Empirical Evaluation of Neural Process Objectives

Tuan Anh Le[1*] tuananh@robots.ox.ac.uk
Hyunjik Kim[1,2] hyunjikk@google.com
Marta Garnelo[2] garnelo@google.com
Dan Rosenbaum[2] danro@google.com
Jonathan Schwarz[2] schwarzjn@google.com
Yee Whye Teh[2] ywteh@google.com

Abstract

Neural processes (NPs) [3, 4] are parametric stochastic processes that can be trained from a dataset consisting of sets of input-output pairs. During test time, given a context set of input-output pairs and a set of target inputs, they allow us to approximate the posterior predictive of the target outputs. NPs have shown promise in applications such as image super-resolution, conditional image generation or scalable Bayesian optimization. It is, however, unclear which objective and model specification should be used to train NPs. This abstract empirically evaluates the performance of NPs for different objectives and model specifications. Given that some objectives and model specifications clearly outperform others, our analysis can be useful in guiding future research and applications of NPs.

1 Introduction

A NP [3, 4] describes a stochastic process (random function). To learn an NP, we use a distribution over datasets. The idea being that each dataset corresponds to a single instantiation of the random function, and the distribution over datasets allows us to learn the distribution of the random function. At each iteration of learning, one dataset $\{(x_i, y_i)\}_{i=1}^n$ (or a minibatch thereof) is presented to learn from. We assume that this dataset is split into a context set $\{(x_i, y_i)\}_{i=1}^m$ of size $m \leq n$, and a target set which consists of all n data pairs. Let $x_\mathcal{C}, y_\mathcal{C}, x_\mathcal{T}, y_\mathcal{T}$ denote the inputs and outputs of the context and target respectively, with $\mathcal{C} = \{1, \dots, m\}$ and $\mathcal{T} = \{1, \dots, n\}$. There are multiple objectives and model specifications that one can use to learn the NP, and our aim is to evaluate these choices empirically.

2 Objectives and Model Specifications

We investigate the following set of objectives and model specifications:

$$\Big(\underbrace{\{[T|\varnothing], [C|\varnothing], [T|C]\}}_{\text{objectives with a latent variable}} \times \overbrace{\{\text{no det}, \text{det}, \text{att}\}}^{\text{deterministic path and attention}} \cup \underbrace{\{\text{det}, \text{det+att}\}}_{\text{objectives without a latent variable}}\Big) \times \overbrace{\begin{Bmatrix}\text{learned var},\\ \text{fixed var}\end{Bmatrix}}^{\text{observation variance}} \times \underbrace{\begin{Bmatrix}(\downarrow m, \uparrow n),\\ (\uparrow m, \uparrow n),\\ (\updownarrow m, \updownarrow n)\end{Bmatrix}}_{\text{training distribution of } (m,n)},$$

which we describe below.

Joint distribution objectives with latent variable. The simplest way to describe a random function is by using a latent random variable z. We then parameterize the neural process as a deterministic

[*]Work completed during a Deepmind internship. [1]University of Oxford, [2]Deepmind.

Third workshop on Bayesian Deep Learning (NeurIPS 2018), Montréal, Canada.

function $f_z : \mathcal{X} \to \mathcal{M}_1(\mathcal{Y})$ which depends on z. $\mathcal{M}_1(\mathcal{Y})$ is the space of distributions (probability measures) over $\mathcal{Y}$. The randomness in z induces the randomness in f_z. The random function maps each input $x \in \mathcal{X}$ into a distribution over the corresponding output $y \in \mathcal{Y}$. Note that we assume that outputs are conditionally independent given the function f_z, e.g. in case of iid normal observation noise.

The simplest case is when the target and context sets are the same, $n = m$, and we use a standard variational evidence lower bound to learn the NP. Specifically, we aim to optimize objective $[T|\varnothing]$:

$$[T|\varnothing]: \ \log p(y_\mathcal{T}|x_\mathcal{T}) \geq \int \left(\left(\sum_{i=1}^{n} \log p(y_i|f_z(x_i)) \right) + \log \frac{p(z)}{q(z|x_\mathcal{T}, y_\mathcal{T})} \right) q(z|x_\mathcal{T}, y_\mathcal{T}) dz \quad (1)$$

where $p(z)$ is the prior, and $q(z|x_\mathcal{T}, y_\mathcal{T})$ is a variational posterior (encoder) over the latent variable z. This is the supervised learning equivalent of the neural statistician (NS) [2].

The second approach takes the same objective $\log p(y_\mathcal{T}|x_\mathcal{T})$, but lower bounds it using a variational posterior that only depends on the context $x_\mathcal{C}, y_\mathcal{C}$, leading to objective $[C|\varnothing]$:

$$[C|\varnothing]: \ \log p(y_\mathcal{T}|x_\mathcal{T}) \geq \int \left(\left(\sum_{i=1}^{n} \log p(y_i|f_z(x_i)) \right) + \log \frac{p(z)}{q(z|x_\mathcal{C}, y_\mathcal{C})} \right) q(z|x_\mathcal{C}, y_\mathcal{C}) dz \quad (2)$$

The idea here is that at test time we are interested in evaluating the NP by evaluating its generalization from a context set to a target set, so we should also train it under the same context/target regime. This corresponds to the supervised version of the variational homoencoder (VHE) [5].

Conditional distribution objective with latent variable. Instead of modelling the joint distribution of all outputs, we can model the conditional of the target given the context instead, leading to objective $[T|C]$:

$$\begin{aligned}[T|C]: \log p(y_\mathcal{T}|x_\mathcal{T}, x_\mathcal{C}, y_\mathcal{C}) &\geq \int \left(\left(\sum_{i=1}^{n} \log p(y_i|f_z(x_i)) \right) + \log \frac{p(z|x_\mathcal{C}, y_\mathcal{C})}{q(z|x_\mathcal{T}, y_\mathcal{T})} \right) q(z|x_\mathcal{T}, y_\mathcal{T}) dz \\ &\approx \int \left(\left(\sum_{i=1}^{n} \log p(y_i|f_z(x_i)) \right) + \log \frac{q(z|x_\mathcal{C}, y_\mathcal{C})}{q(z|x_\mathcal{T}, y_\mathcal{T})} \right) q(z|x_\mathcal{T}, y_\mathcal{T}) dz \quad (3)\end{aligned}$$

The last line is an approximation as the true conditional $p(z|x_\mathcal{C}, y_\mathcal{C})$ is intractable.

Conditional distribution objective without latent variable. Rather than using latent variables to define a stochastic process (random function), we can simply directly model conditional distributions of targets given contexts. The simplest approach is as follows: let $r(x_\mathcal{C}, y_\mathcal{C})$ be a deterministic function of the context $x_\mathcal{C}, y_\mathcal{C}$. Then we learn by optimizing the objective "det" which is used in the conditional NP [3]:

$$\text{det}: \ \log p(y_\mathcal{T}|x_\mathcal{T}, x_\mathcal{C}, y_\mathcal{C}) = \sum_{i=1}^{n} \log p(y_i|f_{r(x_\mathcal{C}, y_\mathcal{C})}(x_i)) \quad (4)$$

where $f_{r(x_\mathcal{C}, y_\mathcal{C})}(x)$ is simply a function of both the representation of the context and the input x.

Objectives with latent variable and deterministic path. The deterministic approach loses dependence among the target outputs induced by the latent variable z. We can re-introduce the latent variable z, but now think of z not as capturing the stochasticity in the whole stochastic process, but rather as capturing the stochasticity in the conditional process given the context. To do this, define $f_{r,z}$ with a deterministic $r = r(x_\mathcal{C}, y_\mathcal{C})$ representing dependence on context, and a stochastic z describing the stochasticity in the random function conditional on the context. We can train NPs using objectives in (1)–(3) with the decoder being $f_{r(x_\mathcal{C}, y_\mathcal{C}), z}$ instead of f_z. We denote these objectives by det+$[T|\varnothing]$, det+$[C|\varnothing]$, and det+$[T|C]$ respectively.

Attention. Kim et al. [6] propose using attention [8, 7] in the deterministic encoder as a way to address the issue of underfitting that is observed in the original model specification of NPs. Here, the deterministic representation $r(x_\mathcal{C}, y_\mathcal{C})$ additionally takes in the target input x_i so that the representation attends to relevant context points. This results in a decoder of the form $f_{r_{x_i}(x_\mathcal{C}, y_\mathcal{C}), z}$ or $f_{r_{x_i}(x_\mathcal{C}, y_\mathcal{C})}$ depending on whether it is used in objectives (1)–(3) for NPs or in objective (4) for a conditional NP. We denote these objectives by att+$[T|\varnothing]$, att+$[C|\varnothing]$, att+$[T|C]$, and det+att.

Observation variance. The likelihood term $p(y_i|f_z(x_i))$ is typically a Normal distribution whose parameters are given by the output of the decoder neural network $f_z(x_i)$. While Garnelo et al. [4] fix the variance of the Normal distribution, Kim et al. [6] learn it. We investigate the effect of this design choice.

Training distribution of (m, n). NPs are typically trained by choosing a distribution of (m, n) conditioned on which we form the context and target sets $x_\mathcal{C}, y_\mathcal{C}, x_\mathcal{T}, y_\mathcal{T}$. Training with different relative sizes of m and n can presumably affect the predictive performance. Thus, we consider the following distributions of (m, n). Small context set and large target set "$\downarrow m, \uparrow n$", where $m \sim [3, 47), n \sim [50, 100)$; large context set and large target set "$\uparrow m, \uparrow n$", where $m \sim [50, 97), n \sim [m + 1, 100)$; and the originally proposed "$\updownarrow m, \updownarrow n$", where $m \sim [3, 97), n \sim [m + 1, 100)$.

3 Experiments

We run experiments on 1D data of synthetic Gaussian processs (GPs) and 2D data of MNIST images where x is the pixel position and y is the pixel color. The data and neural network architectures are described in the Appendices A and B respectively.

Quantitative evaluation. We evaluate the performance through the (normalized) predictive LL $\frac{1}{n-m} \log p(y_{\mathcal{T}\setminus\mathcal{C}}|x_{\mathcal{T}\setminus\mathcal{C}}, x_\mathcal{C}, y_\mathcal{C})$ and the (normalized) joint LL $\frac{1}{n} \log p(y_\mathcal{T}|x_\mathcal{T})$. For models without a latent variable, we only evaluate the predictive LL which can be done directly. In models with a latent variable, we must resort to evaluating lower bounds:

$$\mathbb{E}_{q(z_k|x_\mathcal{C},y_\mathcal{C})}\left[\log\left(\frac{1}{K}\sum_{k=1}^{K}\frac{p(y_{\mathcal{T}\setminus\mathcal{C}}|x_{\mathcal{T}\setminus\mathcal{C}}, z_k)p(z_k|x_\mathcal{C}, y_\mathcal{C})}{q(z_k|x_\mathcal{C}, y_\mathcal{C})}\right)\right] \leq \log p(y_{\mathcal{T}\setminus\mathcal{C}}|x_{\mathcal{T}\setminus\mathcal{C}}, x_\mathcal{C}, y_\mathcal{C}), \quad (5)$$

$$\mathbb{E}_{q(z_k|x_\mathcal{T},y_\mathcal{T})}\left[\log\left(\frac{1}{K}\sum_{k=1}^{K}\frac{p(y_\mathcal{T}|x_\mathcal{T}, z_k)p(z_k)}{q(z_k|x_\mathcal{T}, y_\mathcal{T})}\right)\right] \leq \log p(y_\mathcal{T}|x_\mathcal{T}), \quad (6)$$

where we sample K independent importance samples $z_k \sim q(z|x_\mathcal{T}, y_\mathcal{T})$ similarly to Burda et al. [1]. Note that to estimate the lower bound (5), we approximate $p(z|x_\mathcal{C}, y_\mathcal{C}) \approx q(z|x_\mathcal{C}, y_\mathcal{C})$, which cancels with the term in the denominator. The joint LL metric favors models trained with objectives $[T|\varnothing]$ and $[C|\varnothing]$ as they are also lower bounds to the joint LL. The predictive LL metric is different to the objectives used for training, however it is similar to the objective $[T|C]$ which is a lower bound to the conditional LL, $\log p(y_\mathcal{T}|x_\mathcal{T}, x_\mathcal{C}, y_\mathcal{C})$. We evaluate these metrics on new context and target sets sampled from the "$\updownarrow m, \updownarrow n$" distribution of (m, n). We show quantitative evaluation of different objectives and model specifications in Figure 1.

Qualitative evaluation. Given $x_\mathcal{C}, y_\mathcal{C}, x_\mathcal{T}$, we plot (i) a **multi-sample** estimate of the mean of the posterior predictive $p(y_\mathcal{T}|x_\mathcal{C}, y_\mathcal{C}, x_\mathcal{T})$, obtained by sampling $z \sim q(z|x_\mathcal{C}, y_\mathcal{C})$ multiple times and

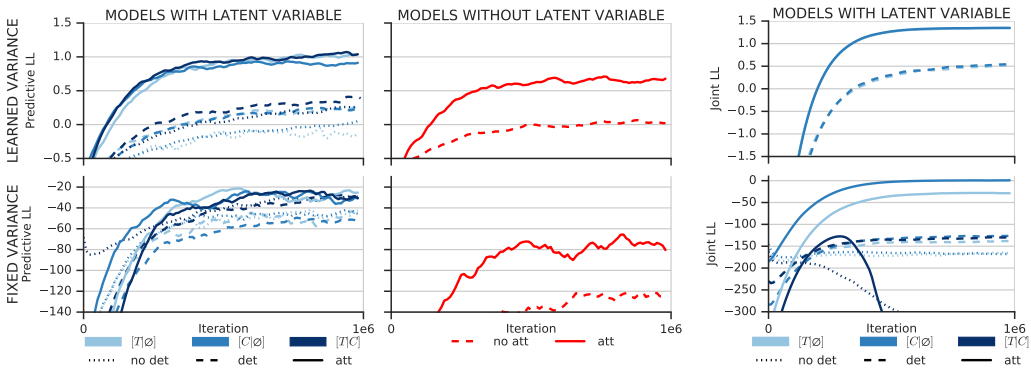
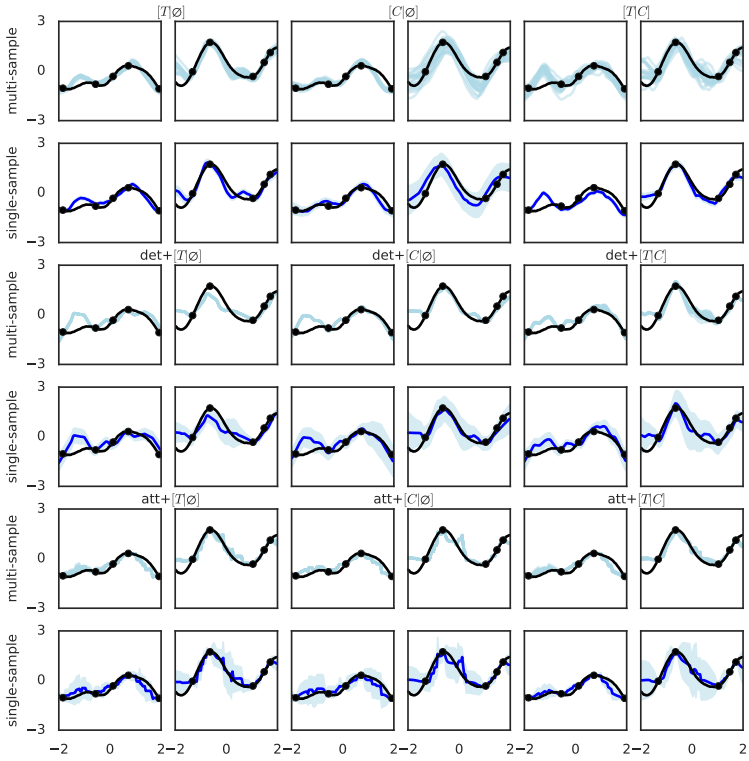
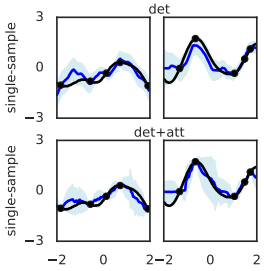


Figure 1: Predictive log likelihood (LL) (5) and joint LL (6) ($K = 1000$) vs training iteration for NP models trained with different objectives and model specifications on 1D synthetic GPs.

plotting the mean obtained by the decoder neural network f for each z, and (ii) a **single-sample** estimate of posterior predictive, obtained by sampling $z \sim q(z|x_\mathcal{C}, y_\mathcal{C})$ once and plotting the mean and standard deviation given by the output of the decoder neural network. The former shows whether the latent variable is useful in capturing variability of the stochastic process; the latter shows the contribution of the observation noise (for models without a latent variable, we can only show the latter). The posterior predictive samples for models trained with learned and fixed variance are shown in Figures 2 and 3 respectively.



(a) Models with a latent variable.

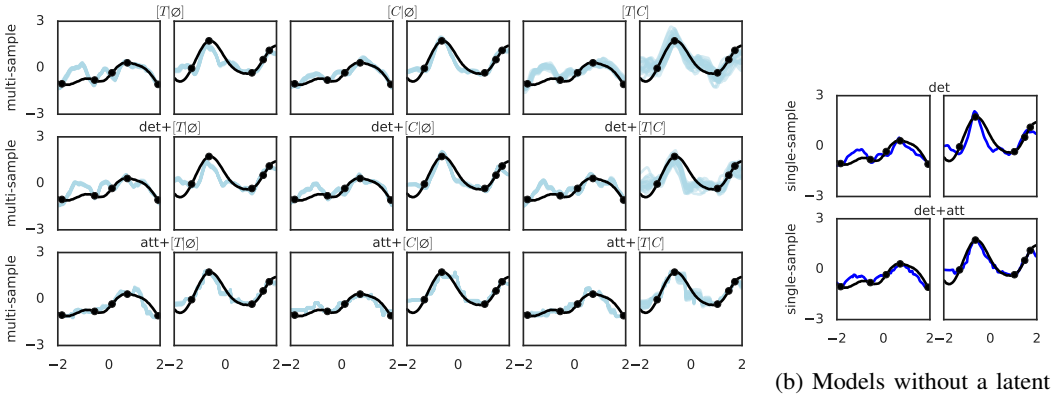


(b) Models without a latent variable.

Figure 2: Posterior predictive distribution for models with a learned observation variance, trained on 1D synthetic GPs. Black lines and points show the target points and context points respectively. Multi-sample plots are shown with light blue lines. Single-sample plots show the mean with a blue line and one standard deviation using light blue shading.

Effect of objective on latent variable models. Quantitatively, there is no objective that clearly outperforms the others. When using the predictive LL as a metric, objective $[T|C]$ outperforms the other objectives in most cases (Figure 1, left). This can potentially be because this metric is more similar to the conditional LL which is targeted by the $[T|C]$ objective. However, the difference in performance is often small, especially for models that are best-performing overall (ones that use attention). On the other hand, the joint LL of models trained with objective $[T|C]$ is divergent (Figure 1, right). This can potentially be due the objective $[T|C]$ targeting a lower bound to the conditional LL (3) instead, and so using the learned encoder q can lead to a bad estimator of the joint LL in (6).

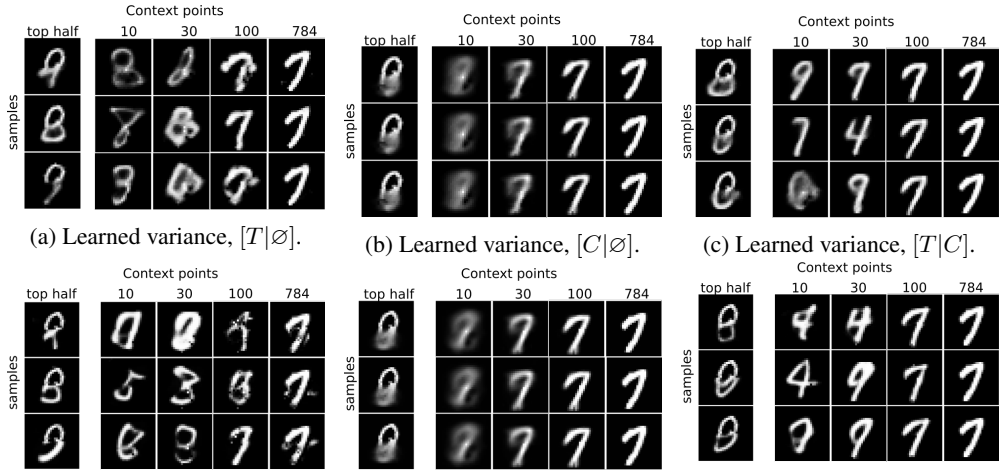
Qualitatively, we observe that training with objective $[T|C]$ typically results in higher variability in the posterior predictive (Figure 2a and 3a). Posterior predictive distributions with higher variability can be useful in applications of NPs where exploration is needed, like Thompson sampling or Bayesian optimization [3, 4]. Higher variability in the posterior predictive distribution is also observed on 2D MNIST data (Figure 4). Samples from the model trained with objective $[T|C]$ are diverse and look realistic. Samples from the model trained with objective $[C|\varnothing]$ are almost identical and blurry. Samples from the model trained with objective $[T|\varnothing]$ are diverse and sharp, however not realistic. Qualitative results agree more with the predictive LL than with the joint LL.



(a) Models with a latent variable.

(b) Models without a latent variable.

Figure 3: Posterior predictive distribution $p(y_\mathcal{T}|x_\mathcal{T}, x_\mathcal{C}, y_\mathcal{C})$ for models with a fixed observation variance, trained on 1D synthetic GPs. Line styles same as Figure 2.



(a) Learned variance, $[T|\varnothing]$.

(b) Learned variance, $[C|\varnothing]$.

(c) Learned variance, $[T|C]$.

(d) Fixed variance, $[T|\varnothing]$.

(e) Fixed variance, $[C|\varnothing]$.

(f) Fixed variance, $[T|C]$.

Figure 4: Posterior predictive samples for MNIST data. Each subfigure corresponds to training a latent variable model with a different objective and with or without learning the observation variance. Attention is used. In a subfigure, each row corresponds to the posterior predictive mean obtained from three different samples from $q(z|x_\mathcal{C}, y_\mathcal{C})$. The context set is either the top half of the image or 10, 30, 100, or 784 random points in the image. The target set is the full image.

Effect of the deterministic path. Quantitatively, having a deterministic path is typically better than not having it (Figure 1), with the exception of 1D models trained with fixed variance as measured by the predictive LL. However, in this case, the performance gap is small.

Qualitatively, in the case of latent variable models with learned observation variance, using a deterministic path leads to the model using the latent variable less for modeling the variability of the stochastic process, which is instead being modeled using the observation variance (Figure 2a).

Effect of having a latent variable. Quantitatively, not using a latent variable decreases performance (Figure 1). Having a latent variable increases the size of the model family which potentially allows us to fit the data more accurately. Qualitatively, this can be seen most clearly in the case of models with a fixed observation variance, where not using a latent variable prevents us to learning any variability of the stochastic process (Figure 3b).

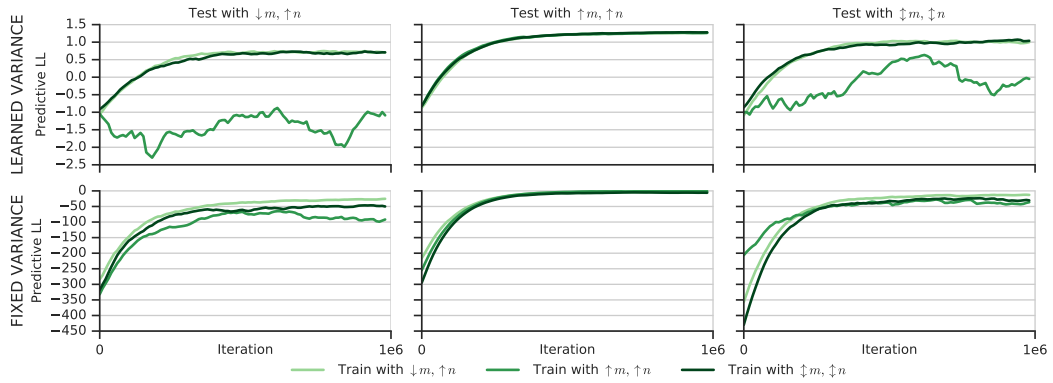


Figure 5: Predictive LL (5) for different combinations of training and testing distribution of (m, n) for a latent variable model trained with objective $[T|C]$ and learned observation variance on 1D synthetic GPs.

Effect of attention. Quantitatively, having attention improves performance regardless of the choice of the other settings (Figure 1). Qualitatively, attention helps by having low predictive uncertainty near context points. This confirms findings of Kim et al. [6].

Effect of observation variance. Quantitatively, fixing the observation variance decreases performance (Figure 1). This is because fixing the observation variance decreases the size of the model family, making it harder to learn a good model.

Fixing the observation variance means that any variability of the stochastic process needs to be learned by the latent variable which can be seen by the variability of the multi-sample plots in Figure 3a. Training models without a latent variable in this case is not suitable since we cannot capture the variability of the stochastic process (Figure 3b).

When the observation variance is learned, the variability of the stochastic process can be learned by the latent variable (shown by multi-sample plots) or by the observation variance (shown by the single-sample plots). This division is roughly equal for models trained without attention and a deterministic path (Figure 2a, top two rows). However, for models trained with a deterministic path or attention, the variability of the stochastic process is learned mainly by the observation variance (Figure 2a, last four rows). For models without a latent variable, the variability of the stochastic process can only be captured by the observation variance (Figure 2b). Hence, for models with no latent variable and fixed observation variance, the fit is poor (Figure 2b).

Effect of training data distribution. Quantitatively, training with $\downarrow m, \uparrow n$ and $\updownarrow m, \updownarrow n$ is the best; we get noticeably worse performance when training with $\uparrow m, \uparrow n$ (Figure 5). In Figure 5, we show predictive LL for a latent variable model trained with objective $[T|C]$ and learned observation variance, however the conclusions also hold for different objectives and model specifications. Qualitative evaluation confirms this. Posterior predictive samples from a model trained with $\uparrow m, \uparrow n$ look the worst as they place low uncertainty on out of context points (Figure 6).

4 Conclusions

As measured by the predictive LL (eq. (5)), the objective $[T|C]$ used to train the original NPs [4] outperform the supervised VHE objective $[C|\varnothing]$ (eq. (2)) [5] and supervised NS objective $[T|\varnothing]$ (eq. (1)) [2]. Measuring the performance using the joint LL (eq. (6)) doesn't yield the same conclusion. However, qualitative results agree more with the conclusions given by the predictive LL as training with $[T|C]$ results in a more diverse and realistic posterior predictive distribution.

Regarding model specifications, we have observed that using a deterministic path, attention and a latent variable helps both quantitatively and qualitatively. Learning an observation variance leads to a better quantitative performance, however the latent variable is used less, as the variability of the stochastic process is mainly learned by the observation variance. Regarding training data distribution

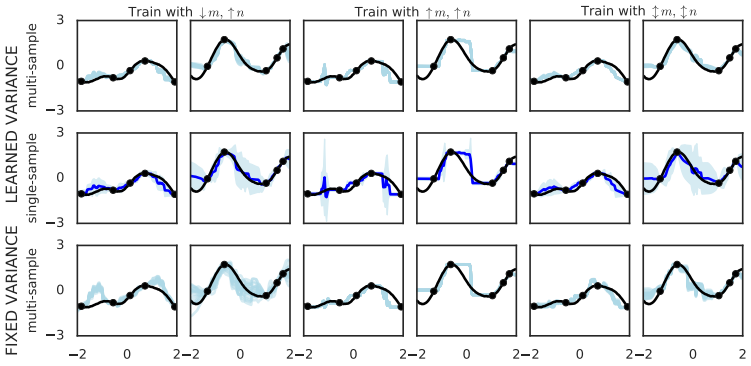


Figure 6: Posterior predictive samples for different combinations of training and testing distribution of (m, n). Line styles same as in Figure 2.

of the (m, n) (the number of context and target points), the originally proposed $m \sim [3, 97), n \sim [m + 1, 100)$ performs the best, closely followed by $m \sim [3, 47), n \sim [50, 100)$.

Given that some objectives and model specifications clearly outperform others, our analysis can be useful in guiding future research and applications of NPs.

References

[1] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *ICLR*, 2016.

[2] Harrison Edwards and Amos Storkey. Towards a neural statistician. In *ICLR*, 2017.

[3] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *ICML*, 2018.

[4] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. In *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

[5] Luke B Hewitt, Maxwell I Nye, Andreea Gane, Tommi Jaakkola, and Joshua B Tenenbaum. The variational homoencoder: Learning to learn high capacity generative models from few examples. In *UAI*, 2018.

[6] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *NIPS Bayesian Deep Learning workshop*, 2018.

[7] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, and Alexander Ku. Image transformer. In *ICML*, 2018.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

Appendix A Data

1D dataset. This dataset consists of sets of 1D input-output pairs generated from a GP with the squared exponential kernel $k(x, x') = \sigma^2 \exp(-(x - x')^2/2\ell^2)$. The context and target set sizes are sampled from the "$\updownarrow m, \updownarrow n$" distribution $m \sim [3, 97)$ and $n \sim [m + 1, 100)$ unless otherwise specified. The target inputs $x_\mathcal{T}$ are sampled $x_i \sim \text{Uniform}(-2, 2)$. The corresponding target outputs $y_\mathcal{T}$ are obtained by sampling from a GP with $\sigma \sim \text{Uniform}(0.1, 1)$ and lengthscale $\ell \sim \text{Uniform}(0.1, 0.6)$ with the observation noise standard deviation 0.02. The context set is a subset of the target set.

2D dataset. As described in [6].

Appendix B Neural Network Architecture

The neural network architecture for 2D dataset is the same as in [6]. Following is the description of the neural network architecture for the 1D dataset.

Training is perform using Adam with default hyperparameters and learning rate 5×10^5, batch size 16. Latent variable $z \in \mathbb{R}^{d_z}$ has $d_z = 64$ dimensions. Deterministic representation $r(x_\mathcal{C}, y_\mathcal{C}) \in \mathbb{R}^{d_r}$ has $d_r = 64$ dimensions. Model architectures are summarized in Figure 7.

Decoder. $f_z(x_i) : (1 + d_z) \xrightarrow{\text{lin+relu}} 64 \xrightarrow{\text{lin+relu}} 64 \xrightarrow{\text{lin+relu}} 2 \xrightarrow{\text{split}} \texttt{loc, scale}'$, where $\texttt{scale} = 0.1 + 0.9 \cdot \text{softplus}(\texttt{scale}')$ (or fixed to 0.02). The observation distribution is $p(y_i|f_z(x_i)) = \text{Normal}(y_i|\texttt{loc}, \texttt{scale}^2)$. When using deterministic path, input dimension of $f_{r(x_\mathcal{C}, y_\mathcal{C}), z}(x_i)$ is $(1 + d_z + d_r)$.

Prior. $p(z) = \text{Normal}(z|0, I)$.

Latent encoder.

$$q(z|x_\mathcal{C}, y_\mathcal{C}) : 2 \times |\mathcal{C}| \underbrace{\xrightarrow{\text{lin+relu}} 128 \times |\mathcal{C}|}_{\text{3 times}} \xrightarrow{\text{mean}} 128 \xrightarrow{\text{lin+relu}} 96 \begin{cases} \xrightarrow{\text{lin+relu}} 64\,(\texttt{loc}) \\ \xrightarrow{\text{lin+relu}} 64\,(\texttt{scale}') \end{cases},$$

where $\texttt{scale} = 0.1 + 0.9 \cdot \text{sigmoid}(\texttt{scale}')$. The variational posterior distribution is $q(z|x_\mathcal{C}, y_\mathcal{C}) = \text{Normal}(\texttt{loc}, \text{diag}(\texttt{scale}^2))$

Deterministic encoder. $r(x_\mathcal{C}, y_\mathcal{C}) : 2 \times |\mathcal{C}| \underbrace{\xrightarrow{\text{lin+relu}} 128 \times |\mathcal{C}|}_{\text{6 times}} \underbrace{\xrightarrow{\text{lin+relu}} 64 \times |\mathcal{C}|}_{\text{twice}} \xrightarrow{\text{mean}} 64$. For attention, last step takes in x_i and performs multihead attention, described in [6, Section 2] and summarized in Figure 7.

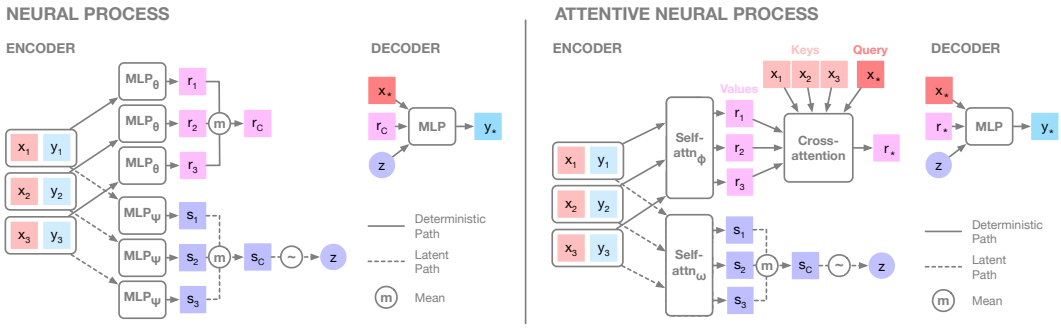


Figure 7: Model architecture for NPs and attentive NPs. Our architecture doesn't use self-attention. Used with permission of Kim et al. [6].

Appendix C Quantitative Evaluation for MNIST data

We include the quantitative evaluation for models trained on MNIST data. Similar conclusions are drawn from quantitative evaluation of models trained on 1D synthetic GP data (Figure 1).

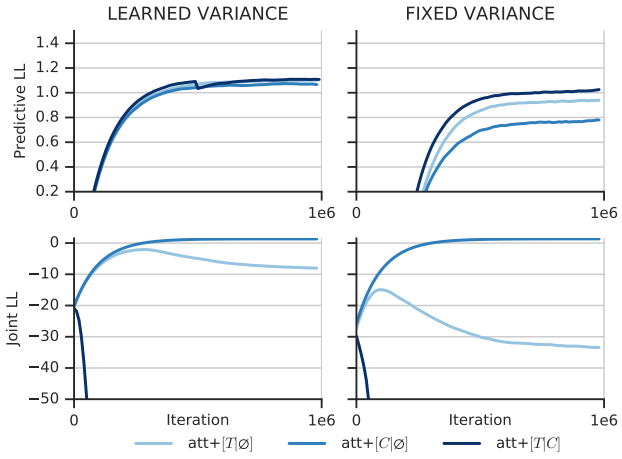


Figure 8: Predictive LL (5) and joint LL (6) vs training iteration for NP models trained with different objectives and model specifications on MNIST data.