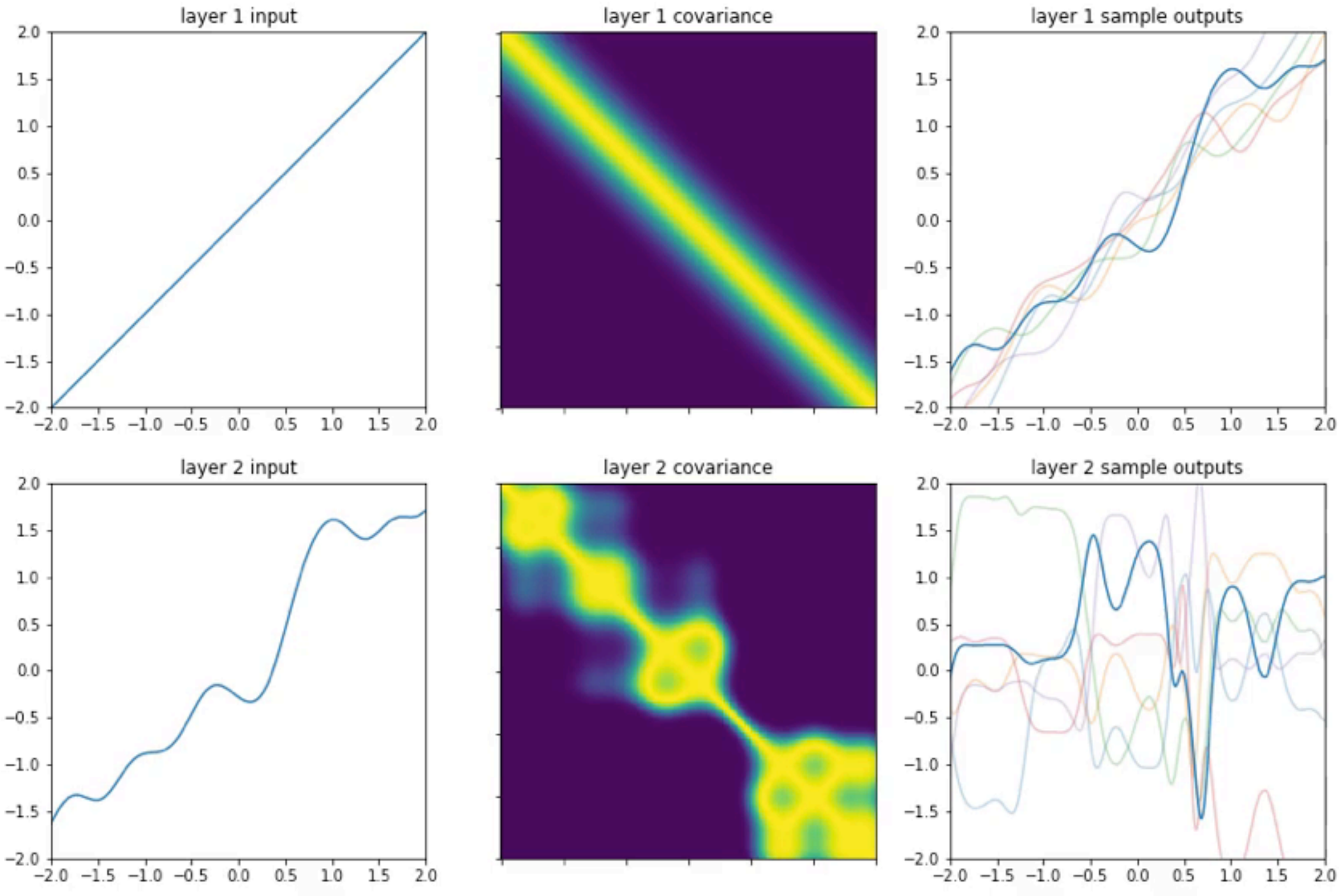
- This model has a pathology
- The posterior becomes increasingly degenerate with more layers
- We lose the prior assumptions of the single layer model completely
- There is no hyperparameter setting that corresponds to a single layer model
- (Also inference is impossible…)

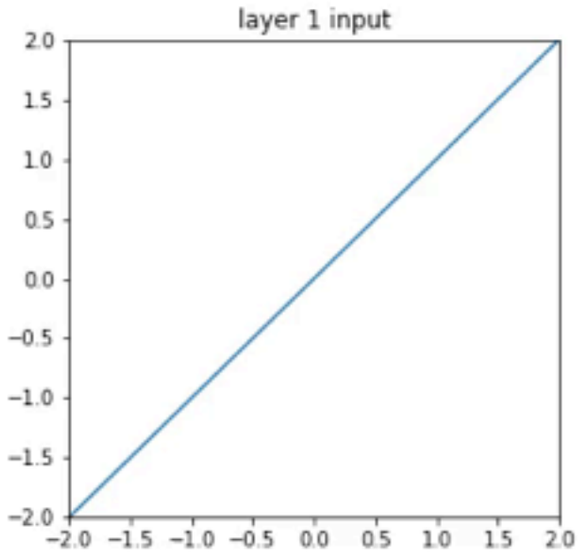
A fix: use a mean function

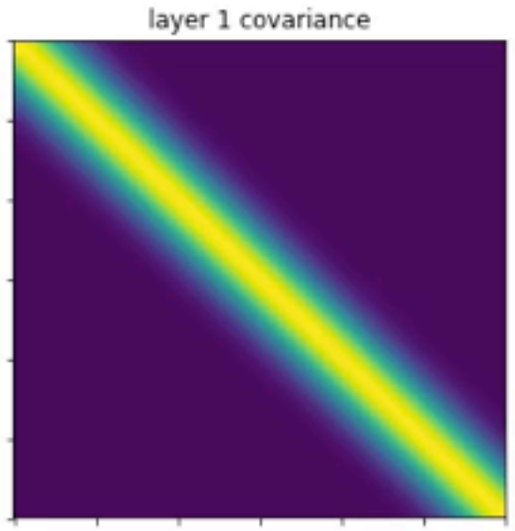


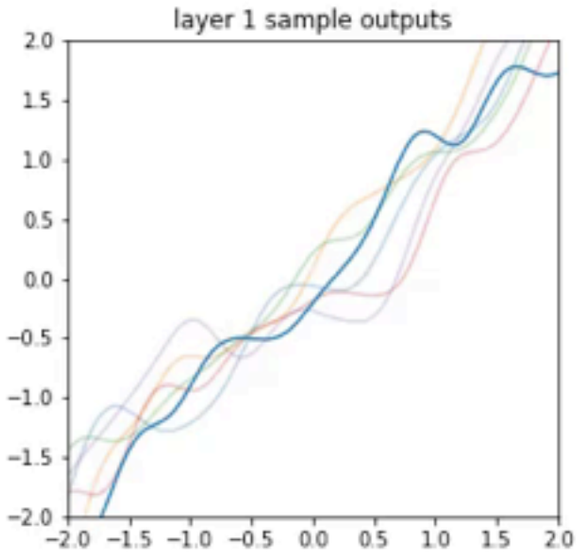

layer 1 input
layer 1 covariance
layer 1 sample outputs
layer 2 input
layer 2 covariance
layer 2 sample outputs
layer 3 input
layer 3 covariance
layer 3 sample outputs

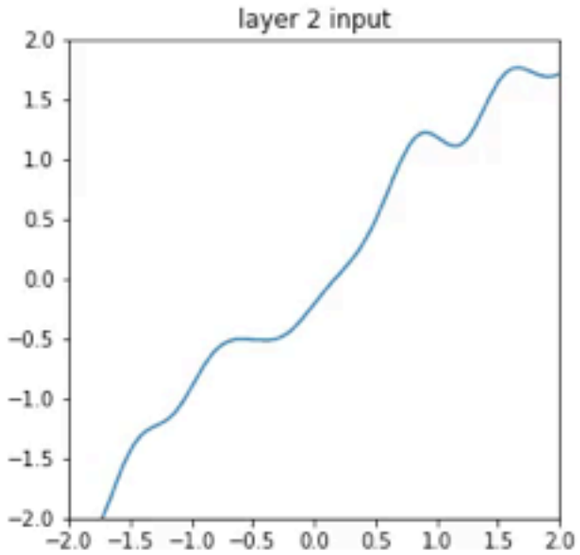
Tuneable complexity

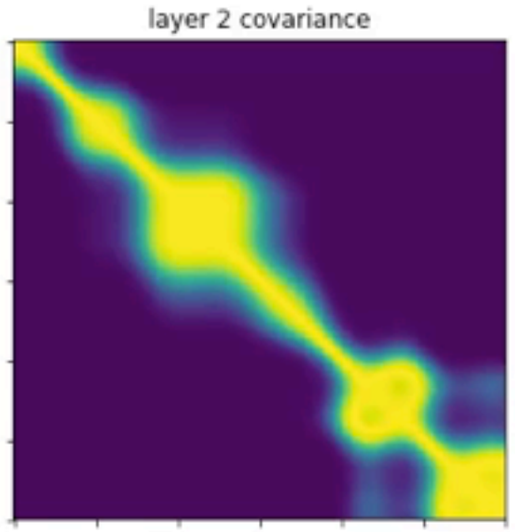


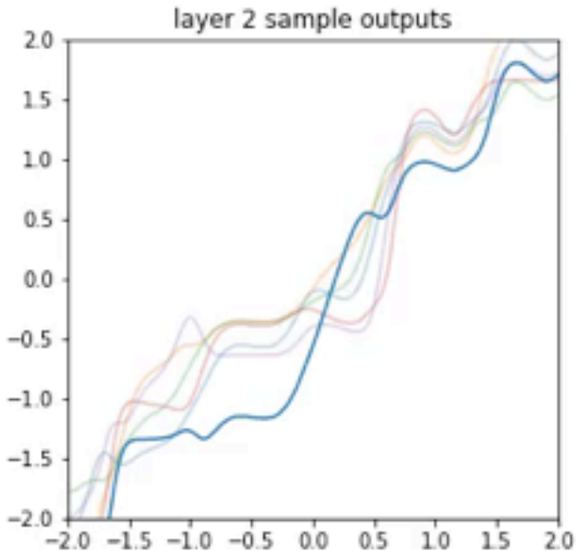

layer 1 input
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0

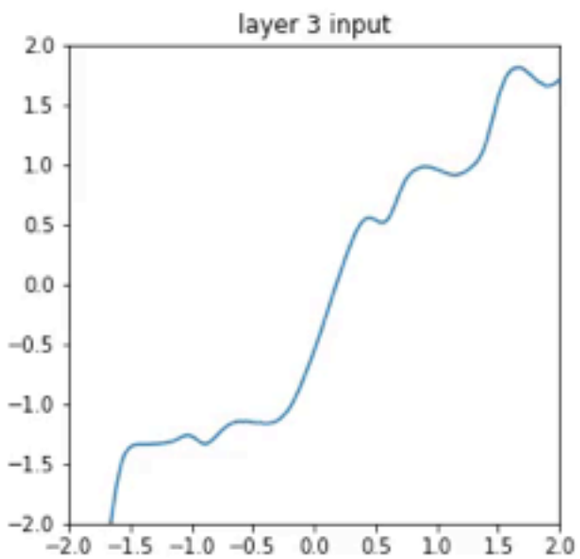

layer 1 covariance

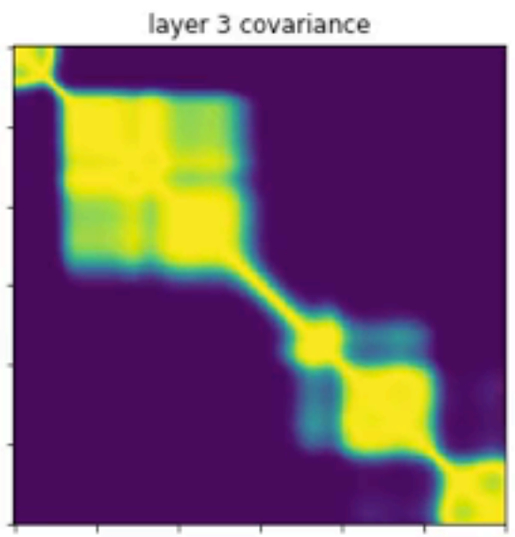

layer 1 sample outputs
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0

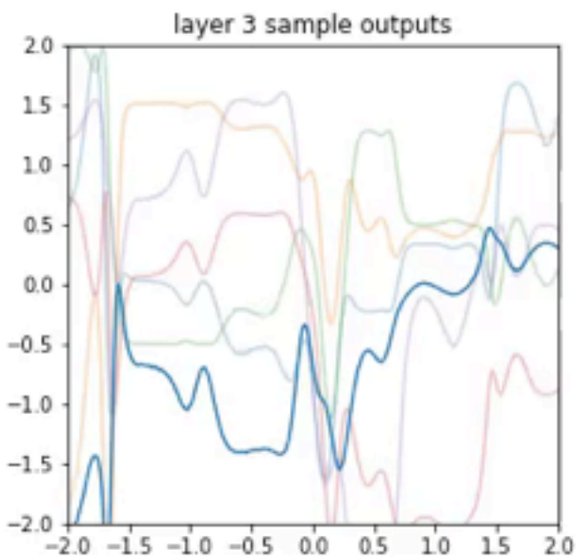

layer 2 input
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0


layer 2 covariance


layer 2 sample outputs
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0


layer 3 input
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0


layer 3 covariance

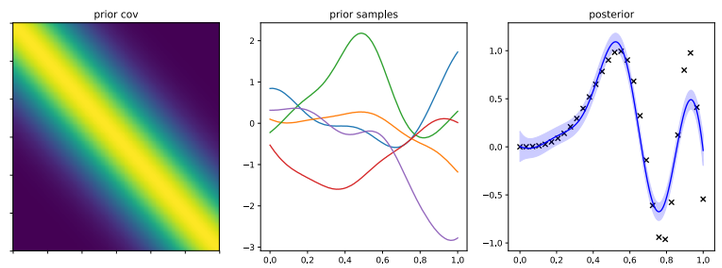

layer 3 sample outputs
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0

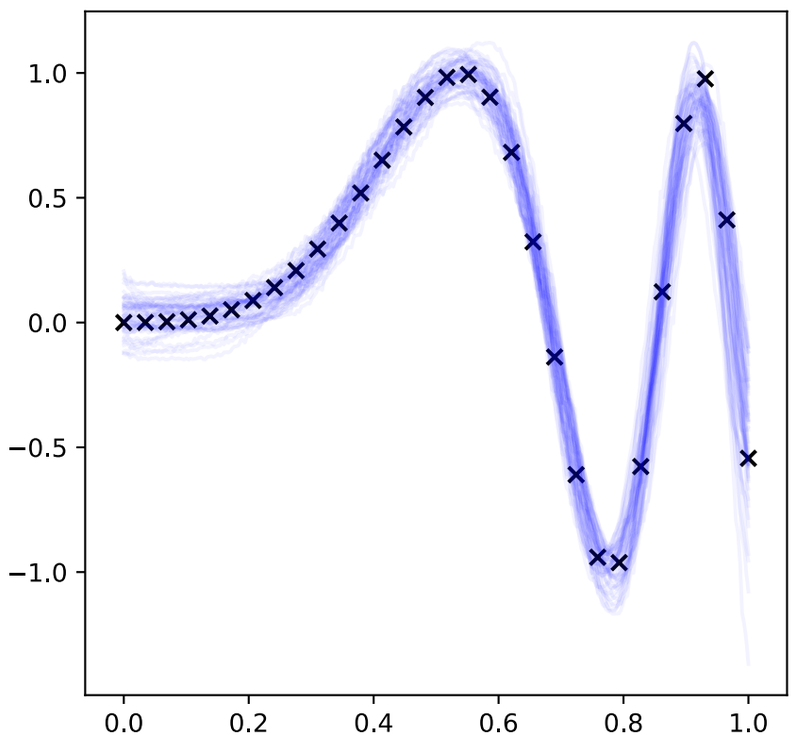
What is a Deep GP?

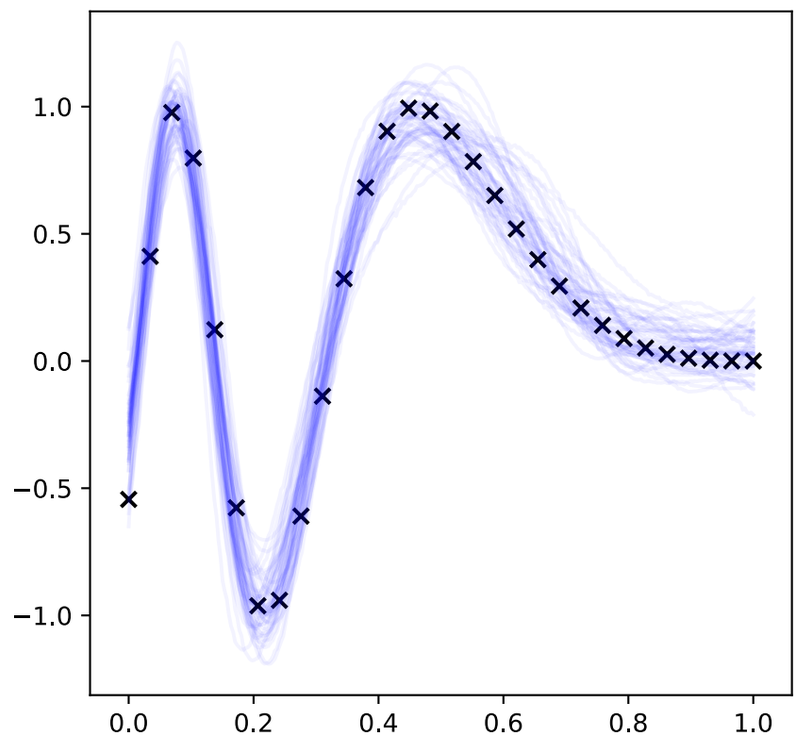
- A Gaussian process with covariance that depends on another Gaussian process
- Our formulation is just one way to define a DGP

What does the posterior look like for the toy problem?

Using the same kernel for both layers (this one), with a linear mean function



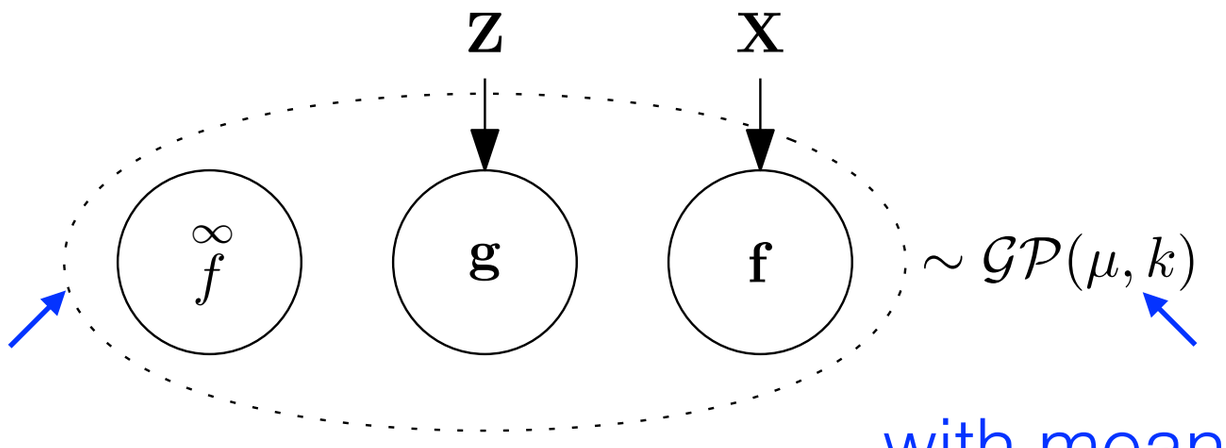




Part 2: Inference

Outline:

1. Variationally sparse GP [Hensman 2013]
2. DGP with Doubly Stochastic VI [Salimbeni 2017]

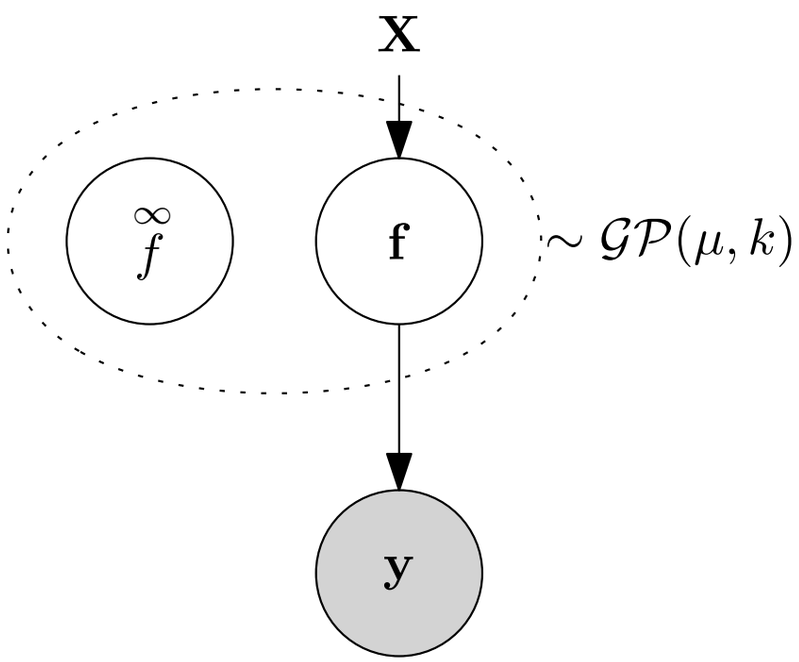


All variables are jointly Gaussian…

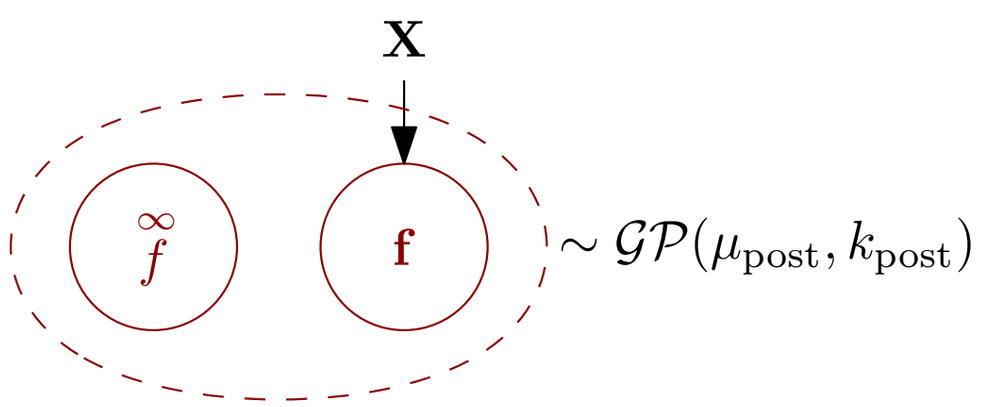
…with mean and covariance given by these functions

GP with Gaussian likelihood

Model



Posterior



N^3 to evaluate

$$\mu_{\text{post}}(\mathbf{x}) = \mu(\mathbf{x}) + k(\mathbf{x}, \mathbf{X})(k(\mathbf{X}, \mathbf{X}) + \sigma^2\mathbf{I})^{-1}(\mathbf{y} - \mu(\mathbf{x}))$$

$$k_{\text{post}}(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}') - k(\mathbf{x}, \mathbf{X})(k(\mathbf{X}, \mathbf{X}) + \sigma^2\mathbf{I})^{-1}k(\mathbf{X}, \mathbf{x}')$$

This isn't good enough in general

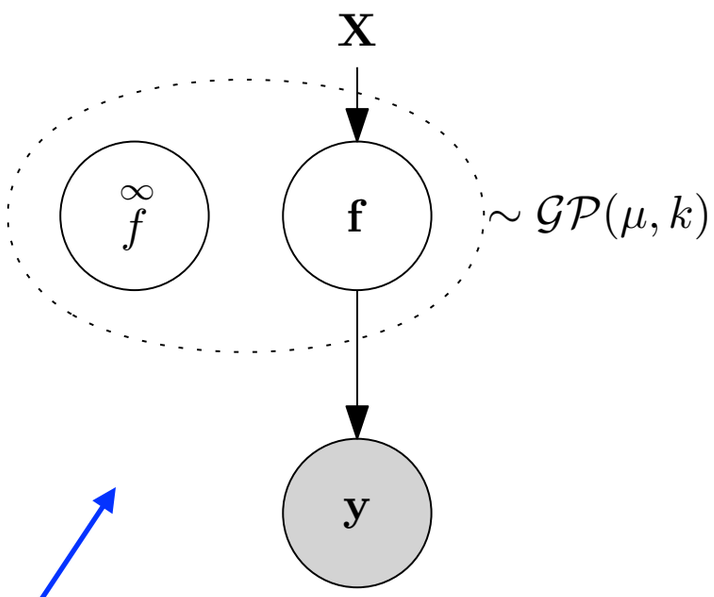
- Only works if the likelihood is Gaussian
- Only practical if N small (<10K)

We proceed with variational inference

$$\text{ELBO} = \mathbb{E}_{\text{simpler model}} \log \frac{\text{actual model}}{\text{simpler model}}$$

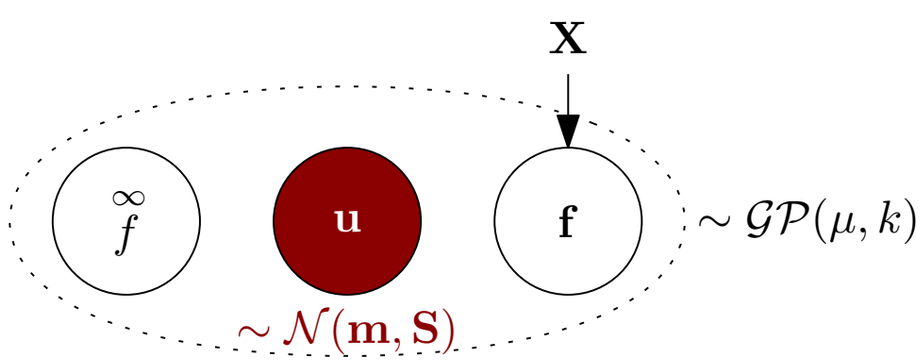
Variational inference with inducing variables

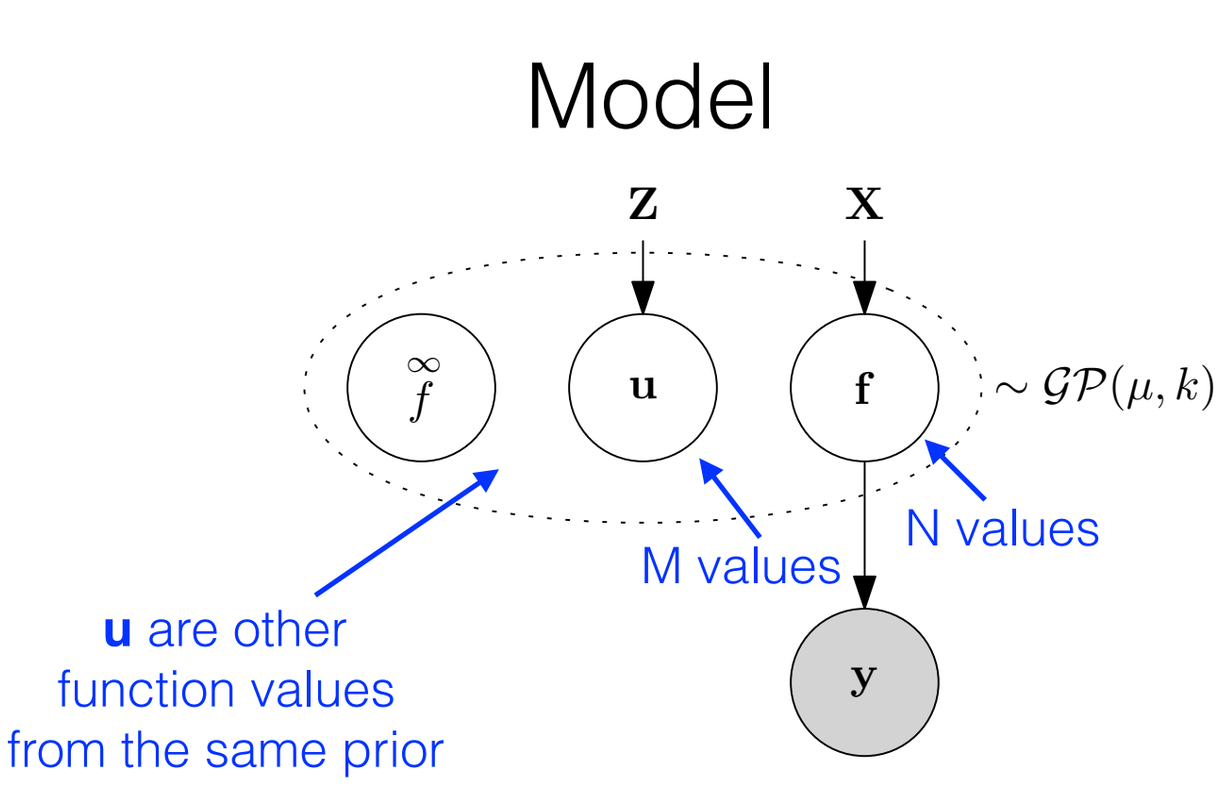
Model

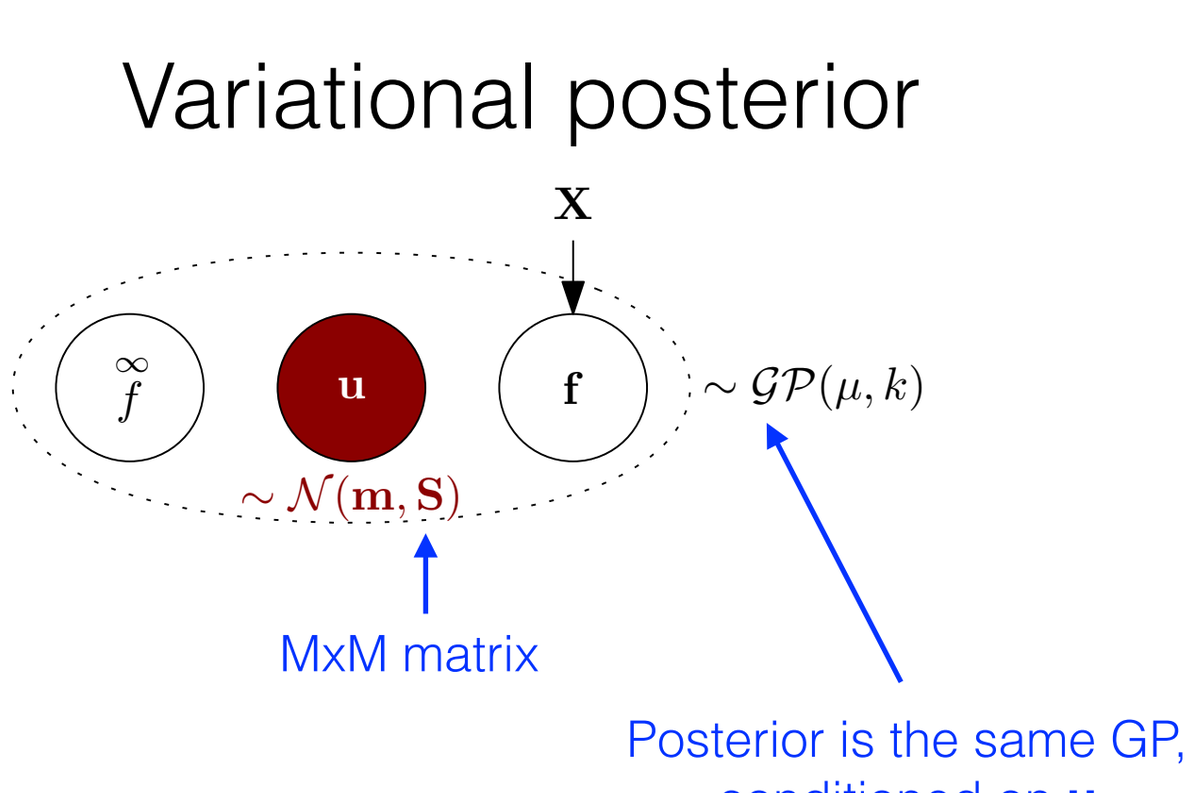


Where is **u**??

Variational posterior

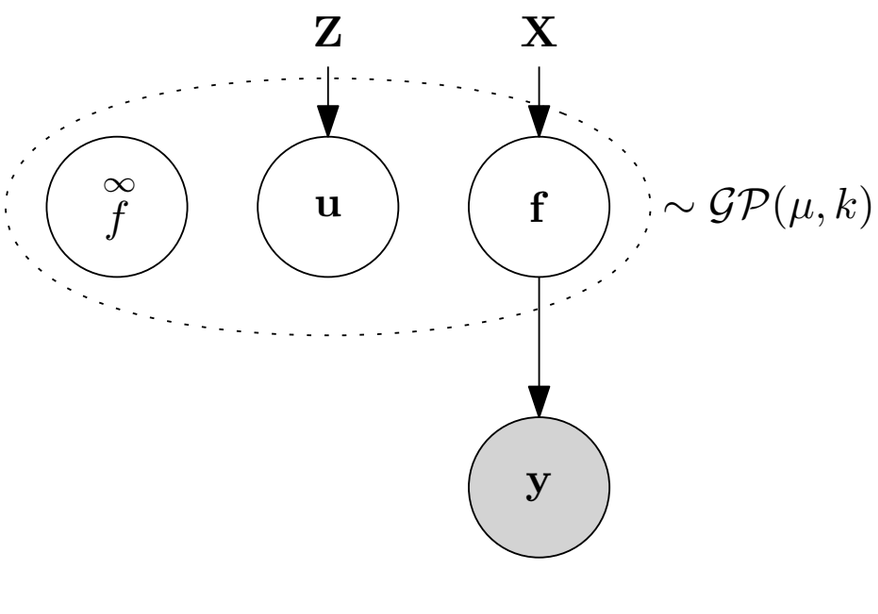




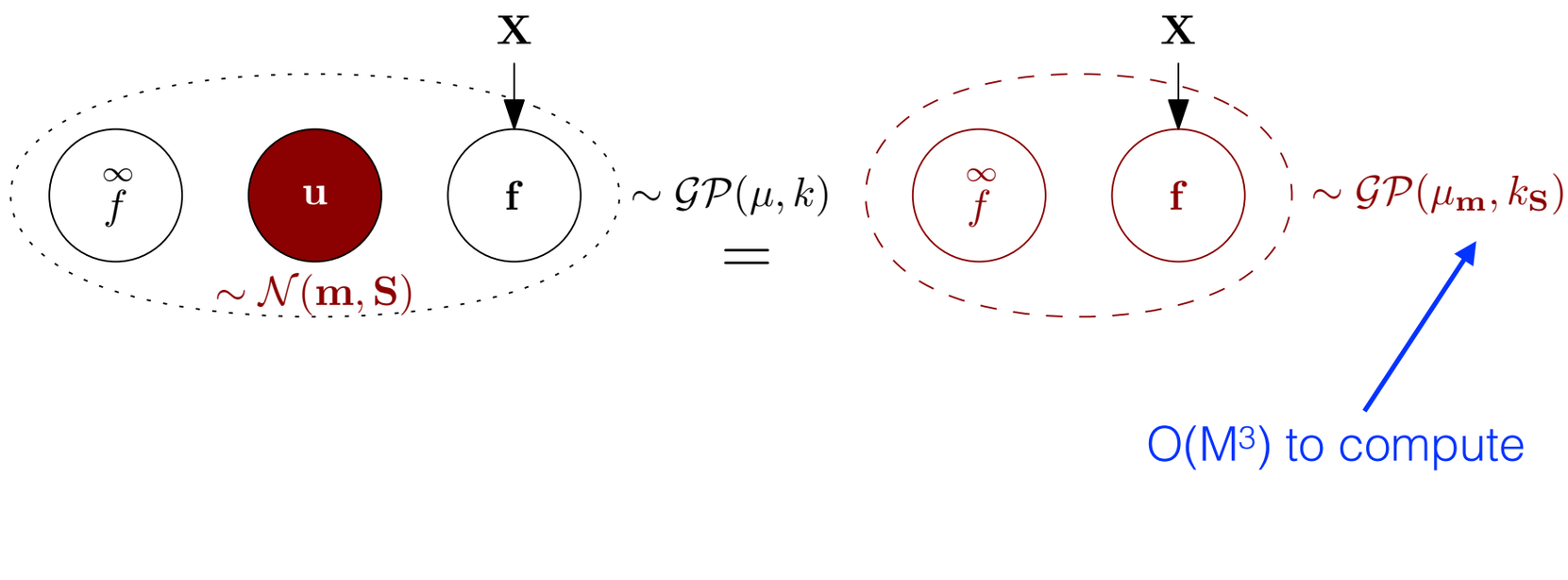


The Gaussian q(**u**) is conjugate to the GP prior, so marginalization is possible in closed form

Model



Variational posterior



The variational distribution can be seen two ways:

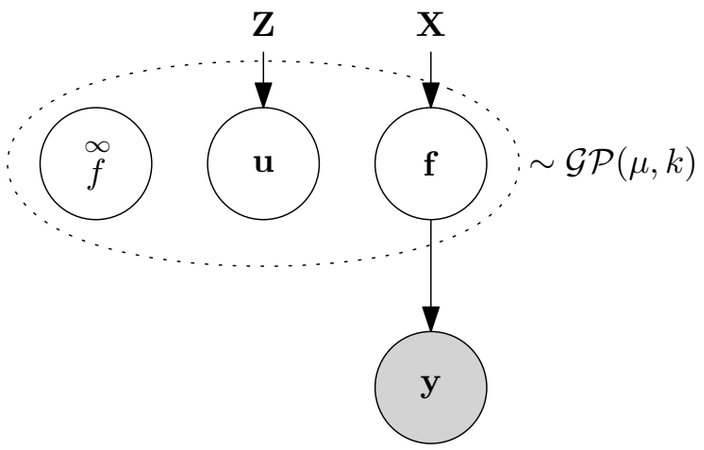
1) It matches the prior everywhere except **u**
2) $q(f) \sim \mathcal{GP}(\mu_{\mathbf{m}}, k_{\mathbf{S}})$

Very nice properties, e.g. marginals/conditionals are Gaussian

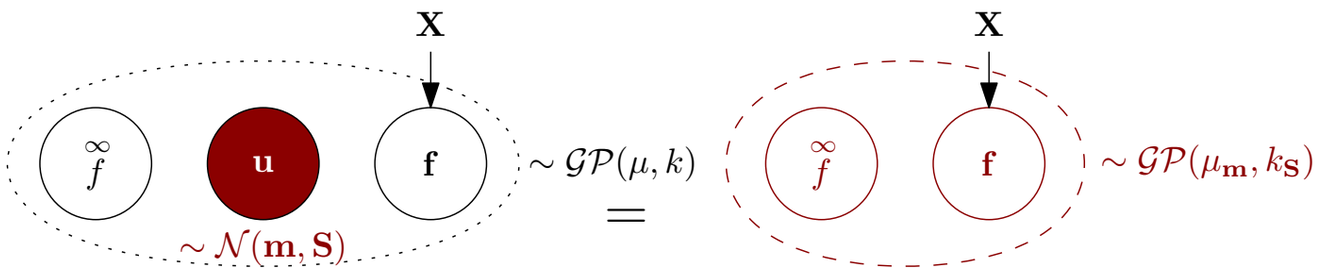
Ratio of joint to variational posterior cancels in two terms

Computing the bound

Model



Variational posterior



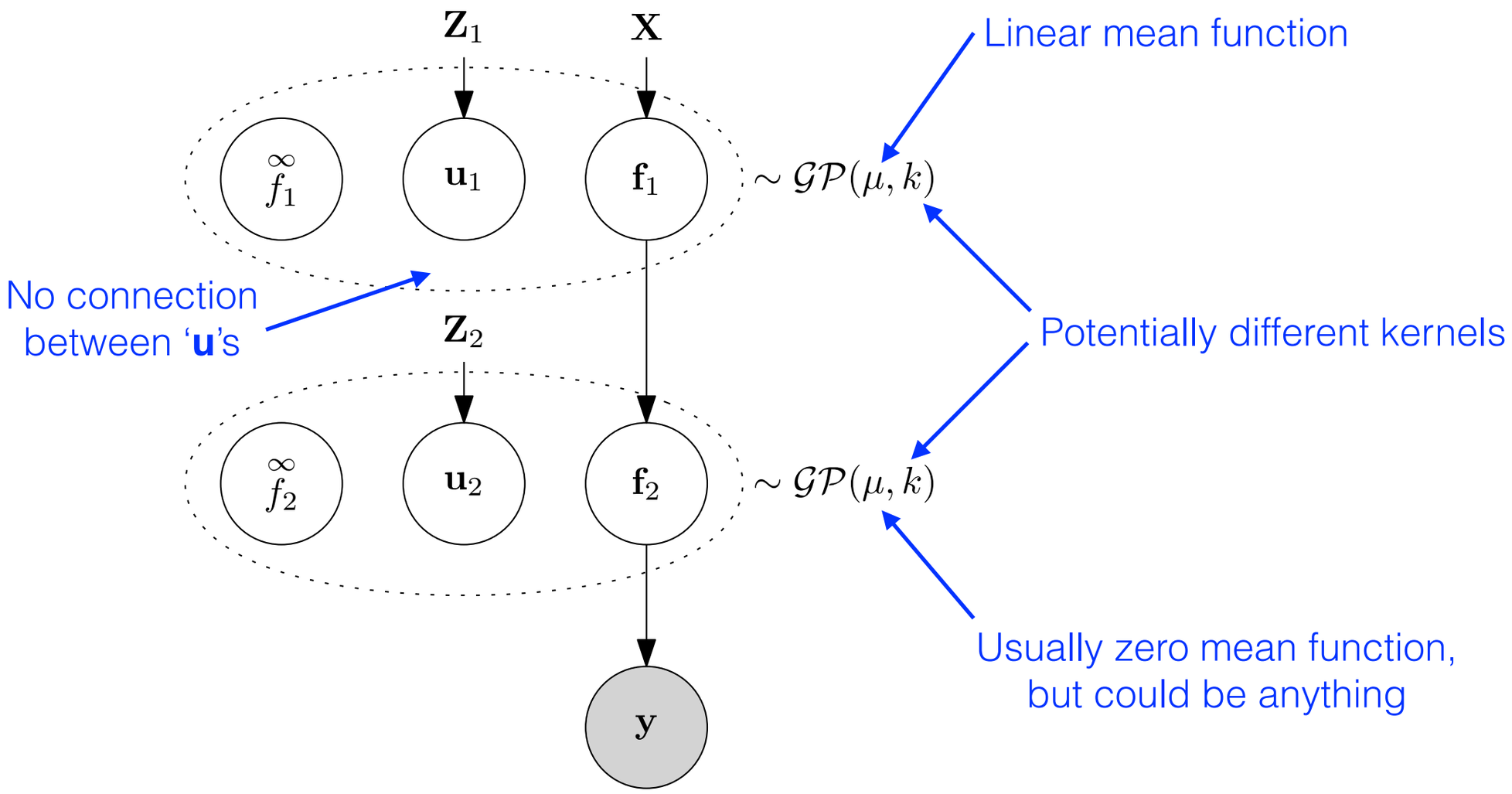
$$\text{ELBO} = \mathbb{E}_{\text{Variational posterior}} \log \frac{\text{Model}}{\text{Variational posterior}}$$

$$= \sum_{i=1}^{N} \mathbb{E}_{q(f_i)} \log(y_i|f_i) - \text{KL}[q(\mathbf{u})||p(\mathbf{u})]$$

$$q(f) \sim \mathcal{GP}(\mu_{\mathbf{m}}, k_{\mathbf{S}})$$

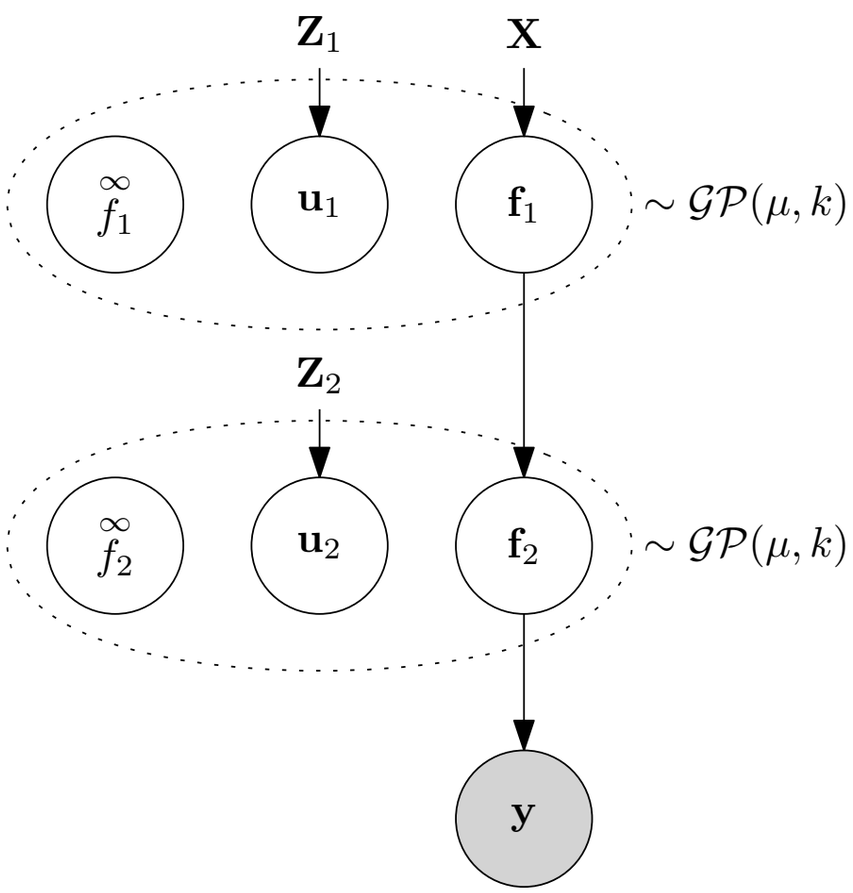
$$q(f_i) \sim \mathcal{N}(\mu_{\mathbf{m}}(\mathbf{x}_i), k_{\mathbf{S}}(\mathbf{x}_i, \mathbf{x}_i)))$$

Deep GP

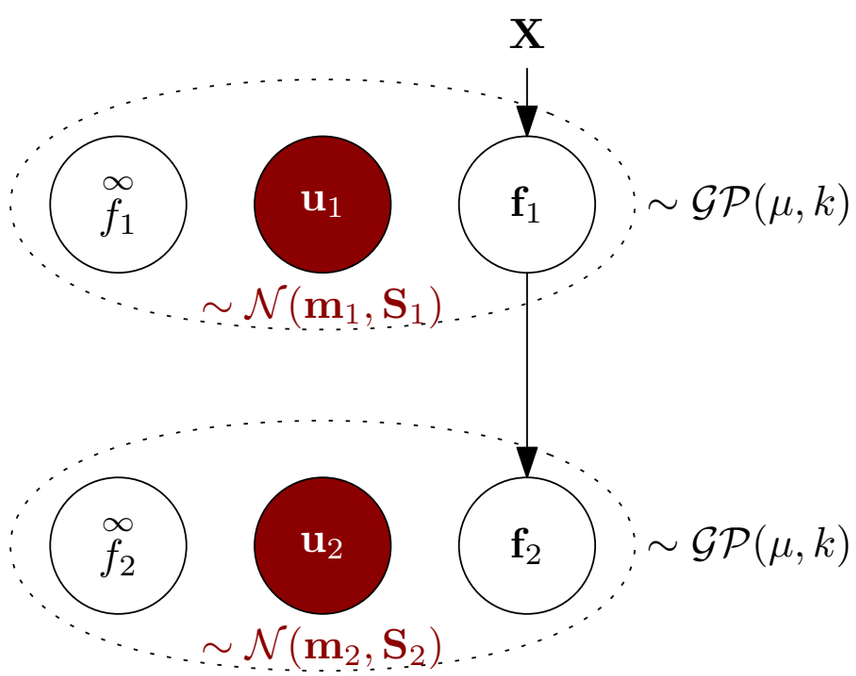

$\mathbf{Z}_1$
$\mathbf{X}$
$\overset{\infty}{f}_1$
$\mathbf{u}_1$
$\mathbf{f}_1$
$\sim \mathcal{GP}(\mu, k)$
Linear mean function
No connection between '**u**'s
$\mathbf{Z}_2$
Potentially different kernels
$\overset{\infty}{f}_2$
$\mathbf{u}_2$
$\mathbf{f}_2$
$\sim \mathcal{GP}(\mu, k)$
Usually zero mean function, but could be anything
$\mathbf{y}$

Deep GP variational posterior

Model



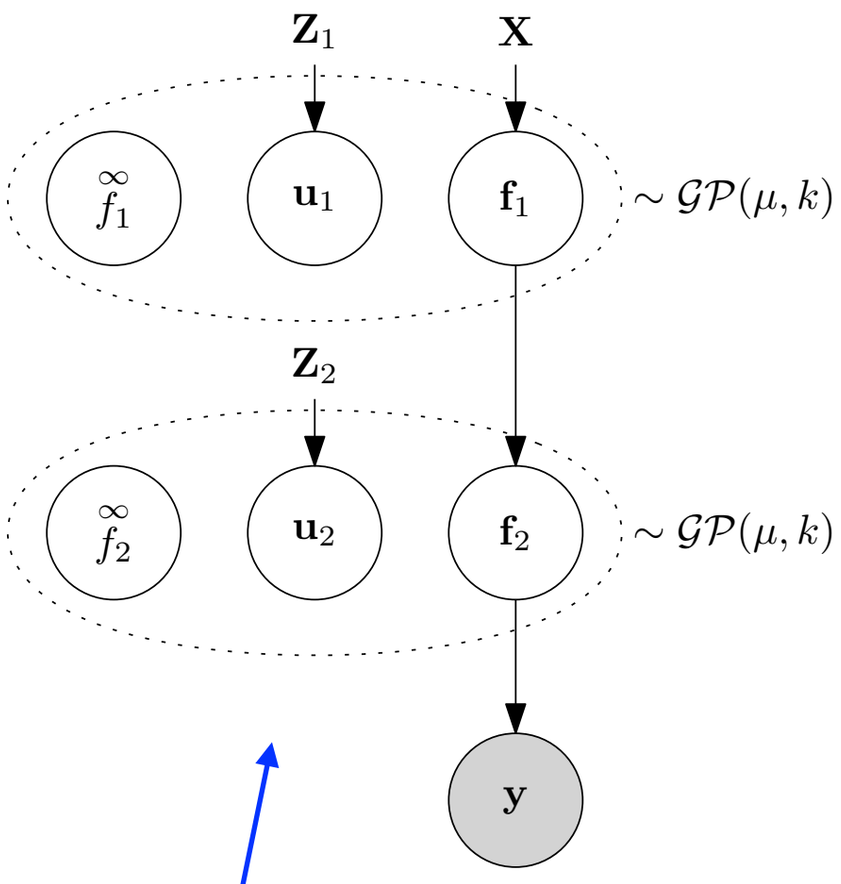
Variational posterior



We can marginalize **u** within each layer

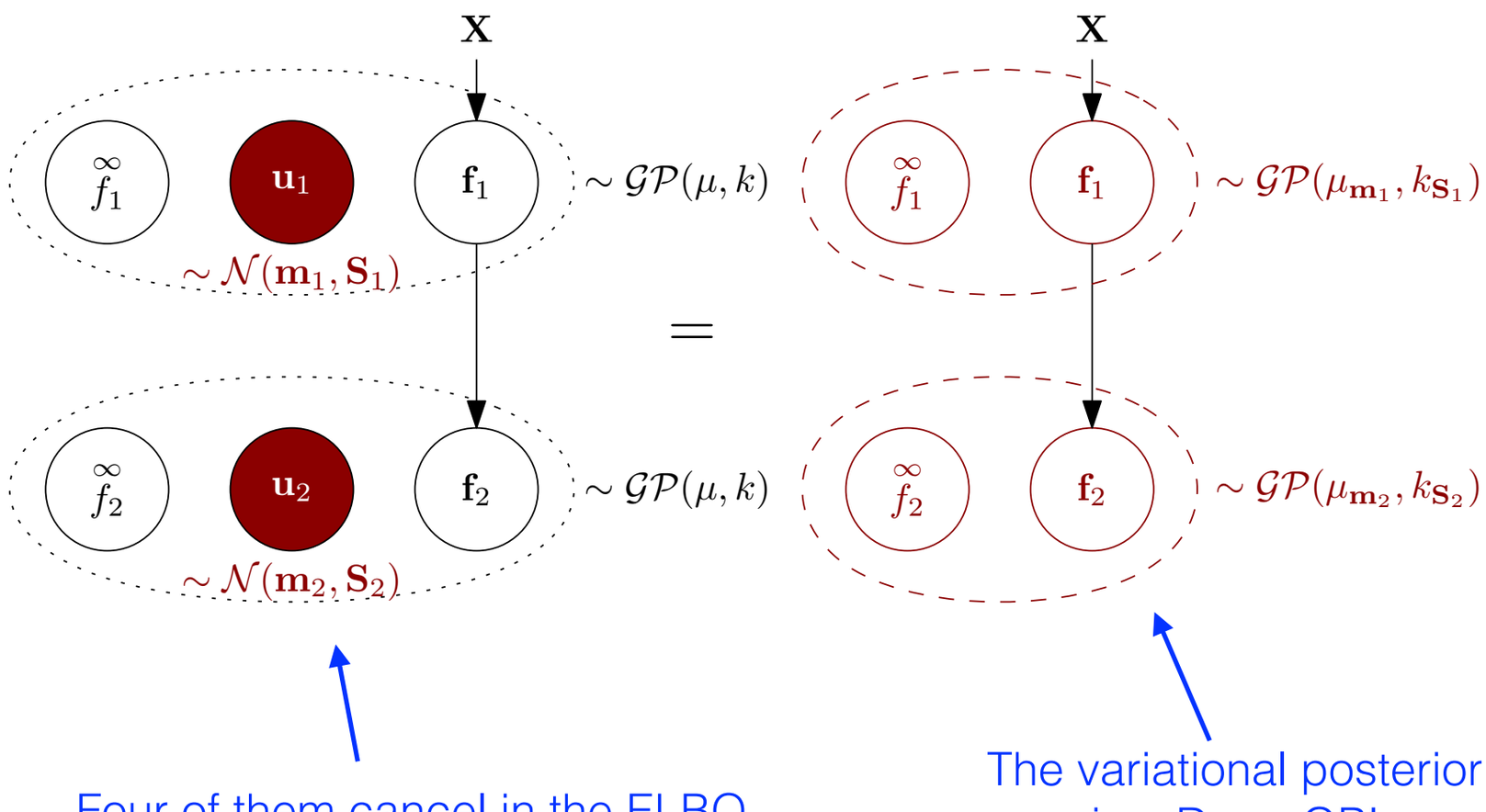
Deep GP variational posterior

Model



Seven terms

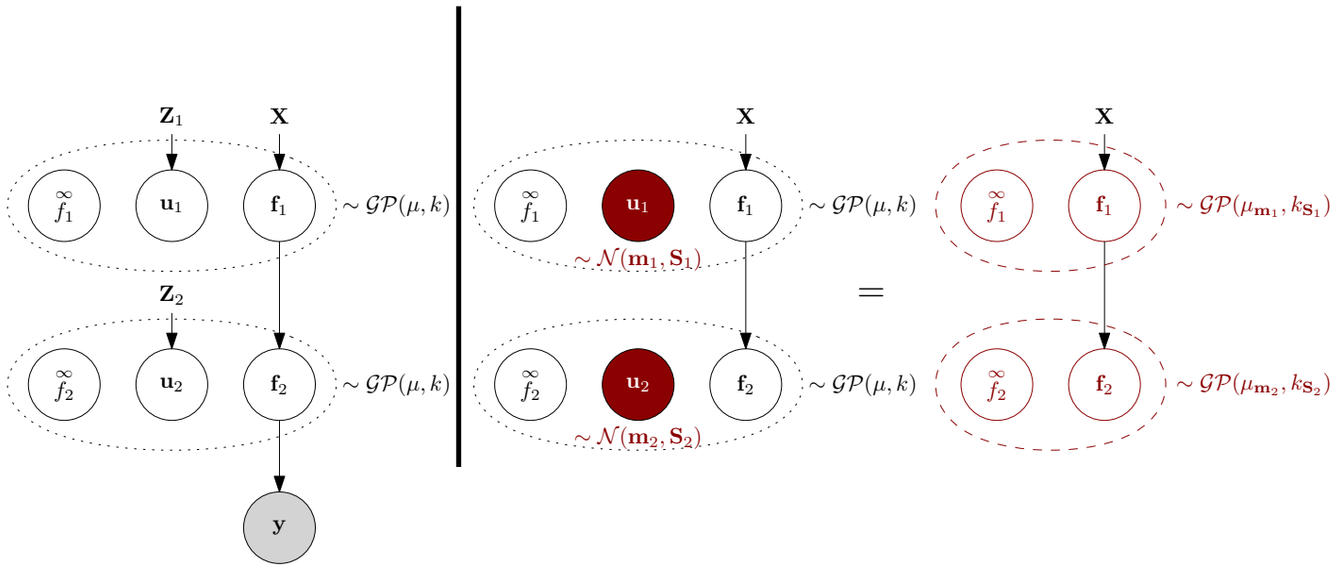
Variational posterior



Four of them cancel in the ELBO leaving three: two KL terms and a likelihood expectation

The variational posterior is a Deep GP!

The Deep GP ELBO



$$\text{ELBO} = \mathbb{E}_{q(f_1,f_2)} \log p(\mathbf{y}|\mathbf{f}_2) - \text{KL}[q(\mathbf{u}_1)||p(\mathbf{u}_1)] - \text{KL}[q(\mathbf{u}_2)||p(\mathbf{u}_2)]$$

likelihood factorizes

$$\mathbb{E}_{q(f_1,f_2)} \sum_{i=1}^{N} \log p(y_i|[\mathbf{f}_2]_i)$$

We need to estimate this term

Analytic

Estimation of the log-likelihood expectation

$$\mathbb{E}_{q(f_1,f_2)} \sum_{i=1}^{N} \log p(y_i|[\mathbf{f}_2]_i)$$

- We only need the **marginals** of the variational distribution.
- Samples from the variational distribution can be taken using only univariate Gaussians (just a prior draw from a Deep GP)

$$\begin{aligned} q([f_1]_i, [f_2]_i) &= q([f_1]_i|[f_2]_i)q([f_2]_i|\mathbf{x}_i) \\ &= \mathcal{N}([f_2]_i|a,b)\mathcal{N}([f_1]_i|c,d) \\ a &= \mu_{\mathbf{m}_2}([f_1]_i), \quad b = k_{\mathbf{S}_2}([f_1]_i,[f_1]_i) \\ c &= \mu_{\mathbf{m}_1}(x_i), \quad d = k_{\mathbf{S}_1}(\mathbf{x}_i,\mathbf{x}_i) \end{aligned}$$

Estimation of the log-likelihood expectation

$$\mathbb{E}_{q(f_1,f_2)} \sum_{i=1}^{N} \log p(y_i|[\mathbf{f}_2]_i)$$

We make two stochastic approximations :

1. Approximate the integration with the simple Monte-Carlo estimate

2. Approximate the full sum with a minibatch

We use reparameterization trick for gradients, with Adam optimizer.

Results highlights (big data)

Table 2: Regression test RMSE results for large datasets

N	D	SGP	SGP 500	DGP 2	DGP 3	DGP 4	DGP 5
year	463810	90	10.67	9.89	9.58	8.98	8.93	**8.87**
airline	700K	8	25.6	25.1	24.6	24.3	24.2	**24.1**
taxi	1B	9	337.5	330.7	281.4	270.4	268.0	**266.4**

Results highlights (small data)

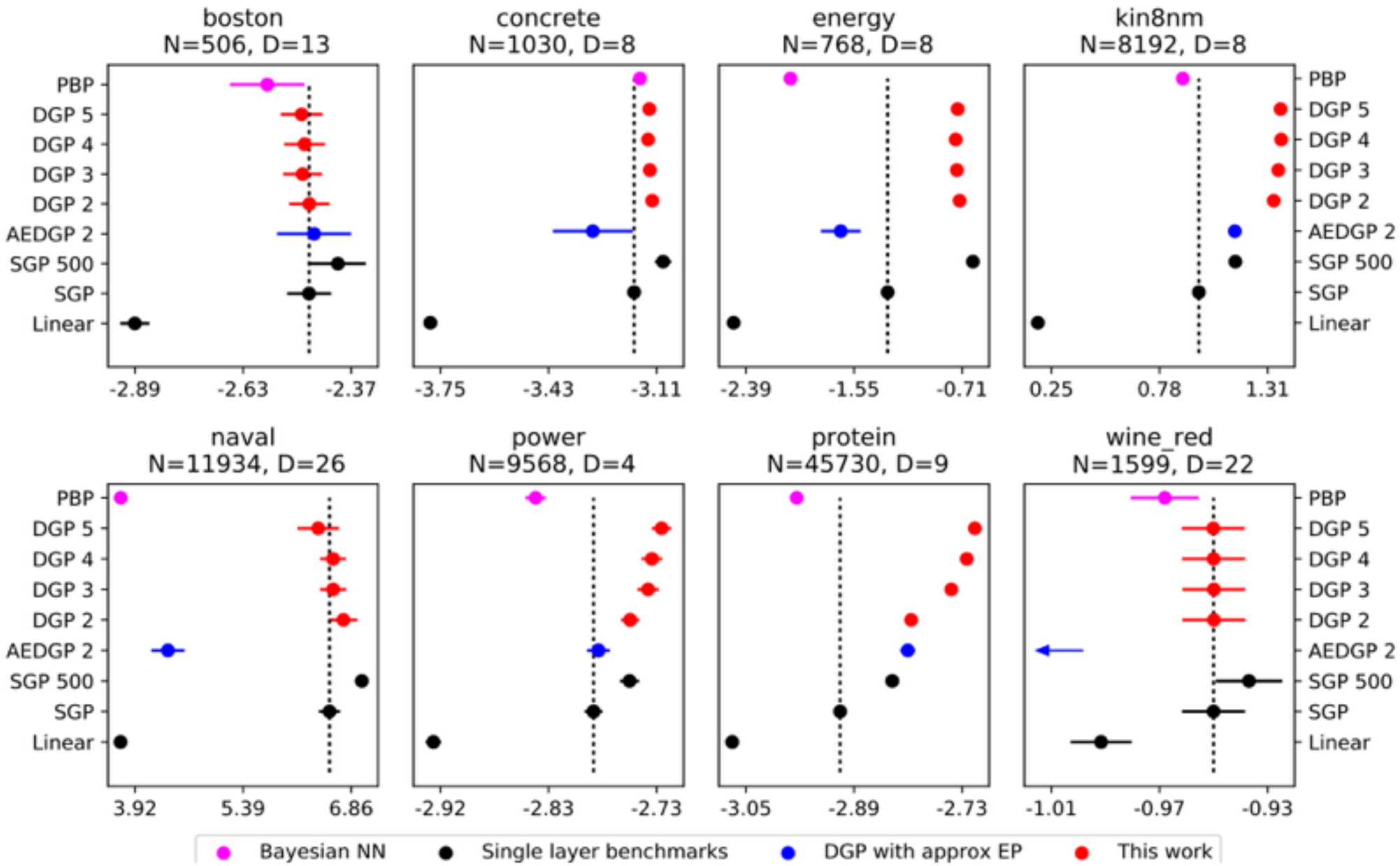


Figure 1: Regression test loglikelihood results on benchmark datasets. Higher (to the right) is better. The sparse GP with the same number of inducing points is highlighted as a baseline.

In practice: when is the Deep GP a useful model?

- Any problem where a GP is a useful model (will reduce to the single layer model as necessary)
- As a black box function approximator with a generic kernel (e.g. Matern/RBF)
- For non-stationary data
- For data that is mostly well modelled by a GP, but has some nasty patches which are not

Deep GP limitations

- Cannot model bimodal data*
- Does not have convolutional structure*
- Scaling is linear in inner layer dimension as well as depth
- GP warping might not be an appropriate prior (even with mean function)

* work in progress

Thanks for listening