1. Conditional Neural Processes

2. Neural Processes

Marta Garnelo [1] Jonathan Schwarz [1] Dan Rosenbaum [1] Fabio Viola [1] Danilo J. Rezende [1] S. M. Ali Eslami [1]
Yee Whye Teh [1]

ICML-2018

Introduction

- supervised problems
  - a **parametric function** g

    Prior info: architecture of g, the loss function, or the training details
  - a **probabilistic** stance： stochastic processes

    Prior info: distributional assumptions about the prior process

- Gaussian Processes

  Prior info: a parametric kernel function

  computationally intractable

  difficult to design appropriate priors

- Motivation

  combine neural networks with features reminiscent of Gaussian Processes

Gaussian Process

■ 定义

是观测值出现在一个连续域的随机过程。在高斯过程中，连续输入空间中每个点都是与一个正态分布的随机变量相关联。此外，这些随机变量的每个有限集合都有一个多元正态分布。

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} \sim \mathcal{N}(\mu, \Sigma)$$

$$P_{X,Y} = \begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N}(\mu, \Sigma) = \mathcal{N}\left(\begin{bmatrix} \mu_X \\ \mu_Y \end{bmatrix}, \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix} \right)$$

$$p_X(x) = \int_y p_{X,Y}(x, y)dy = \int_y p_{X|Y}(x|y)p_Y(y)dy$$

$$\Sigma = \mathrm{Cov}(X_i, X_j) = E\left[(X_i - \mu_i)(X_j - \mu_j)^T\right]$$

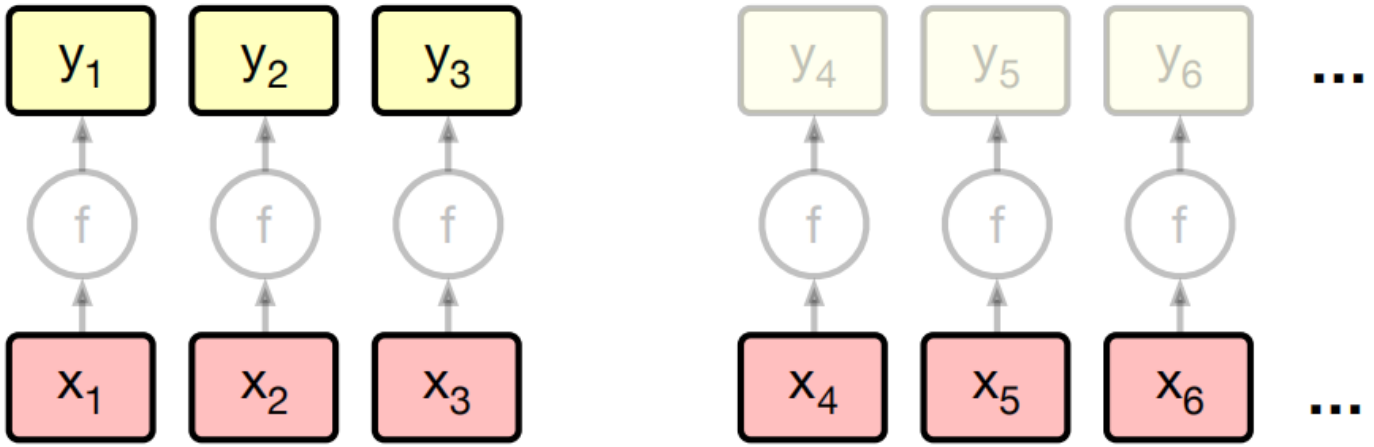
$$X|Y \sim \mathcal{N}(\,\mu_X + \Sigma_{XY}\Sigma_{YY}^{-1}(Y - \mu_Y),\ \Sigma_{XX} - \Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX}\,)$$
$$Y|X \sim \mathcal{N}(\,\mu_Y + \Sigma_{YX}\Sigma_{XX}^{-1}(X - \mu_X),\ \Sigma_{YY} - \Sigma_{YX}\Sigma_{XX}^{-1}\Sigma_{XY}\,)$$

$$\mathcal{O}((n+m)^3)$$

https://www.jgoertler.com/visual-exploration-gaussian-processes/

Problem Setting

- Stochastic Processes



$$O = \{(x_i, y_i)\}_{i=0}^{n-1} \subset X \times Y \qquad T = \{x_i\}_{i=n}^{n+m-1} \subset X$$

Assume that the outputs are a realization of the following process

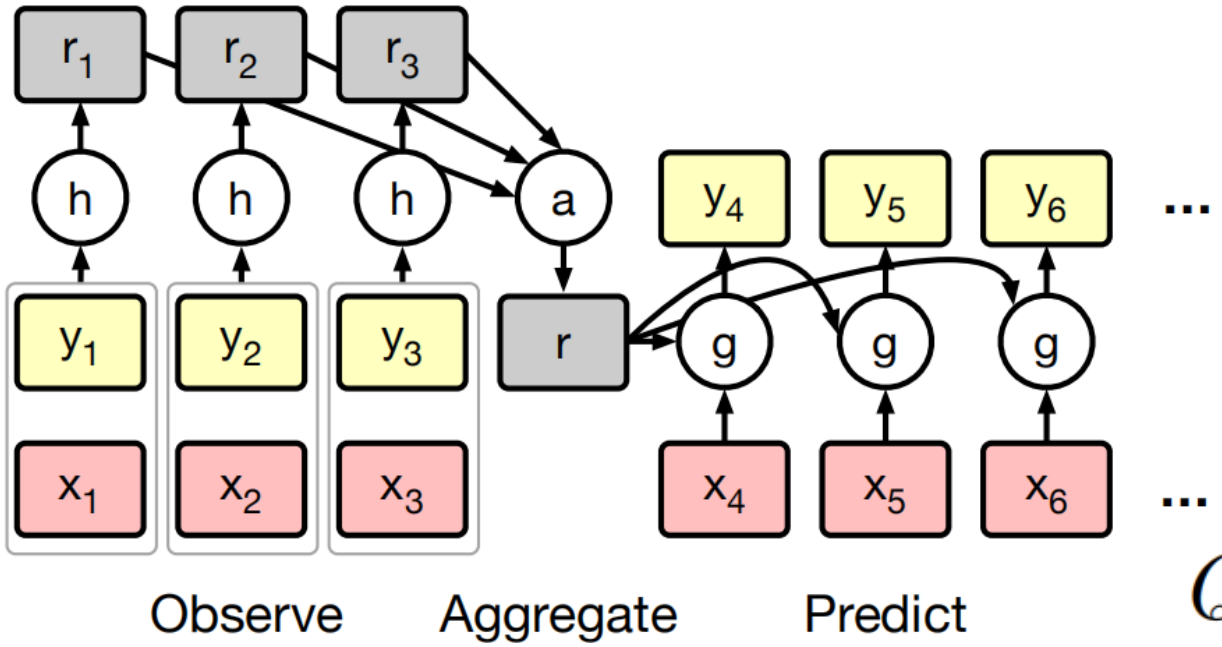
P defines a joint distribution over functions $f: X \to Y$. For $f \sim P$ set $y_i = f(x_i)$

The task is to predict the output values $f(x)$ for every $x \in T$ given O

Conditional Neural Processes (CNPs)

Directly parametrize conditional stochastic processes without imposing consistency with respect to some prior process.

CNPs parametrize distributions over $f(T)$ given a distributed representation of O of fixed dimensionality



$$r_i = h_\theta(x_i, y_i) \qquad \forall (x_i, y_i) \in O$$

$$r = r_1 \oplus r_2 \oplus \ldots r_{n-1} \oplus r_n$$

$$\phi_i = g_\theta(x_i, r) \qquad \forall (x_i) \in T$$

$$Q_\theta(f(x_i) \mid O, x_i) = Q(f(x_i) \mid \phi_i)$$

Conditional Neural Processes (CNPs)

- Regression

  Use $\phi_i$to parametrize the mean and variance $\phi_i = (\mu_i, \sigma_i^2)$ of a Gaussian distribution $N(\mu_i, \sigma_i^2)$ for every $x_i \in T$

- Classification

  Use ϕ_i to parametrize the logits of the class probabilities p_c over the c classes of a categorical distribution

■ Training CNPs

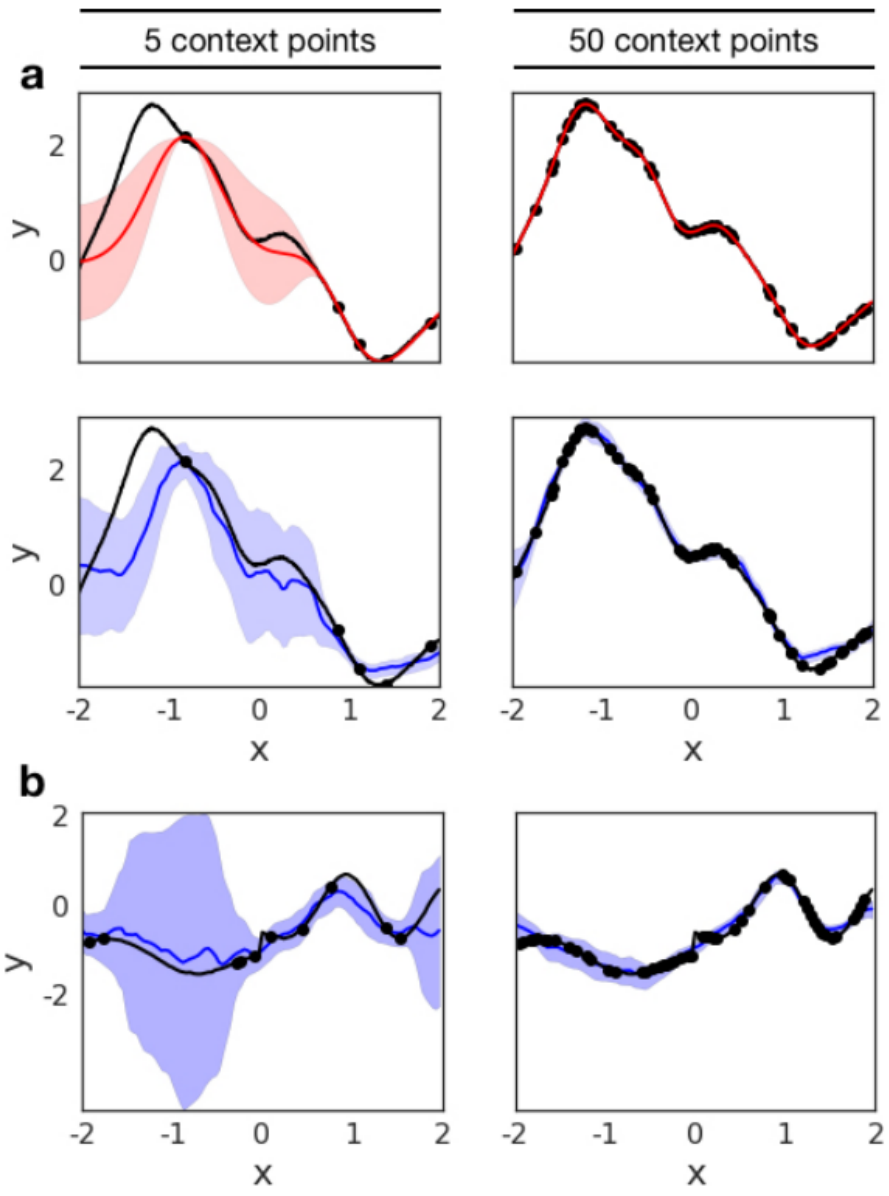
$$f \sim P, O = \{(x_i, y_i)\}_{i=0}^{n-1} \quad N \sim \text{uniform}[0, \ldots, n-1]$$

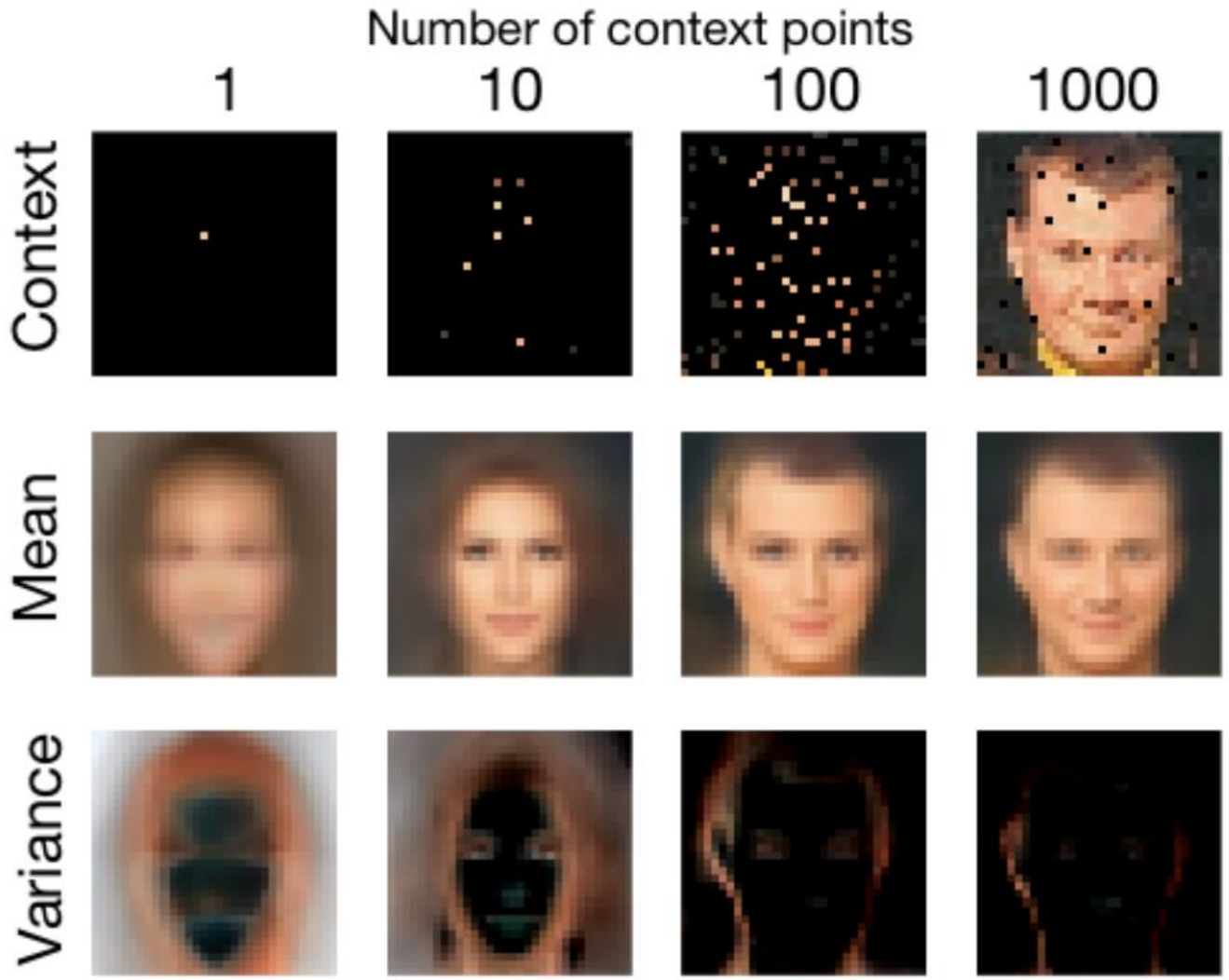
condition on the subset $O_N = \{(x_i, y_i)\}_{i=0}^{N} \subset O$

$$\mathcal{L}(\theta) = -\mathbb{E}_{f\sim P}\left[\mathbb{E}_N\left[\log Q_\theta(\{y_i\}_{i=0}^{n-1} | O_N, \{x_i\}_{i=0}^{n-1})\right]\right]$$

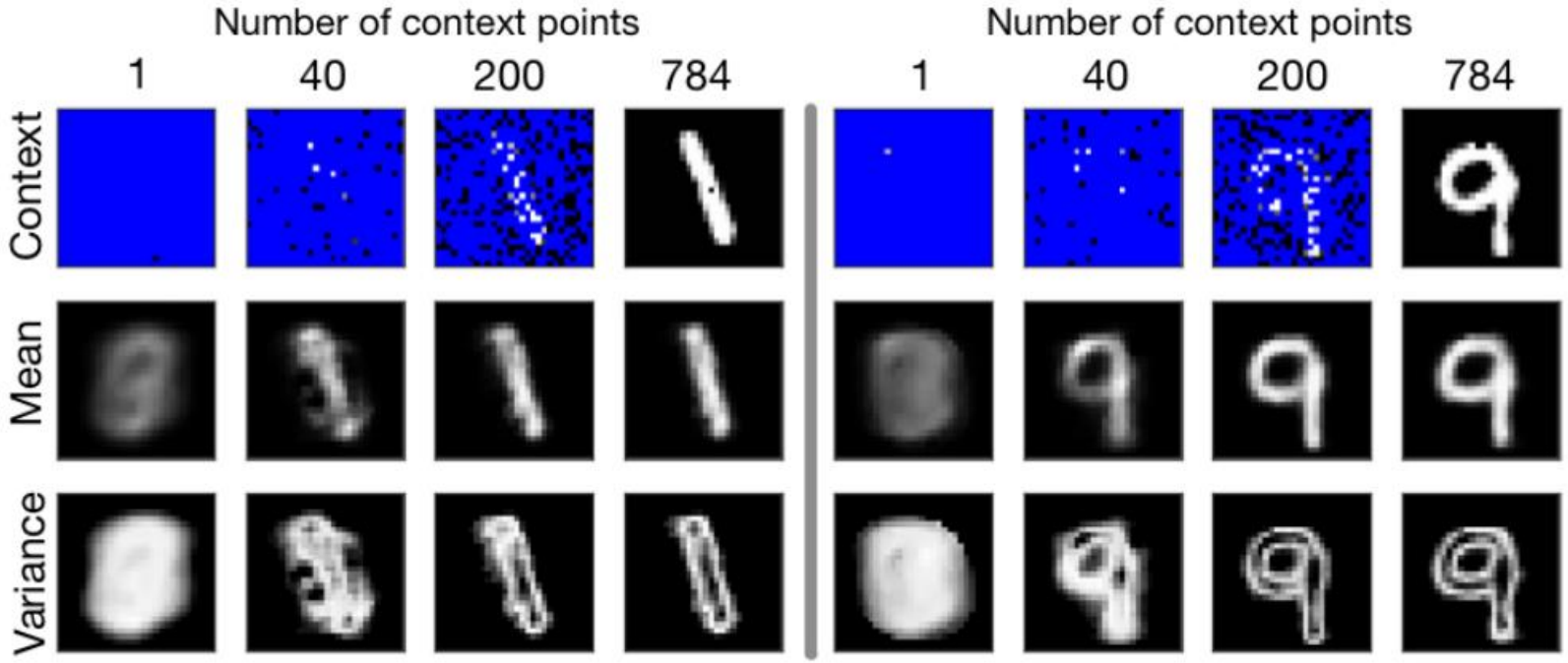
take Monte Carlo estimates of the gradient of this loss by sampling f and N

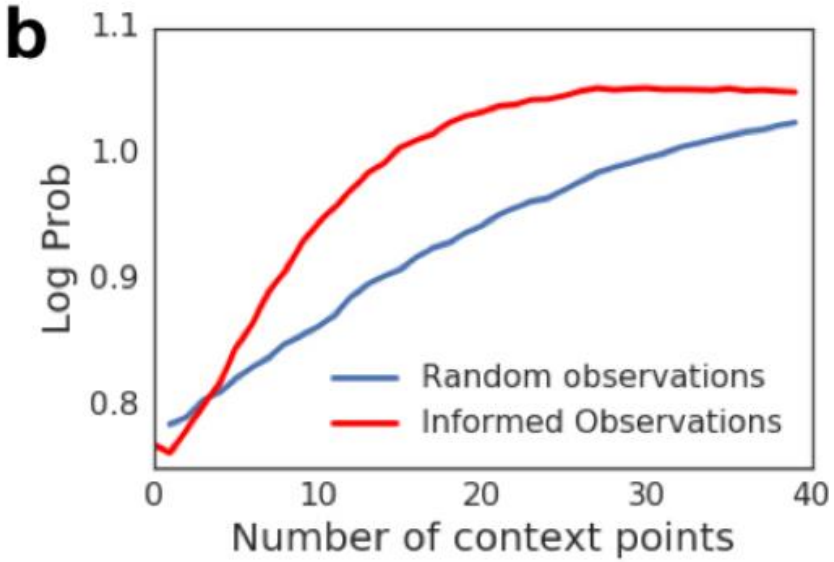
Experimental



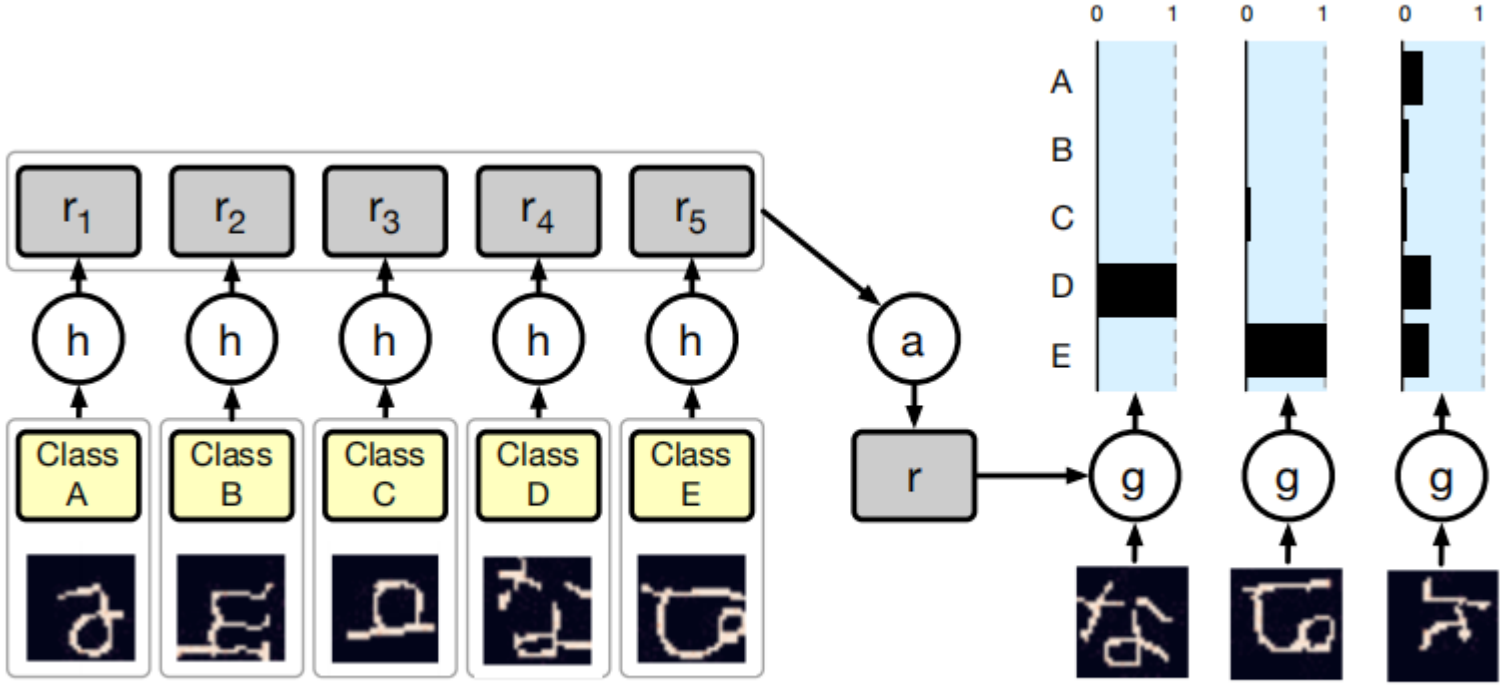


Experimental





Experimental



	5-way Acc		20-way Acc		Runtime
	1-shot	5-shot	1-shot	5-shot	
MANN	82.8%	94.9%	-	-	$\mathcal{O}(nm)$
MN	**98.1%**	**98.9%**	**93.8%**	**98.5%**	$\mathcal{O}(nm)$
CNP	95.3%	98.5%	89.9%	96.8%	$\mathcal{O}(n+m)$

Neural Processes

Define a stochastic process F such that

$\rho_{x_{1:n}}$ is the marginal distribution of

$$(F(x_1), \ldots, F(x_n))$$

exchangeability and consistency

$$y_n = f(x_n) + \epsilon_n$$

$$y|\mathrm{f}(x) \sim \mathcal{N}(y|\mathrm{f}(x), \beta^{-1} I_N)$$

$$p(y|x) = \int p(y|\mathrm{f}(x))p(\mathrm{f})d\mathrm{f}$$

Given a particular instantiation of the stochastic process f the joint distribution is defined as:

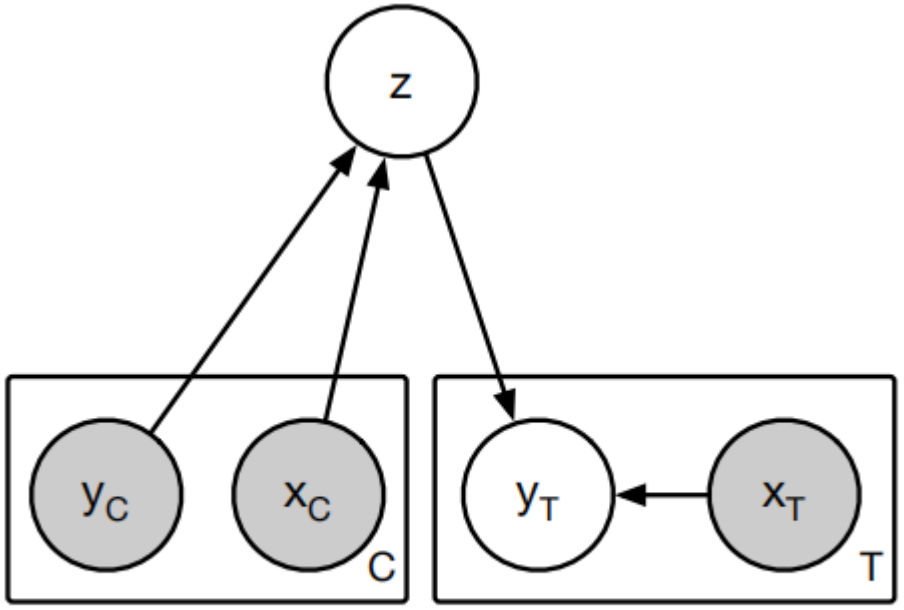
$$\rho_{x_{1:n}}(y_{1:n}) = \int p(f)p(y_{1:n}|f, x_{1:n})df.$$

$$\rho_{x_{1:n}}(y_{1:n}) = \int p(f) \prod_{i=1}^{n} \mathcal{N}(y_i|f(x_i), \sigma^2)df$$

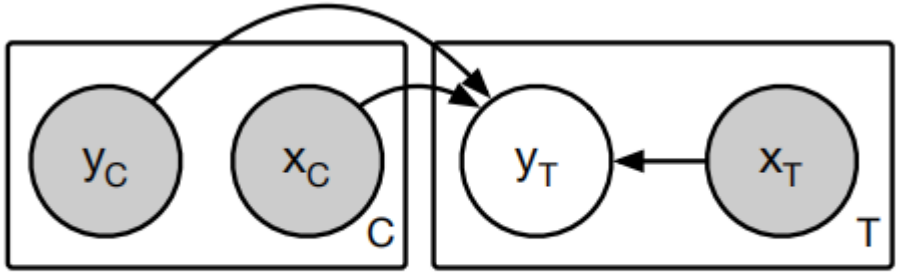
$$p(y_{1:n}|f, x_{1:n}) = \prod_{i=1}^{n} \mathcal{N}(y_i|f(x_i), \sigma^2)$$

$$F(x) = g(x, z)$$

Neural Processes



(d) Neural process



(c) Conditional neural process

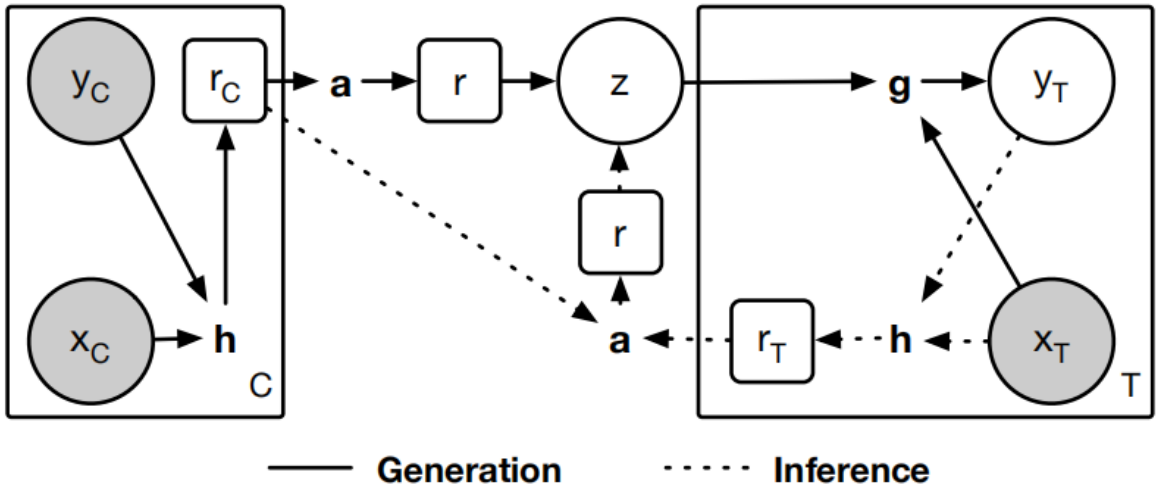
$$\rho_{x_{1:n}}(y_{1:n}) = \int p(f) \prod_{i=1}^{n} \mathcal{N}(y_i | f(x_i), \sigma^2) df$$

$$F(x) = g(x, z)$$

$$p(z, y_{1:n} | x_{1:n}) = p(z) \prod_{i=1}^{n} \mathcal{N}(y_i | g(x_i, z), \sigma^2)$$

$$p(y_{1:n} | x_{1:n}) = \sum_{z} p(z, y_{1:n} | x_{1:n})$$

Neural Processes



(b) Computational diagram

h: An encoder $r_i = h((x, y)_i)$

a: An aggregator $r = a(r_i) = \frac{1}{n}\sum_{i=1}^{n} r_i$

g: conditional decoder $g(x_t, z) \sim f(x_t) = y_t$

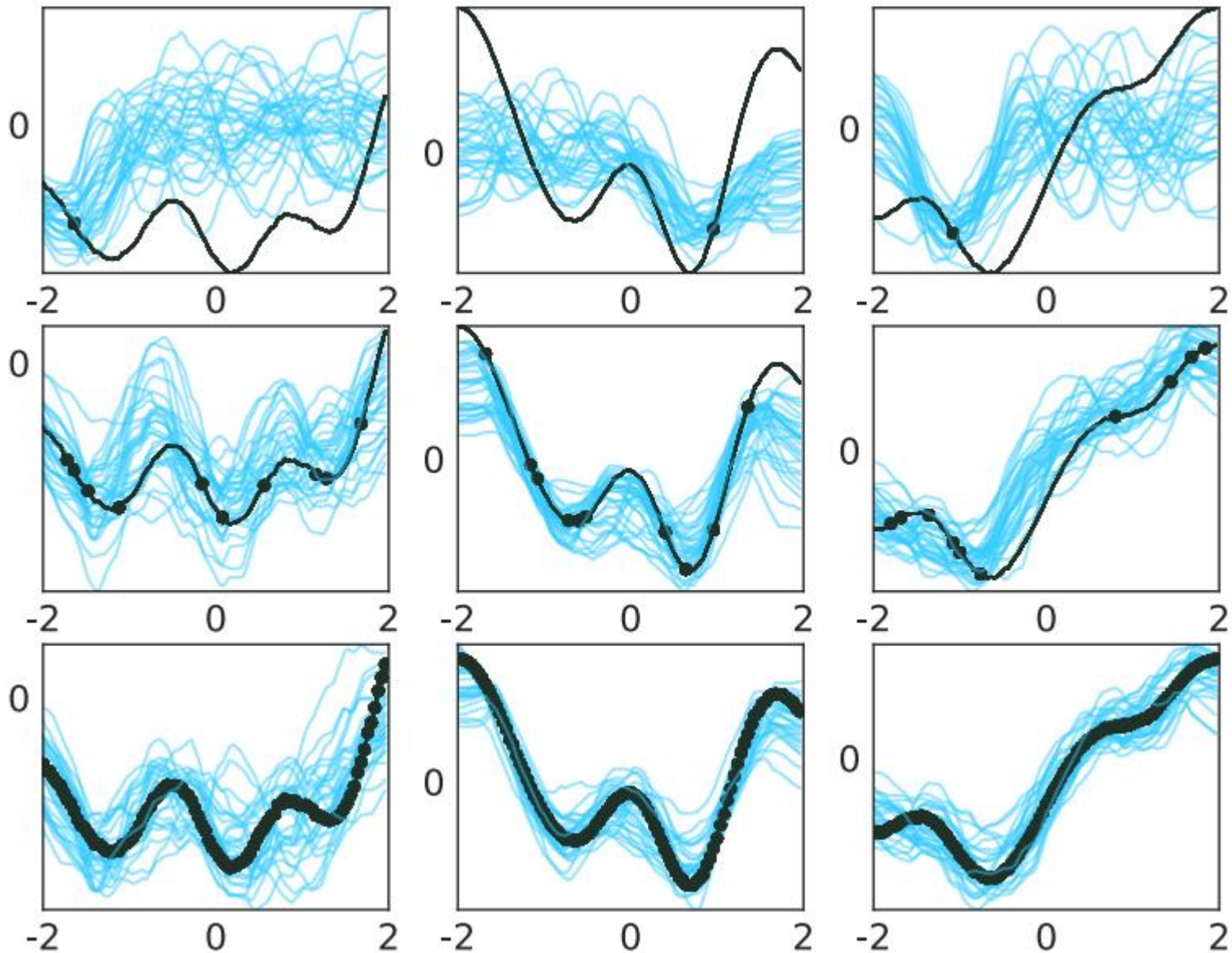
$$\log p(y_{1:n}|x_{1:n}) \tag{7}$$

$$\geq \mathbb{E}_{q(z|x_{1:n},y_{1:n})}\left[\sum_{i=1}^{n} \log p(y_i|z, x_i) + \log \frac{p(z)}{q(z|x_{1:n}, y_{1:n})}\right]$$

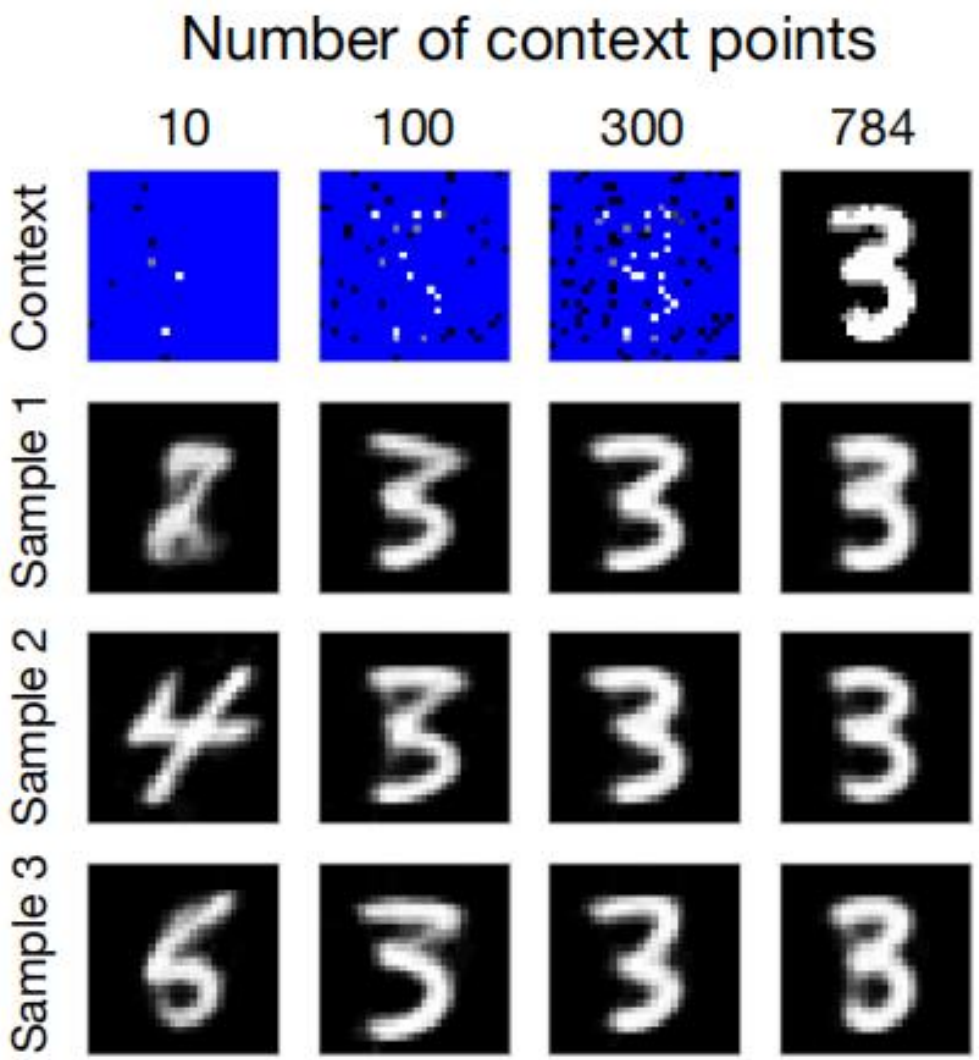
$$\log p(y_{m+1:n}|x_{1:n}, y_{1:m})$$

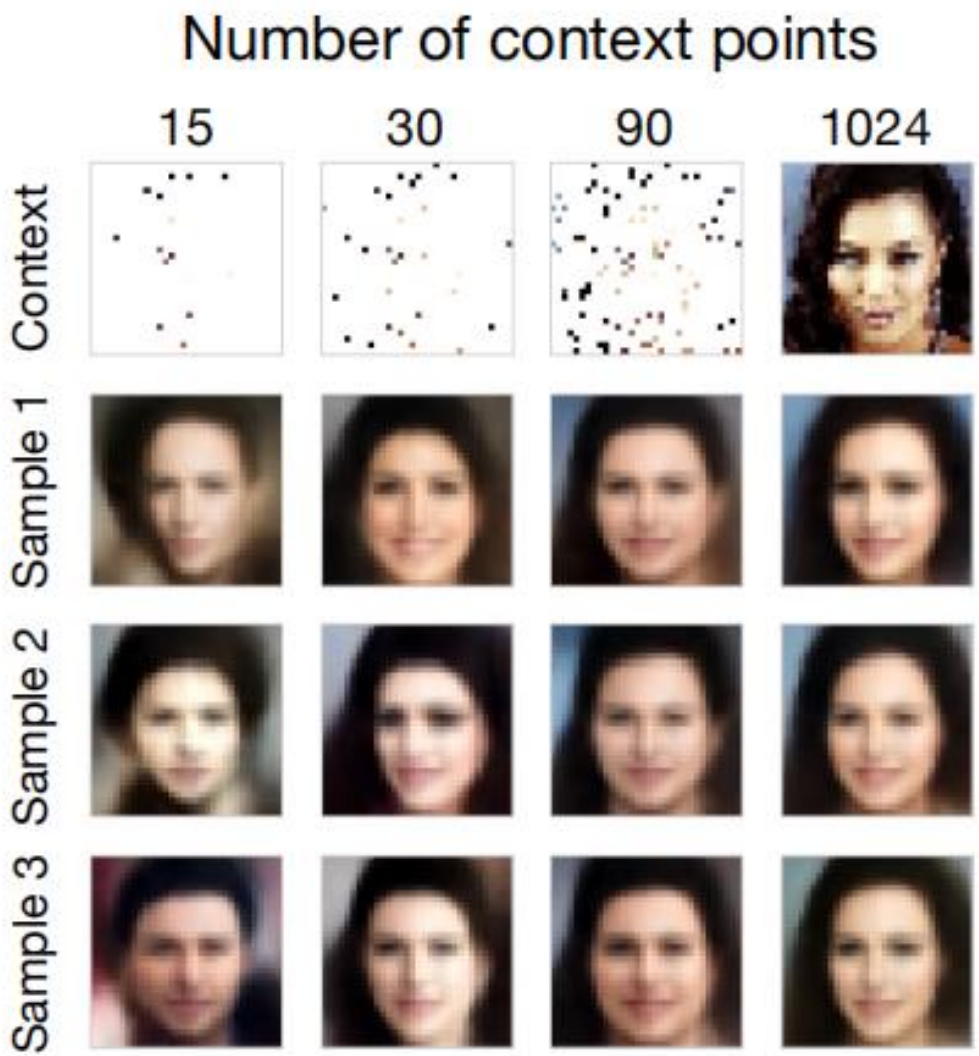
$$\geq \mathbb{E}_{q(z|x_{1:n},y_{1:n})}\left[\sum_{i=m+1}^{n} \log p(y_i|z, x_i) + \log \frac{p(z|x_{1:m}, y_{1:m})}{q(z|x_{1:n}, y_{1:n})}\right]$$

Experimental



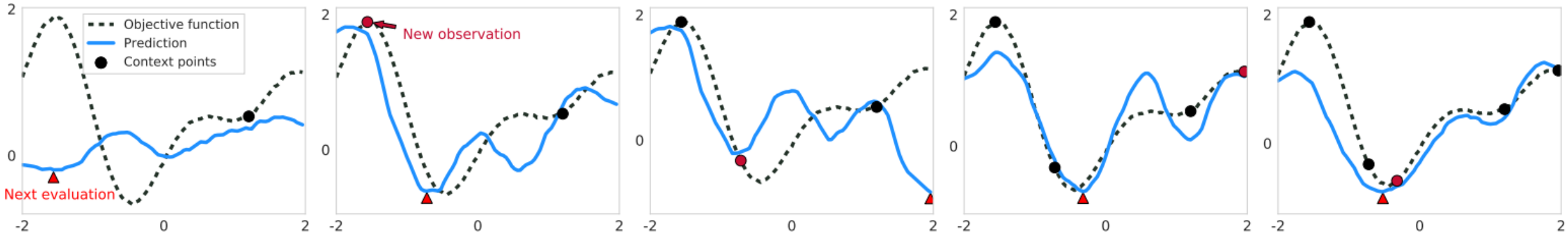
Experimental





Experimental

Black-box optimisation with Thompson sampling



Neural process	Gaussian process	Random Search
0.26	0.14	1.00

Average number of optimisation steps needed to reach the global minimum

Experimental