UNIVERSITY OF CAMBRIDGE

Neural Processes

DEPARTMENT OF ENGINEERING

Ginte Petrulionyte ⋮ Yuriko Kobe ⋮ Jack Davis

Motivation

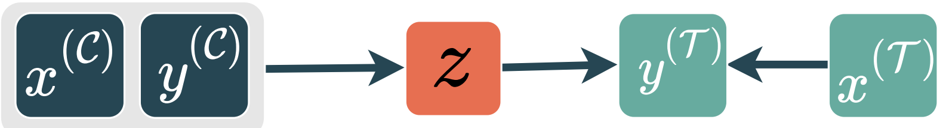
Neural networks (NNs) are effective function approximators, but do not capture uncertainty over their predictions and cannot easily be updated after training.

Gaussian Processes (GPs) are stochastic processes which define a distribution over functions, encapsulating the uncertainty in model predictions. However, GPs are computationally expensive and can be restricted by the functional form of the kernels.

Neural Processes [1] (NPs) combine features of NNs and GPs to address the limitations of both. Like GPs, NPs learn distributions over functions, retaining prediction uncertainty. However, the predictive distribution has the form of an infinite mixture of Gaussians, allowing arbitrary distributions to be modelled and removing the requirement to specify a functional GP kernel.

Graphical Model

Given a context set $\mathcal{C} = \{\mathbf{x}^{(\mathcal{C})},\mathbf{y}^{(\mathcal{C})}\}$ NPs aim to infer target function values $y^{(\mathcal{T})}$ in $\mathcal{T} = \{\mathbf{x}^{(\mathcal{T})},\mathbf{y}^{(\mathcal{T})}\}$ via an intermediate latent variable, allowing for global uncertainty to be modelled, unlike the precursor Conditional NPs [2].



This is accomplished via an Encoder/Decoder Neural Network maintaining the stochastic process exchangeability and consistency characteristics. Inference on a new dataset corresponds to a single forward pass which scales linearly with the number of datapoints, as opposed to cubically for GPs.

Training

One downside of NPs is that the likelihood for a target set is analytically intractable due to the required indefinite integral over latent variables.

$$p_\theta\left(\mathbf{y}^{(\mathcal{T})}|\mathbf{x}^{(\mathcal{T})};\mathcal{C}\right) = \int p_\theta(\mathbf{z} \mid \mathcal{C}) \prod_{t=1}^{n} p_\theta\left(y_t^{(\mathcal{T})} \mid x_t^{(\mathcal{T})};\mathbf{z}\right)\mathrm{d}\mathbf{z}$$

This is mitigated by the use of approximate inference techniques. In particular, in training we optimise a modified Evidence Lower Bound (ELBO), derived using two forward passes, one for $p_\theta(\mathbf{z}|\mathcal{C})$ and one for $p_\theta(\mathbf{z}|\mathcal{D})\,, \mathcal{D} = \mathcal{C} \cup \mathcal{T}$.

$$\log p_\theta\left(\mathbf{y}^{(\mathcal{T})}|\mathbf{x}^{(\mathcal{T})};\mathcal{C}\right) \geq$$
$$\mathbb{E}_{\mathbf{z}\sim p_\theta(\mathbf{z}|\mathcal{D})}\left[\log p_\theta\left(\mathbf{y}^{(\mathcal{T})}|\mathbf{x}^{(\mathcal{T})};\mathbf{z}\right)\right] - \mathrm{KL}(p_\theta(\mathbf{z} \mid \mathcal{D})\|p_\theta(\mathbf{z} \mid \mathcal{C}))$$

1D Function Regression

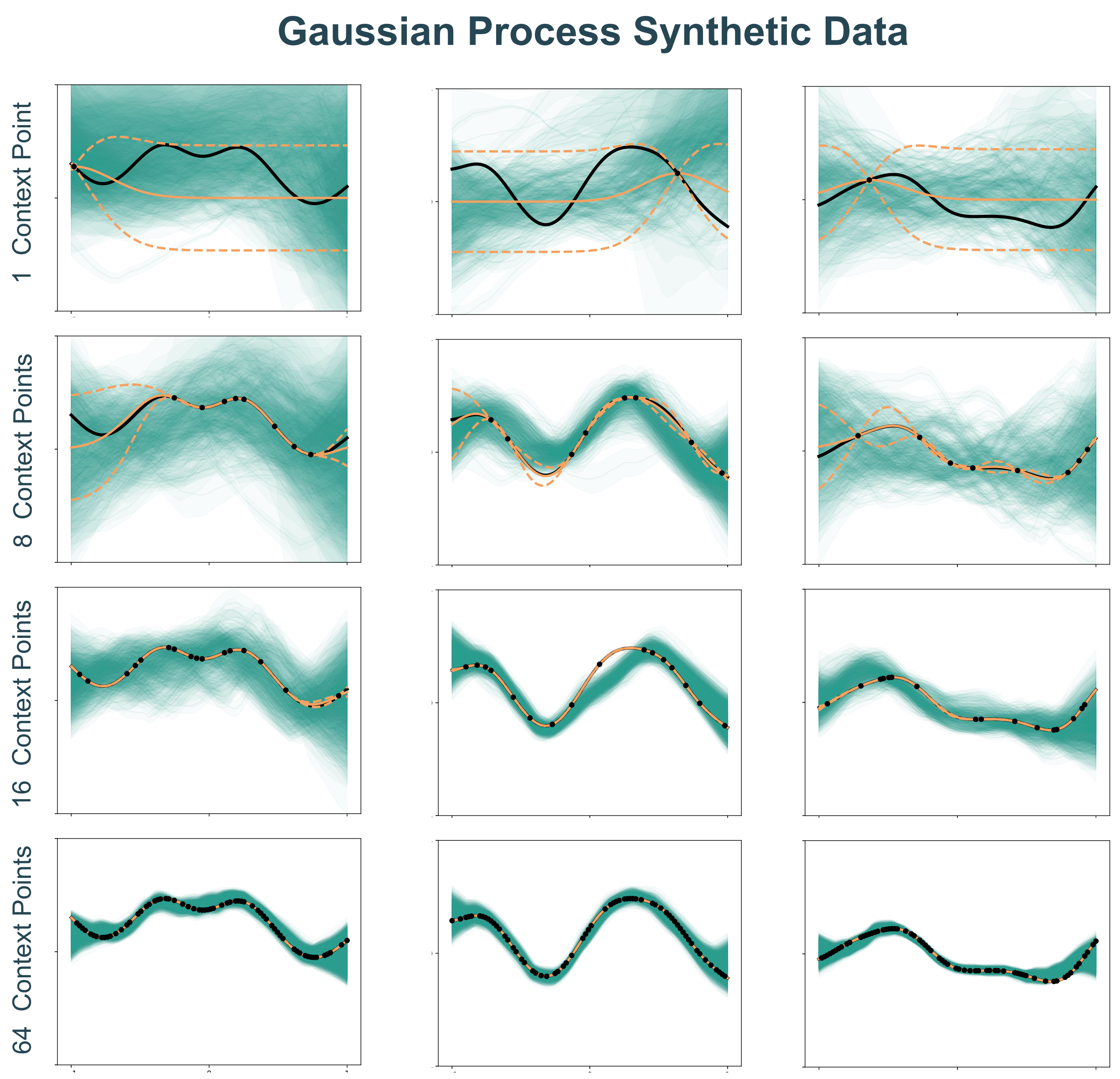
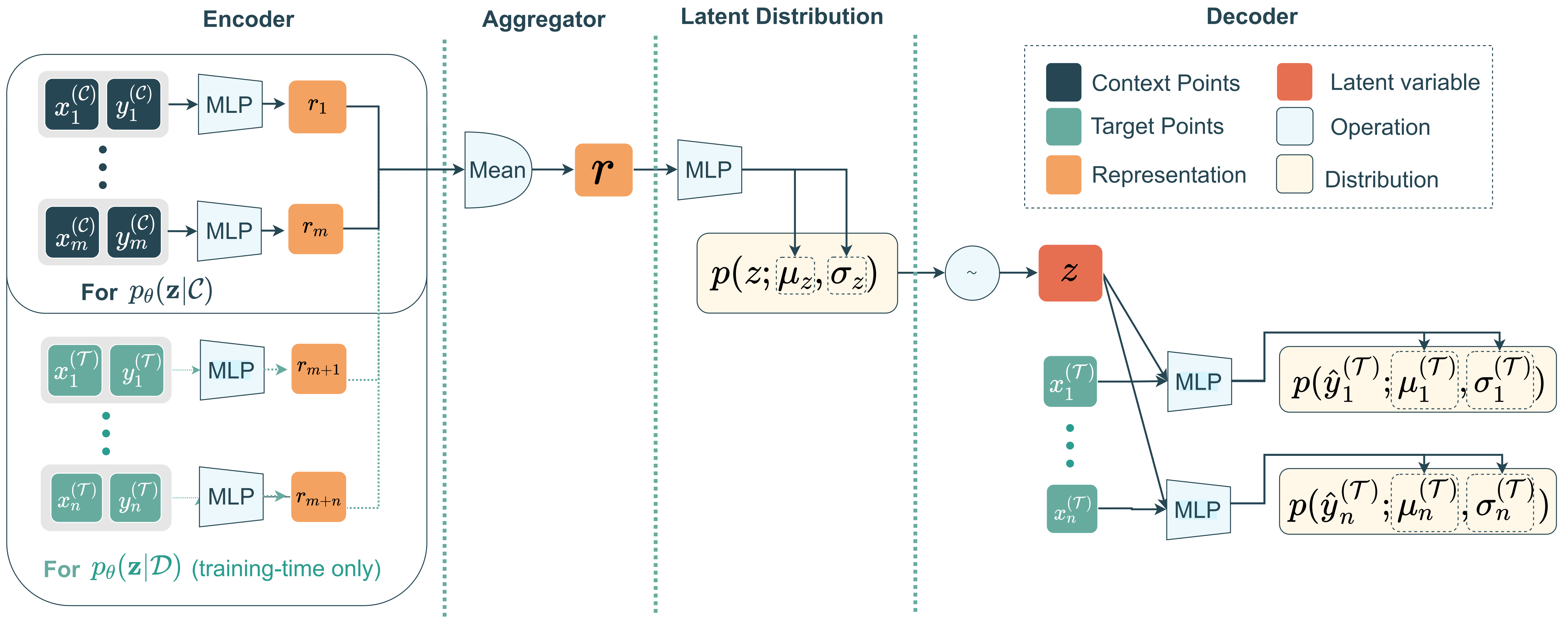

Gaussian Process Synthetic Data

Figure 1: NPs are applied to a 1-D function regression task. The data for this experiment are generated using a GP with an RBF kernel with varying length-scale and scale parameters for each function. In training, a number of context and target points for each generated function are then selected and passed to the network. When visualizing, for the same underlying ground truth curve (black line), several samples (light-blue lines) are generated from an NP using varying numbers of context points. As more context points are observed, the fit improves and the variance decreases. A GP model with RBF kernel is fit on the same context points for comparison (solid orange line - mean, dotted orange line - 95% prediction interval).

Architecture


Encoder

Aggregator

Latent Distribution

Decoder

Context Points

Latent variable

Target Points

Operation

Representation

Distribution

For $p_\theta(\mathbf{z}|\mathcal{C})$

For $p_\theta(\mathbf{z}|\mathcal{D})$ (training-time only)

$x_1^{(\mathcal{C})}$ $y_1^{(\mathcal{C})}$
MLP
r_1
$x_m^{(\mathcal{C})}$ $y_m^{(\mathcal{C})}$
r_m
$x_1^{(\mathcal{T})}$ $y_1^{(\mathcal{T})}$
r_{m+1}
$x_n^{(\mathcal{T})}$ $y_n^{(\mathcal{T})}$
r_{m+n}
Mean
r
$p(z;\mu_z,\sigma_z)$
~
z
$x_1^{(\mathcal{T})}$
$x_n^{(\mathcal{T})}$
$p(\hat{y}_1^{(\mathcal{T})};\mu_1^{(\mathcal{T})},\sigma_1^{(\mathcal{T})})$
$p(\hat{y}_n^{(\mathcal{T})};\mu_n^{(\mathcal{T})},\sigma_n^{(\mathcal{T})})$

2D Function Regression

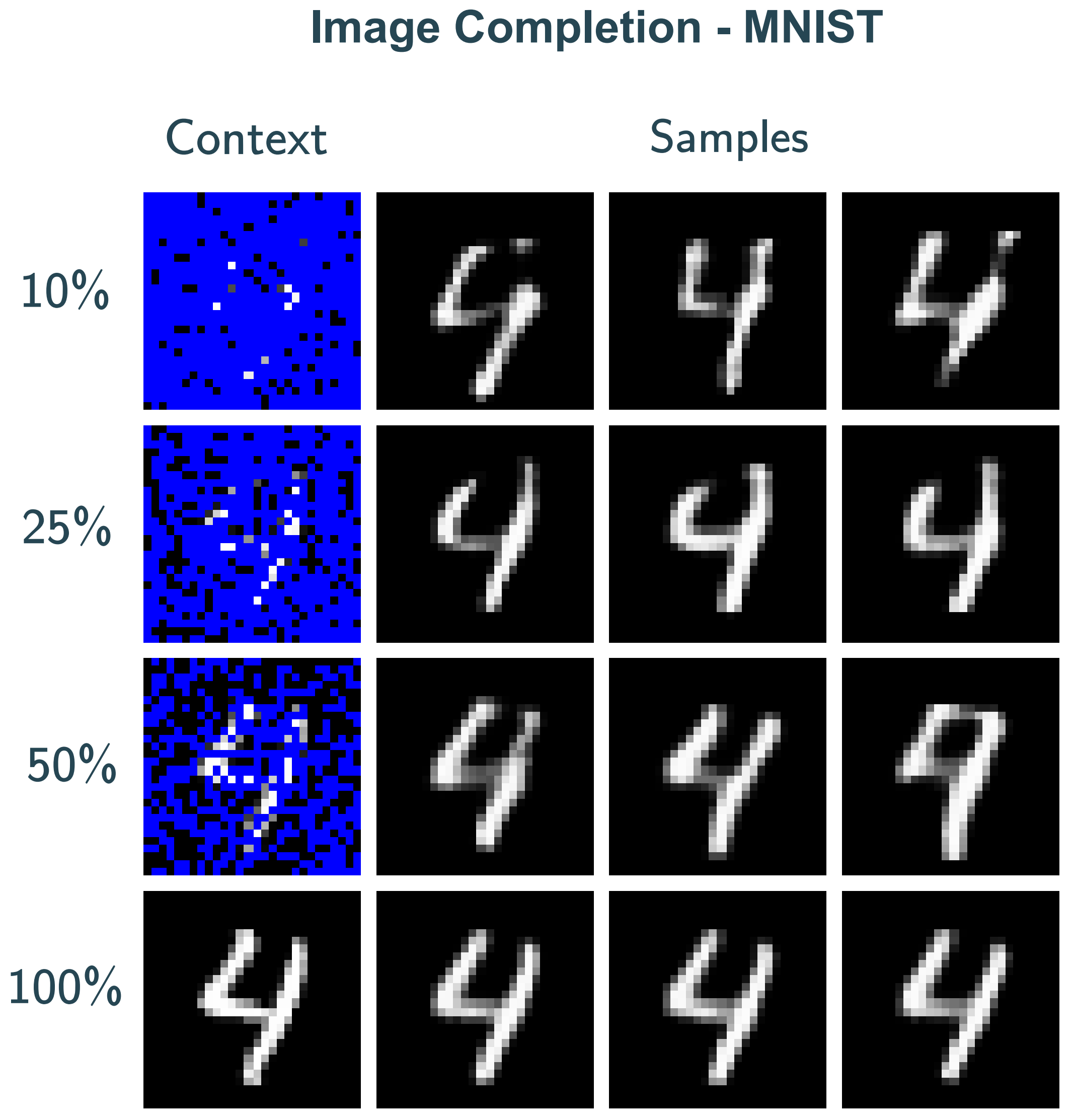


Image Completion - MNIST

Figure 2: NPs are applied to a 2-D regression task, a pixel-wise image completion on MNIST dataset. The uncertainty is reflected in the variability of the generated samples. In this example, the samples begin to resemble the ground truth image with around 25% of context points.

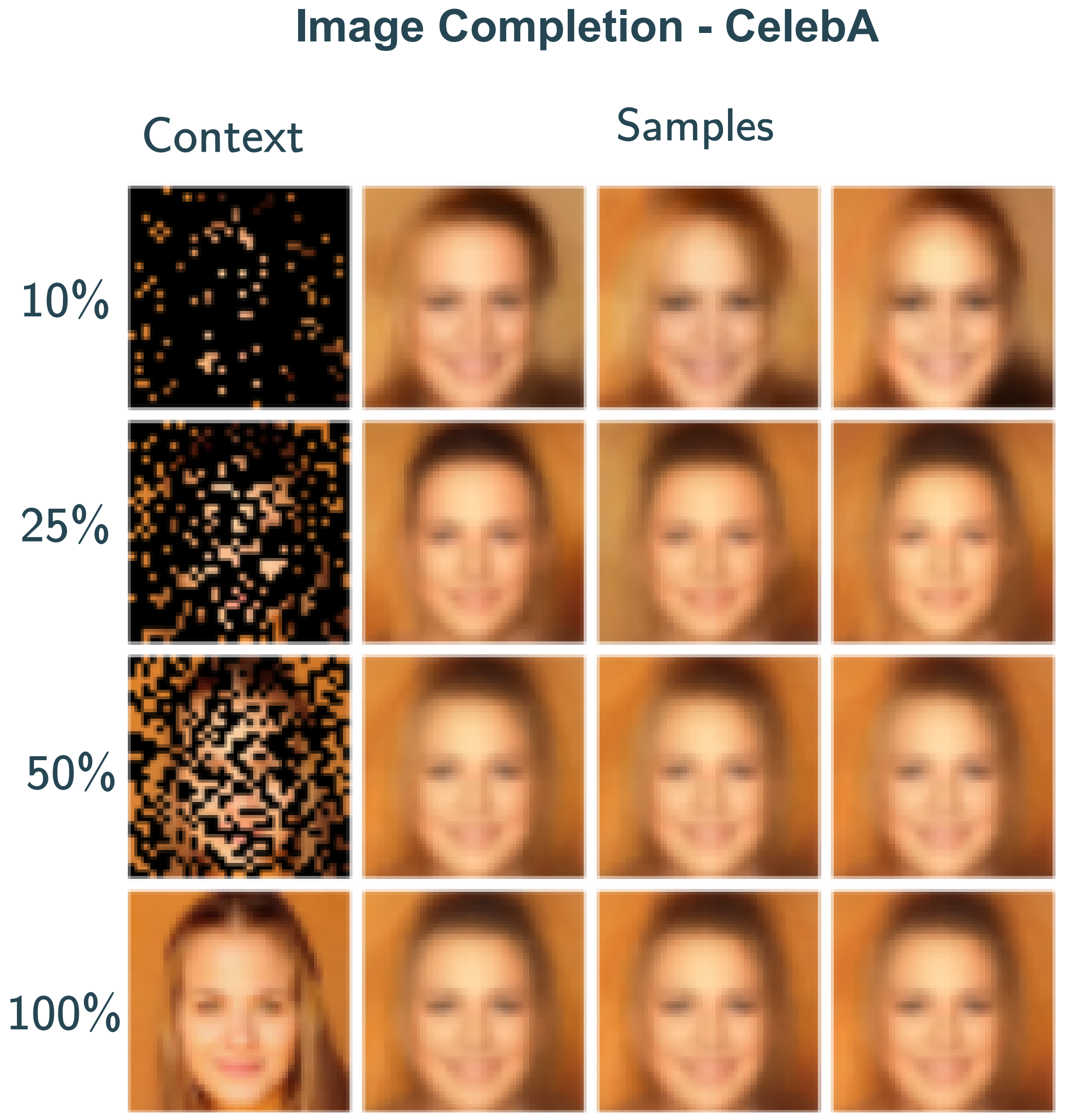


Image Completion - CelebA

Figure 3: NPs are tested on the face-completion task on the CelebA dataset. As they observe more context points, it converges towards very similar looking faces. Note that, even when 100% of the pixels are provided as context points, NPs cannot generate samples identical to the ground truth, since the latent variable z constitutes both a narrow bottleneck and a probabilistic sampling operation.

Thompson Sampling

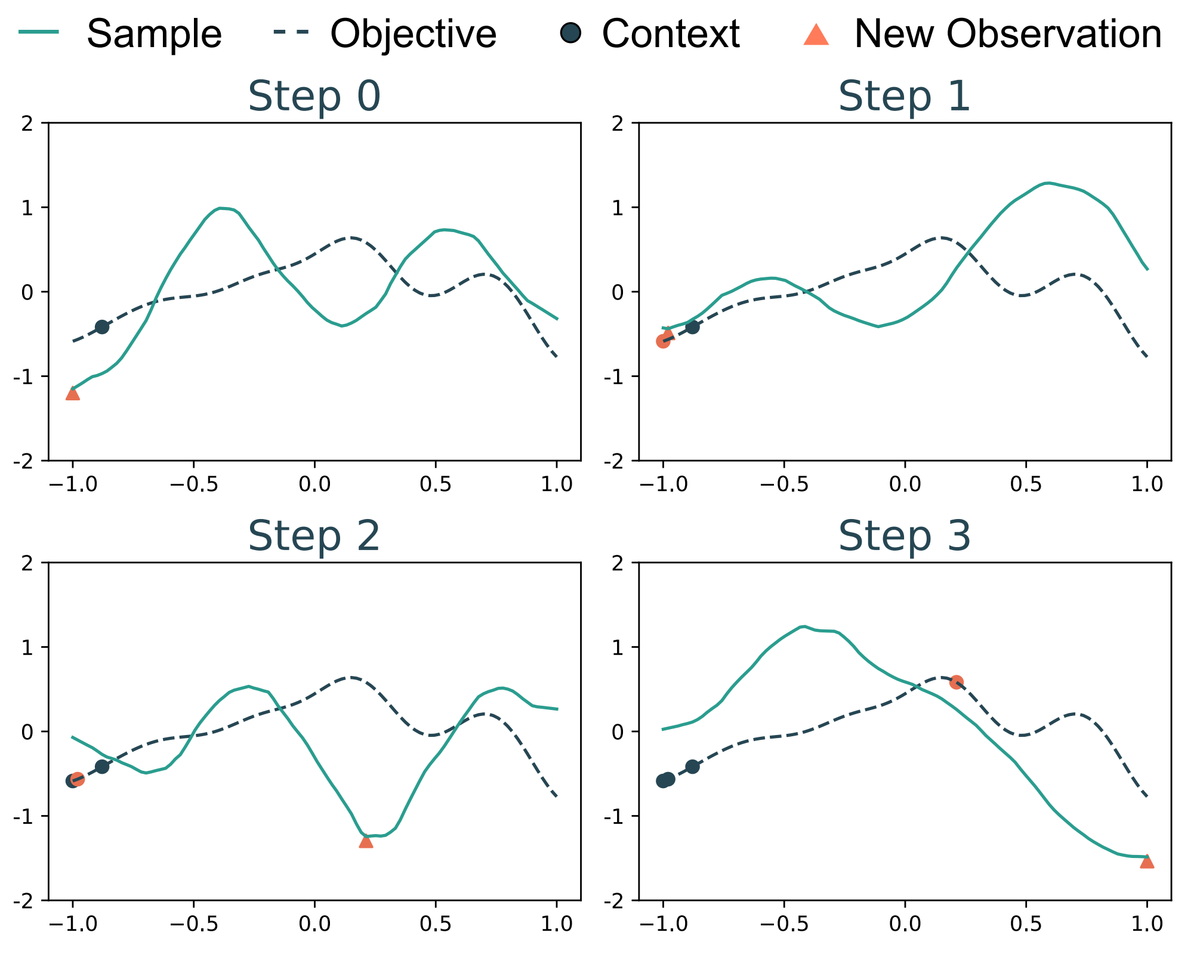


Figure 4: Bayesian optimization of a 1-D function using NPs for Thompson sampling. Context points acquired by sampling functions and evaluating the objective at the sample's minimum.

Neural Process	Gaussian Process	Random Search
0.37	0.17	1.00

Table 1: Empirical comparison of Thompson sampling for an objective sampled from a GP using NPs, GPs, and random search. The table shows the mean number of optimization steps needed (across 5000 objectives), normalized by the random search mean. GPs' samples using the same kernel as the generating GP provide a lower bound on optimization step count.

Limitations

The experiments on different tasks showed NPs' data efficiency, high flexibility, and applicability to different domains, combining the benefits of both NNs and GPs.

However, there is evidence of underfitting for complex tasks.

One possibility is that the approximate variational inference training techniques are known to have drawbacks including systematic biases [3]. Other approximate inference techniques may be more effective, such as directly approximating the likelihood with monte-carlo sampling [4].

Another possibility is that NPs do not maintain translation equivariance. Using standard Convolutional layers is not sufficient as the context sets used during training are of variable length, observed at irregular intervals and are mapped to a fixed dimensional representation. ConvNP's [4] are an extension of NPs which address these challenges by learning a translation-equivariant representation using convolutional deep sets.

References

[1] **Neural Processes:** Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D.J., Eslami, S.M. and Teh, Y.W. Neural processes. ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models 2018.

[2] **Conditional Neural Processes:** Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y.W., Rezende, D. and Eslami, S.A., 2018, July. Conditional neural processes. In International Conference on Machine Learning (pp. 1704-1713). PMLR.

[3] **Two problems with variational expectation maximisation for time series models:** Turner, R. E. and Sahani, M. (2011) "Two problems with variational expectation maximisation for time series models," In Barber, D., Cemgil, A. T., and Chiappa, S. (eds.), Bayesian Time Series Models, chapter, Cambridge, Cambridge University Press, pp. 104–124.

[4] **Convolutional Neural Processes:** Andrew Y. K. Foong, Wessel P. Bruinsma, Jonathan Gordon, Yann Dubois, James Requeima, Richard E. Turner, "Meta-Learning Stationary Stochastic Process Prediction with Convolutional Neural Processes" NeurIPS 2020 arXiv:2007.01332 [stat.ML]