A Comparison of Conditional Neural Processes and Latent Neural Processes

Advanced Machine Learning, University of Cambridge

William Baker, Alexandra Shaw, Tony Wu

March 27, 2023

Contents

1 Abstract

In this paper, we replicate the results from Garnelo et al. on Conditional Neural Processes (CNPs) and from Garnelo et al. on Latent Neural Processes (LNPs), two model types part of the Neural Processes (NP) family [1] [2]. While neural networks excel at function approximation, Gaussian Processes (GPs) address different challenges such as uncertainty prediction, continuous learning, and the ability to deal with data scarcity. Therefore, each model is only suited for a restricted spectrum of tasks that strongly depends on the nature of available data.

Neural Processes use neural networks to encode distributions over functions to approximate the distributions over functions given by stochastic processes like GPs. This allows for efficient inference and scalability to large datasets. The performance of these models will be evaluated on 1D-regression and image completion to demonstrate visually how they learn distributions over complex functions.

The source code for the project is publicly available at our Github repository.

2 Introduction

Gaussian Processes (GPs) [12] provide a tractable way to model uncertainty by learning a distribution over functions given a limited set of input-output pairs as context. Gaussian Processes are highly desirable for inference, as they provide guarantees on uncertainty predictions under the Bayesian framework and are able to provide sensible predictions even with few examples from the dataset. Further, Gaussian Processes are invariant to the ordering of the data, and are able to be used as a basis for continual learning.

GP's main drawback is their poor inference performance, predicting new outputs with $\mathcal{O}((n+m)^3)$ complexity, where n is the number of context points and m the number of target points. When applied to complex functions or large datasets, inference becomes intractable.

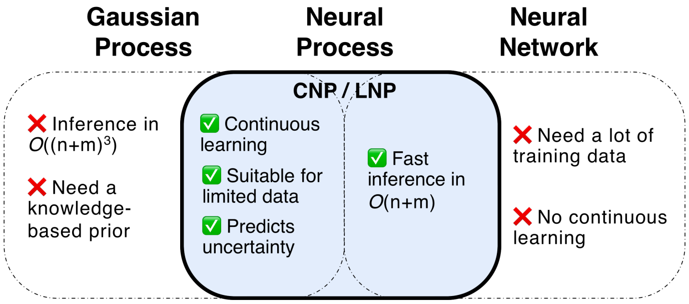


Figure 1: Comparison between Gaussian Process, Neural Network and Neural Process

Conditional Neural Processes (CNPs) and Latent Neural Processes (LNPs) attempt to combine the best characteristics of GPs and neural networks to reduce inference time but maintain uncertainty estimates. Both CNPs and LNPs are part of the Neural Process (NP) family, which model the mean and variance function of a dataset using neural networks. Consequently, inference requires a single forward pass through the network, which takes $\mathcal{O}(n+m)$ time. NPs also do not require a knowledge-based prior, as they use an approximate, data-driven prior implicitly learned by the model during training.

NPs lose Bayesian guarantees on the learned mean and variance in the distribution over functions but add flexibility in function approximation. The NP Family has many broad applications [4], and our team decided to select this project because we were drawn by the idea of combining uncertainty predictions and neural networks to do efficient predictions.

The goal of this report is to replicate the main experiments from CNPs as presented by Garnelo et al. [1]. We also replicate the results of LNPs as presented by Garnelo at al. [2]. Using these results, we extend to compare the properties of CNPs and LNPs. Finally, we extend the results to experiment with Hybrid Neural Processes (HNP), which are a combination of LNPs and CNPs.

This report is structured as follows. In section 3, we describe the CNP and LNP models introduced by Garnelo et al. [1] [2]. In section 4, we describe the overarching training process for all experiments. In section 5, we describe the results of running the main CNP and LNP experiments, showing that we replicate the results of Garnelo et al. [1] [2] in both papers. In section 6, we describe the results of our HNP experiments which extend the results of CNP and LNPs. Finally, in section 7 we give a critique of our work and discuss future work we could do on this topic in section 9.

3 Models

3.1 Stochastic Process

Neural processes model stochastic processes. Stochastic processes define a generative process in which functions $f : X \to Y$ from inputs to outputs are drawn from an underlying probability distribution P. Consider a set of labelled context points $C = \{(x_i, y_i)\}_{i=0}^{n-1} \subset X \times Y$ and a set of unlabelled target points $T = \{(x_i)\}_{i=n}^{n+m-1} \subset X$. The goal in a stochastic process is to accurately predict the output values $f(x)$ for the targets $x \in T$ given the labelled context points C. The probability distribution P over functions defines the conditional distribution over target outputs given the context points and target inputs $P(f(T)|C, T)$.

GPs [12] are a specific type of stochastic process in which we assume that any finite set of function evaluations of f are jointly Gaussian distributed. Although being extremely data-efficient, learning the joint Gaussian distributions of a GP requires computing the inverse of a large covariance matrix. Therefore, inference with GPs scales with $\mathcal{O}((n+m)^3)$. This provides the motivation to explore more efficient methods of predicting uncertainty.

3.1.1 Conditions for Modelling Stochastic Processes

In order to approximate a stochastic process, NPs must satisfy the main properties of stochastic processes: exchangeability and consistency. In order to define these, we must define the joint distribution of a stochastic process $\rho_{x_{1:n}}(y_{1:n})$, and the permutation function π:

$$\rho_{x_{1:n}}(y_{1:n}) = \int p(f)p(y_{1:n}|f, x_{1:n})\,\mathrm{d}f \qquad \pi(x_{1:n} := (x_{\pi(1)}, ..., x_{\pi(n)}) \tag{1}$$

1. **Exchangeability** We consider the stochastic process P that generates functions f of the form $f : X \to Y$. The marginal distribution $\rho_{x_{1:n}}$ for a given sequence of the elements $x_{1:n}$ is $(F(x_1), ..., F(x_n))$. To satisfy exchangeability, any permutation of $x_{1:n}$, $\pi(x_{1:n})$ should result in the same marginal distribution.

$$\rho_{x_{1:n}}(y_{1:n}) = \rho_{\pi(x_{1:n})}(\pi(y_{1:n})) \tag{2}$$

2. **Consistency** Marginalising out part of the sequence that defines the distribution results in the same distribution as the one defined on the original sequence. For any $1 \leq m \leq n$, the following condition should be satisfied.

$$\rho_{x_{1:n}}(y_{1:m}) = \int \rho_{x_{1:n}}(y_{1:n})\,\mathrm{d}y_{m+1:n} \tag{3}$$

3.2 Conditional Neural Processes

3.2.1 Architecture

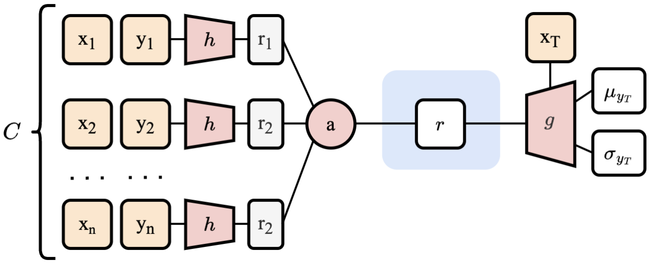


Figure 2: Architecture diagram of CNPs.

Conditional Neural Processes (CNPs) were proposed by Garnelo et al. [1]. As CNPs model stochastic processes, we reuse the notations defined in subsection 3.1. $C = \{(x_i, y_i)\}_{i=0}^{n-1} \subset X \times Y$ is a set of labelled context points and $T = \{(x_i)\}_{i=n}^{n+m-1} \subset X$ is a set of unlabelled target points.

Just like their counterparts, CNPs define distributions over functions f for inputs x in the set of targets. However, instead of using a kernel to condition on context points C, they use a neural network to condition predictions on an intermediate embedding space $\mathbb{R}^d$. They achieve this by using the following architecture:

$$\begin{array}{ll} r_i = h_\theta\left(x_i, y_i\right) & \forall\left(x_i, y_i\right) \in C \\ r = r_1 \oplus r_2 \oplus \ldots r_{n-1} \oplus r_n & \\ \phi_i = g_\theta\left(x_i, r\right) & \forall\left(x_i\right) \in T \end{array} \tag{4}$$

where the encoder function $h_\theta : X \times Y \to \mathbb{R}^d$ and decoder function $g_\theta : X \times \mathbb{R}^d \to \mathbb{R}^e$ are multi-layer neural networks, $\oplus$ is an aggregator that takes elements in $\mathbb{R}^d$ and maps them into a single element of $\mathbb{R}^d$, and ϕ_i parameterise the decoder output. A graphical representation of this architecture is shown in Figure 2.

Exchangeability is guaranteed by ensuring the aggregator $\oplus$ is permutation invariant to the context set C. The aggregator used here is the average over representations r_i. For the regression tasks, we extract a mean and variance prediction $\left(\mu_i, \sigma_i^2\right)$ from each ϕ_i. Then the model outputs y_i are drawn from $Q_\theta(f(x_i) \mid C, T) = Q_\theta(f(x_i) \mid \phi_i)$ *i.e.* from $\mathcal{N}\left(\mu_i, \sigma_i^2\right)$. The Gaussian distribution outputs give CNPs a factored structure over the target elements, ensuring consistency.

$$Q_\theta(f(T) \mid C, T) = \prod_{x \in T} Q_\theta(f(x) \mid C, x) \tag{5}$$

Now we have defined the CNP model and both conditions to model a stochastic process as defined in Section 3.1.1 have been met.

3.2.2 Loss Function

The loss used for CNP is the negative conditional log probability defined by

$$\mathcal{L}_{CNP}(\theta) = -\mathbb{E}_{f \sim P}\left[\mathbb{E}_N\left[\log Q_\theta\left(\{y_i\}_{i=N}^{N+M-1} \mid C_N, \{x_i\}_{i=N}^{N+M-1}\right)\right]\right] \tag{6}$$

where N denotes the number of context points in the context set C_N and $T = \{x_i\}_{i=N}^{N+M-1}$. This loss function maximises the log-likelihood of the target points with respect to the Gaussian distribution defined by the predicted mean and variance.

3.2.3 Properties

CNPs learn an approximate, data-driven prior rather than a true prior, meaning that the prior does not have to be pre-defined, but they do not have the mathematical guarantees of uncertainty of a GP.

CNPs are unable to produce coherent function samples, as they learn the point-wise mean and variance of the input points and no covariance is learned between target points. Thus we see in section 5.1.2 that CNPs are not capable of dealing with piece-wise distributions.

3.3 Latent Neural Processes

3.3.1 Architecture

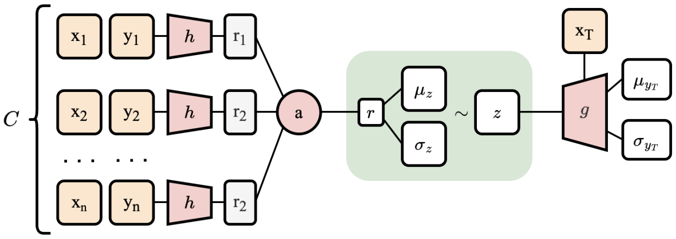


Figure 3: Architecture diagram of LNPs.

Neural Processes from Garnelo et. al [2], which we refer to as Latent Neural Processes (LNP), extend CNPs by adding an intermediate latent space to capture the correlation between input points. This can be interpreted as adding extra degrees of freedom which should encompass a set of different priors. Using these extra priors allows us to draw one function at a time rather than one point output at a time, hence making the set of outputs coherent. The mathematical formulation of the architecture is as follows.

$$\begin{aligned}
&r_i = h_\theta\left(x_i, y_i\right) && \forall\left(x_i, y_i\right) \in C \\
&r = r_1 \oplus r_2 \oplus \ldots r_{n-1} \oplus r_n && \\
&z \sim \mathcal{N}(\mu_z(r), \operatorname{diag}[\sigma_z(r)]) && \\
&\phi_i = g_\theta\left(x_i, r\right) && \forall\left(x_i\right) \in T
\end{aligned} \tag{7}$$

where z is drawn from a multi-variate Gaussian distribution learned from the representations r. This distribution is parameterised by the learned mean function μ_z and learned variance function σ_z^2, represented in the MLP architecture (not statistics of r directly). The rest of the model remains the same as in CNPs (Equation 4).

The generative model for LNPs can be represented as follows:

$$p(z, f(T)|T, C) = p(z|C) \prod_{x \in T} \mathcal{N}(f(x)|\mu(g_\theta(x, z)), \operatorname{diag}[\sigma^2(g_\theta(x, z))]) \tag{8}$$

$p(z|C)$ is the multivariate Gaussian distribution over the latent space $\mathcal{N}(z|\mu_z, \sigma_z^2)$ capturing global uncertainty over the function samples. The second part of the equation is the likelihood of generating the function outputs on the target points given the mean μ and variance σ^2 output by the decoder. The form of the generative function ensures consistency. Exchangeability is ensured, as in CNPs, by the use of the mean aggregator function, the factorised Gaussian latent space, and the factorised Gaussian outputs. Therefore, LNPs meet both conditions to approximate stochastic processes.

3.3.2 Loss Function

The use of the latent space means that LNPs require a new loss function which minimises the distance between the prior $p(z|C)$ and the posterior distribution $p(z|C,T)$. As the neural network is shown new functions in the distribution, it can learn the relationship between context and target points.

The objective for LNPs is to maximise the evidence lower bound (ELBO) of the log probability of the predictive distribution $p(f(T)|T,C)$. The true prior over the latent variable given the context points $p(z|C)$ is intractable, so it is approximated by a multi-variate Gaussian over the latent space $q(z|C) = \mathcal{N}(z|\mu(C), \mathrm{diag}[\sigma^2(C)])$. This also ensures exchangeability, as this can be represented as a factored Gaussian.

$$\begin{aligned} p(f(T)|T,C) &\geq \mathbb{E}_{q(z|C,T)}\left[\sum_{x\in T} \log p(f(x)|z,x) - \log\frac{q(z|C,T)}{p(z|C)}\right] \\ &\approx \mathbb{E}_{q(z|C,T)}\left[\sum_{x\in T} \log p(f(x)|z,x) - \log\frac{q(z|C,T)}{q(z|C)}\right] \\ &= \mathbb{E}_{q(z|C,T)}\left[\sum_{x\in T} \log p(f(x)|z,x) - D_{KL}(q(z|C,T)\|q(z|C))\right] \end{aligned} \tag{9}$$

The first term of Equation 9 is the conditional log probability of the model predicted distribution given the latent distribution and the target inputs. This performs a similar function to the CNP loss, ensuring that the mean and variance output predictions fit the true target outputs. The second term of the LNP loss function is the KL-divergence between the posterior distribution in latent space given the target and context sets $q(z|C,T)$ and the prior distribution in latent space given only the context set $q(z|C)$. The method of maximising the evidence lower-bound is introduced in Variational Auto-Encoders (VAE) [6] and is applied here as well, as both NPs and VAEs involve meta-learning a latent distribution.

3.3.3 Properties

LNPs have many of the same properties of CNPs, but they are able to produce coherent global predictions of mean and variance. This allows LNPs to generate coherent function samples even with few context points, while CNPs cannot. Consequently, LNPs can guess samples from the function class so much so that the more the number of context points the more closely the samples match the correct function.

4 Training

4.1 Data pipeline

Assume a stochastic process P that defines a distribution of functions. Our general training procedure is the following:

1. Draw N functions $f_1, \dots, f_N$ from P
2. For $i = 1, \dots, N$:
   (a) Draw $n_c^{(i)} \sim U(\min, \max)$ the number of context points

   (b) Draw $n_t^{(i)} \sim U(\min, \max)$ the number of target points

(c) Draw $C_i = \{(x_1, y_1), \ldots, (x_{n_c^{(i)}}, y_{n_c^{(i)}})\}$ and $T_i = \{(x_1, y_1), \ldots, (x_{n_t^{(i)}}, y_{n_t^{(i)}})\}$ points from f_i

(d) Compute the loss and use the optimiser to backpropagate and update the weights

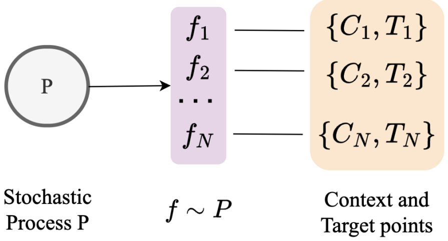


Figure 4: Data pipeline for Neural Processes

Note that in practice, we will use mini-batch training for faster convergence. Predicting the outputs of a limited number of target points T_i reduces the information gained from each iteration in the loss function. Instead, all the sampled points, both C_i and T_i, will be used as targets in the loss function (Figure 5 for examples of generated data). This makes the model learn faster for both LNPs and CNPs.

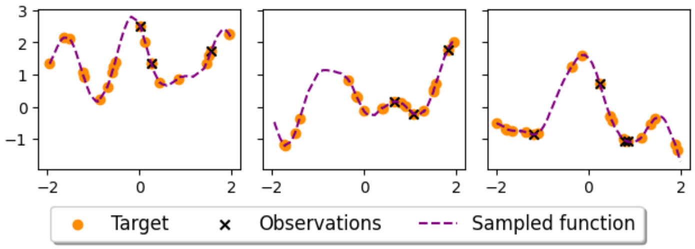


Figure 5: Example generated by the dataloader for 1D-regression

This training process will be used for all experiments. Note that we might not have access to the underlying stochastic process, but we treat each example in the dataset as a function sample from the true distribution over functions, allowing us to use the same overall process shown in Figure 4. In particular, context points in image datasets are selected as described above and the entire image is inputted as the target set.

4.2 Optimisation

The expectations in both CNPs and LNPs loss functions are approximated using a Monte-Carlo approach where the mean is computed over each batch of context-target sets (C_i, T_i). When optimising the objective in LNPs, the gradients are backpropagated using the reparametrization trick, which ensures that gradients are not passed through the non-deterministic links between the encoder and decoder [6].

As both models are neural networks, they are optimised using the Adam optimiser [7]. Table 1 shows the MLP architectures used for all experiments. LNPs in general were found to take longer to train, due to the more complex latent space representation. Note that the last hidden layer of the decoder has 6 units instead of 3 to take into account the 3 channels of the image.

	Encoder MLP	**Latent MLP**	**Decoder MLP**
CNP	[500, 500, 500, 500]	N/A	[500, 500, 500, 2]
LNP	[500, 500, 500, 500]	[500, 500, 500, 1000]	[500, 500, 500, 2]

Table 1: NP architectures used for experiments.

5 Experiments

The ability of NPs to describe complex functions is easily demonstrated for visual datasets. Garnelo et al. demonstrate results for 1D regression tasks and 2D image completion on both the MNIST handwritten digit database [10] and the CelebFaces Attributes Dataset (CelebA) dataset [11]. We replicate all three of these experiments, as they clearly show the performance of the CNP model.

There are a few experiments which we decided to be unhelpful to compare CNP and LNP performance. One such experiment is Omniglot image classification [8], which demonstrates the performance of CNPs on classification tasks and few-shot learning. We decided not to replicate this experiment because classification can be achieved by changing the output layer to a vector of logits rather than a concatenation of mean and variance. Moreover, our extensions also explore regression tasks, so we focus our efforts on the following experiments. Additionally, we do not replicate the latent CNP experiments, as the latent CNP model is underspecified and hard to distinguish from the LNP without more information.

5.1 1D Regression

5.1.1 Target distribution with fixed kernel

To evaluate the performance of CNPs and LNPs on 1D-regression tasks, we draw functions from a GP with a squared exponential kernel function with fixed parameters.

We use the entire input space as the target set for easy visualisation and compare the results to the true GP. To properly compare our results with the original CNP paper [1], we used the parameters used in the original source code. The results for our LNP are also pictured [2].

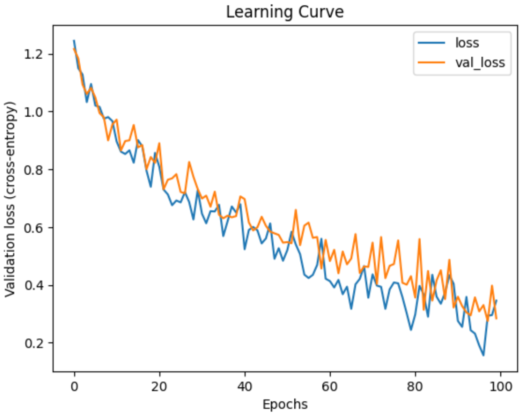


Figure 6: Learning curve for CNP on the 1D-regression task (fixed kernel parameter)

We trained our CNP model for 100 epochs. We show the training results in Figure 6 to verify that the model is indeed learning. As both the training and validation losses are still decreasing at the same pace, the model hasn't *a priori* overfitted.

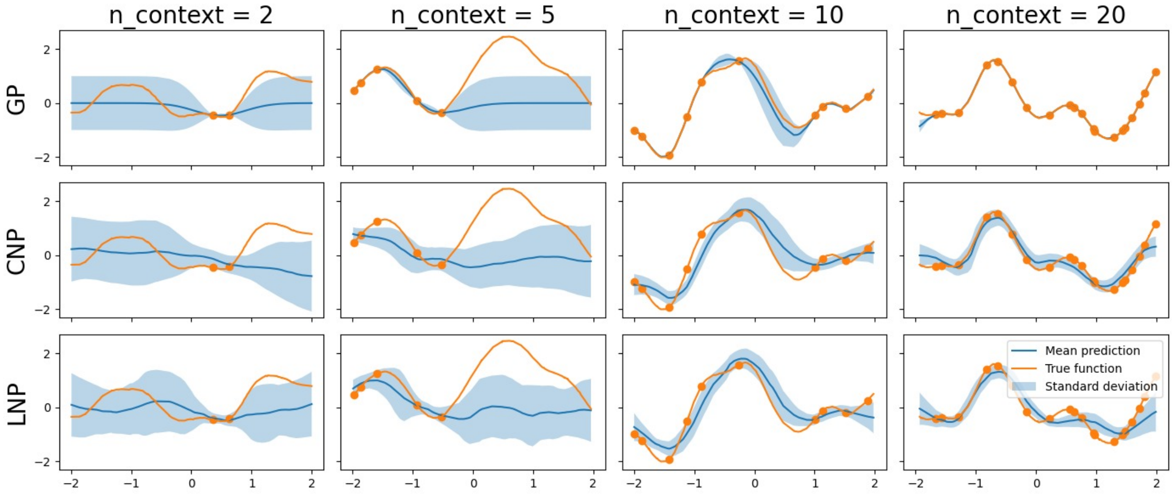


Figure 7: Comparison between GP, CNP and LNP on the 1D-regression task (fixed kernel parameter)

We can observe from the results in Figure 7 that both CNP and LNP have good performance with respect to the original GP for the first 3 columns. The GP variance is smoother than the CNP's and the function distribution learned by the CNP is not fully confident about the context points. CNP cannot achieve perfect reconstruction because of the bottleneck at the representation level after the encoder. As there are no quantitative measures of results in the original paper, we are only able to compare our results visually. Our results look similar to the results in the original paper and we are satisfied that we have replicated their experiments [1] [2].

It's worth noting that neither CNP nor LNP can make good out of distribution predictions, for regression, this means that our results are only valid within the domain $x \in [-2, 2]$. However Convolutional Conditional Neural Process (CONVCNP) [3] are translation invariant so could be used in their place.

5.1.2 Switching-kernel distribution

To demonstrate the power of NPs over GPs, we replicate another 1D-regression experiment conducted by Garnelo et al. [1], what we refer to as a switching-kernel distribution: in our case study such a distribution means that any drawn function switches at some random point on the real line between two functions each sampled with different kernel parameters. To implement that, we tweaked the dataloader to randomly draw two kernel parameters l_1 and l_2 associated to two GPs for each generated example. We randomly split the ordered x values and generate sample points for each side of the split with their respective GP. We also trained and evaluated a LNP model on this task.

To fit a GP to samples generated from this distribution would require guessing the point at which the kernel parameters switch from l_1 to l_2, which would require changes to the GP model. Thus, this demonstrates CNPs and LNPs generating uncertainty predictions in a scenario that GPs are not designed to handle.

Therefore, we should compare the mean and standard deviation with the true function. On Figure 8, we can observe two different scenarios. First for a low number of context points (3 and 8 context

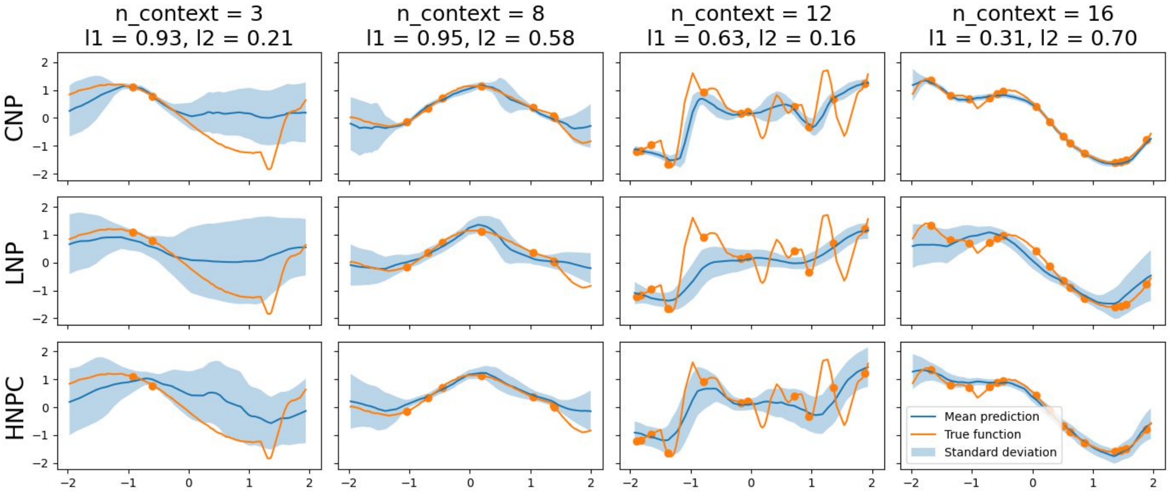


Figure 8: Comparison between CNP, LNP and HNPC (introduced later in the report in section 6.3.1) on the 1D-regression task with varying kernel parameters. Each column column corresponds to given number of context points and an arbitrary kernel length parameter.

points for this example), LNP is better with respect to the uncertainty because CNP is overconfident in its predictions. However, for a relatively high number of context points (12 and 16 context points), CNP performs better as LNP underfits the mean function of the distribution.

5.2 Image Completion

Our goal is to perform image completion: given an image with only a fraction of pixels available, the model should be able to reconstruct the original image. In the case of images we do not have access to the underlying stochastic process, but we view each true image as a sample from the underlying distribution. We focus on a qualitative evaluation of the outputs and use the true images as a reference for comparison.

5.2.1 MNIST

First, we use CNP to complete an image from MNIST [10] given a limited number of context points. The grayscale values of each pixel in flattened from a 2D grid to a 1D vector to make it a suitable input for NPs.

The results displayed in Figure 9 demonstrate that the model is able to provide accurate predictions of digits with a limited number of context points. Note that for the top-left input with 1 uninformative context point, the model outputs an average of all MNIST digits. As the number of context points increase, the mean predictions become more similar to the ground truth. The final image with 1000 context points matches the digit relatively closely, although it is still not a perfect reconstruction even in the areas with all context points revealed.

For each shown example, lighter pixels mean a higher variance. Note that as the number of context points increases, the variance shifts from being almost uniformly spread over the digit, to being localised around the edges of the digit. Indeed, having a larger line thickness would not change the class of a digit.

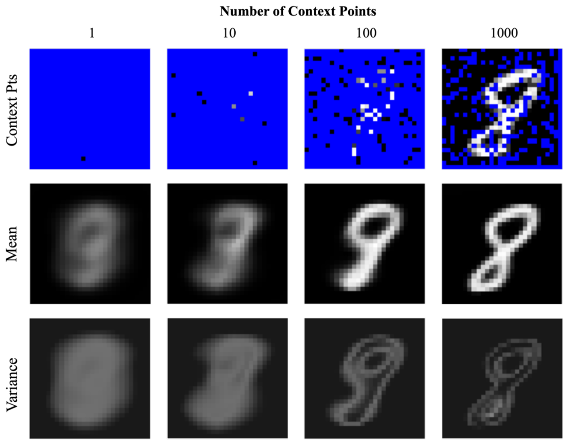


Figure 9: Example of image completion on MNIST dataset using CNPs.

We obtained almost identical results to the authors of the original paper [1]. When there is a single informative context point, the area around that context point is more confident and the model becomes less confident further from that point. In the original paper, there is more incoherence to the mean and variance predictions outside of the context points, moving back towards the mean outside the context points. However, this is not the same example as shown in the paper, so this phenomenon may appear on other examples, as the example used in the paper is a one but we show an eight, as eight could be easily confused with other numbers.

Note that in the CNP paper, Garnelo et al. additionally ran an experiment which showed that adding context points in accordance with the pixel with most variance instead of choosing at random significantly improved the loss for a given number of context points. As a result, the digit was guessed with less context points.

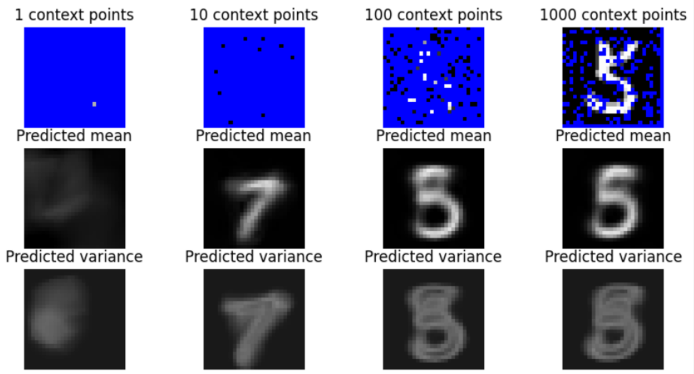


Figure 10: Example of image completion on MNIST dataset using LNPs

We also tested LNP on MNIST which theoretically produces more coherent images than CNP due to the random latent variable[1]. Accordingly, given just 10 uninformative samples of the target distribution, a clear digit is guessed, but this digit is picked randomly. This is far more reasonable than CNP which simply predicts a blob which is the mean over all images it was trained on. We get different results for the digit guessed (represented by the mean) with the 10 context points given in the 2nd column each time we run this prediction. This is also found in the LNP paper [2]. As the number of context points increases the 7 morphs into the correct digit a 5. It's also worth noting that, in general, CNP predicts sharper images with less data, although this could be due to its 'overconfident' deterministic latent representation. Also the general predictions of LNP are more generic and lack the idiosyncrasies and subtle features present in the context samples.

5.2.2 Spatial coherence with sampling constraint

The goal of this subsection is to assess the CNP's capacity to capture global spatial properties. We kept the same training procedure as section 5.2.1 but only sample context points from one half of the image during inference. Looking at the results in Figure 11, both CNPs and LNPs manage to reconstruct the 8 digit while keeping a relative symmetry. Looking more carefully at our results, symmetry is more consistent for LNP. This goes hand-in-hand with our observations from section 5.2.1: in zones of low context point density, CNP is more likely to output the average of all classes which is seen here by the blurry aspect of the right-side of the 8 digit. This is not the case for LNP as the latent variable allows global coherence.

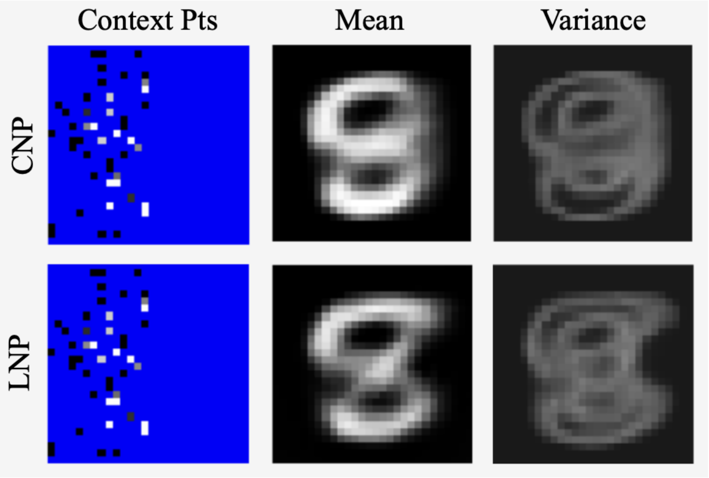


Figure 11: Sampling from one half of the image only, forcing CNP to extrapolate the remaining half

Note that the original paper [1] also proved the ability of CNP to handle image completion for different resolutions - including previously unseen ones.

5.2.3 Spatial coherence with mixture of images

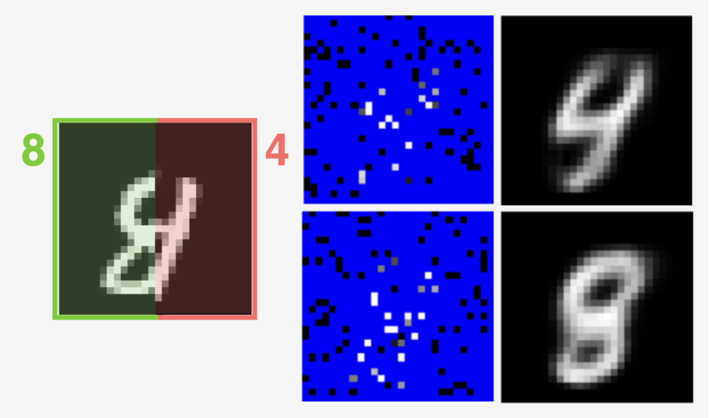


Figure 12: Combining two images, sampling the CNP result can yield either parent image

As an additional experiment, which was not displayed in the original paper, we used CNP to predict out of distribution samples, created by concatenating two different digit image halves together. Interestingly, when drawing samples, the CNP reproduced either digit. The network predicted 8 more often in this example, which is desired as the new digit appears similar to an 8 (Figure 12). This demonstrates that the model is interpreting small subtleties in local regions rather than just the global context.

5.2.4 CelebA

Next, we replicate the results by Garnelo et al. to demonstrate more complex image completion on the CelebA dataset [11]. CelebA images are bigger (original size of 178 × 218, then resized to square format) in colour (3 channels), and the distribution is more diverse (40 classes).

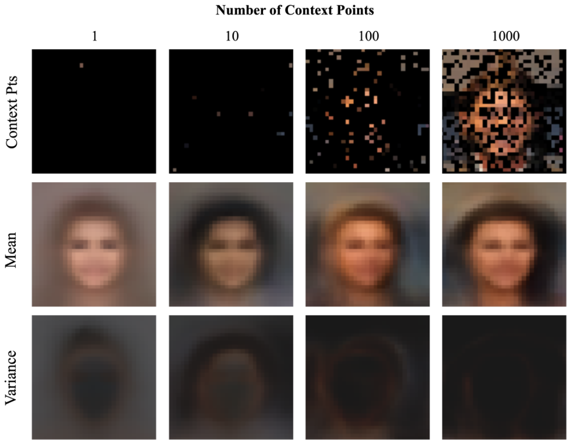


Figure 13: Image completion on CelebA dataset using CNP.

Similar observations can be made for image completion using CNP on CelebA as we made in subsection 5.2.1. With few context points the predicted face looks like the mean over all faces. With more context points, the face attributes become more defined. Lighter pixels imply higher variance, and the variance decreases as more context points are revealed and the variance increases away from the context points, in a similar manner to the original paper. However, the model can be overconfident about the predictions that it makes with few context points.

The model is quite confident about the middle of the face while being quite uncertain with the face edges. The model managed to reconstruct worn accessories such a the sunglasses seen in Figure 14. As mentioned before, CNP cannot achieve perfect reconstruction with a high number of context points: hence a blurry prediction even with 1000 context points Figure 13.

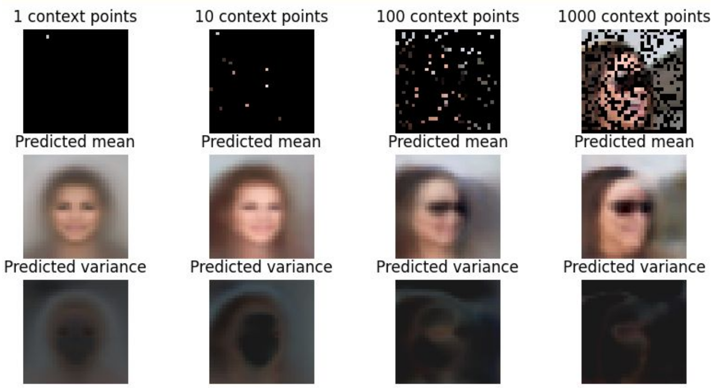


Figure 14: Image completion on CelebA dataset using CNP for a person wearing sunglasses

We do not replicate the CelebA experiment using LNPs as they are not the main focus of this report and the MNIST experiments provide a strong basis for comparison between CNPs and LNPs already.

6 Extensions

6.1 Latent Representation

We investigated how the distribution of the latent space changed during training. Figure 15 shows how the prior latent distribution matches the posterior latent distribution throughout training. This is thanks to the KL divergence term in the LNP loss function that seeks to minimise the difference between these two distributions. Practically this means that the latent representation for a subset of a training example will always be the same as the latent representation for the entire training example. This is a useful characteristic to extrapolate an entire distribution from a subset of its points. The figure also demonstrates that the non-deterministic latent representation is being fully utilised as the standard deviation remains strongly positive, we were concerned that the standard deviation would tend to 0, causing the latent representation z to behave deterministically, like r does in a CNP.

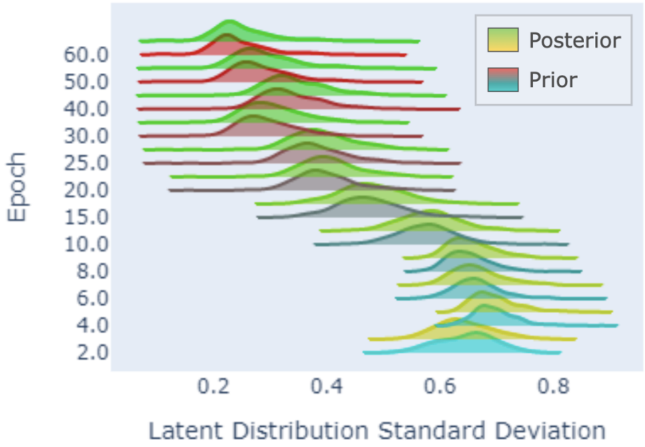


Figure 15: Spread of standard deviation across all dimensions of latent space distribution in LNP model over training epochs.

6.2 Hybrid Neural Processes

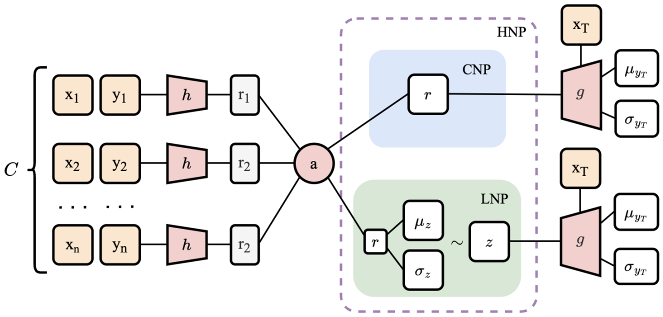


Figure 16: Architecture diagram of HNPs.

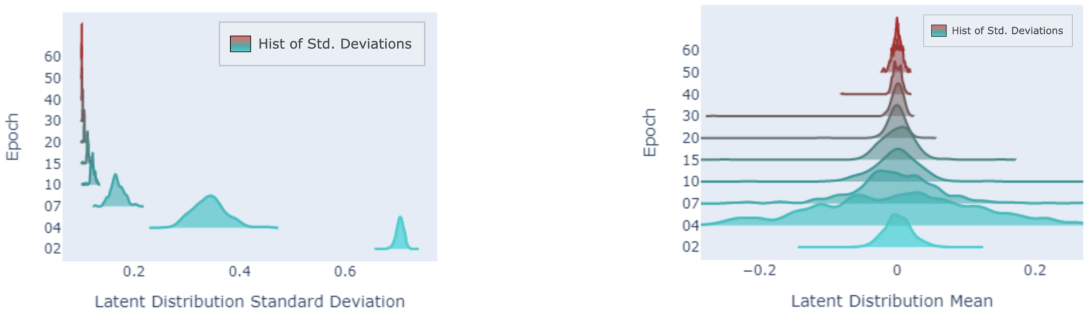
Our final experiment involves combining Conditional and Latent Neural Processes, in what we are calling Hybrid Neural Processes (HNPs). These are mentioned by Le et al. [9], but are not discussed in depth. It is also possible that the HNP model is what is referred to by Garnelo et al. in [1] as a Latent Conditional Neural Process, but the discussion of latent CNPs is short and the model is not fully specified, so we cannot be sure.

Our hypothesis is that HNPs, which combine the deterministic link between the context representations (used by CNP) with the non-deterministic link from the latent space representation space (used by LNP), will result in a model with a richer embedding space.

Given the HNP structure, HNPs are suitable to be trained using the optimisation objective for LNPs (described in section 3.3.2), which includes both a KL-divergence term to match the prior and posterior distributions over the latent variable z, as well as a conditional log probability term which penalises the model for an unlikely true target output under the output mean and variance.

We view the latent space as N independent Gaussian distributions, one for each dimension of the

N-dimensional latent vector, with means $\{\mu_{z_i}\}_{i=1}^{N}$ and variances $\{\sigma_{z_i}\}_{i=1}^{N}$. We plot the statistics of these N means and variances to visualise the evolution of the latent distribution over training epochs. A larger spread indicates a meaningful latent space.



(a) Standard deviations across all dimensions.

(b) Means across all dimensions.

Figure 17: Showing statistics of mean and variance across all dimensions of latent space distribution in HNP model.

When a deterministic link is included between the encoder and decoder, the mean and standard deviation of the latent distribution tends towards zero during training (Figure 17a, Figure 18). This makes the samples z behave deterministically, causing the model to act the same as an equivalent CNP that only uses a deterministic latent representation r. In other words, HNPs converge to CNPs. If we contrast this with the latent representation from Figure 15 you can clearly see that the representation learnt by LNPs fully utilises the latent space and the representation learnt by HNPs does not.

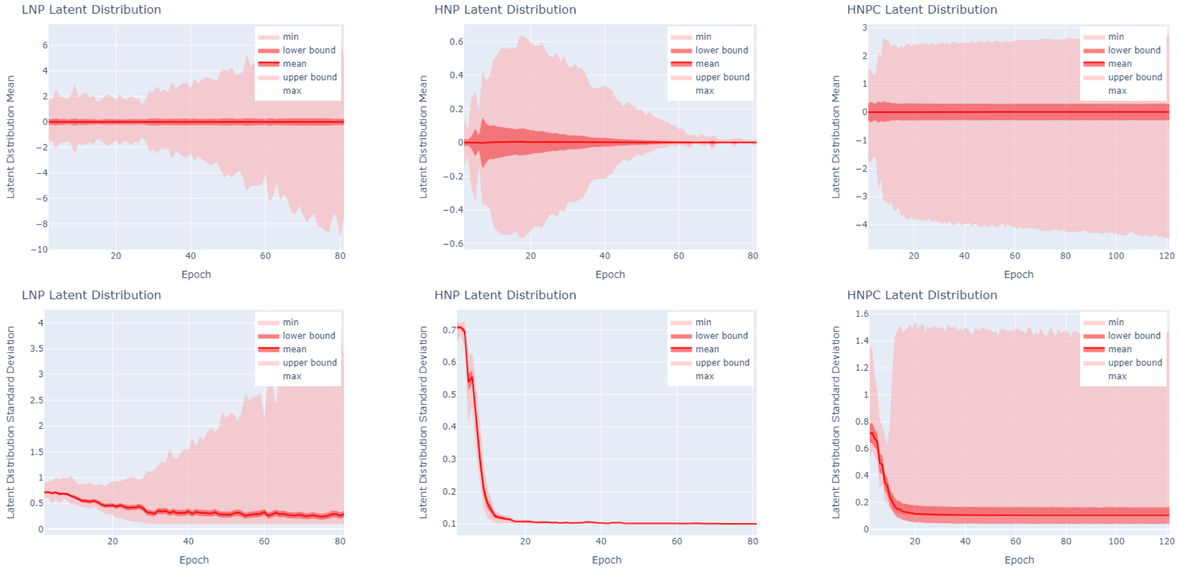


Figure 18: Latent Variable Distribution - Mean and Standard Deviation Statistics during training. (HNPC is introduced later in subsection 6.3)

Throughout training, the mean μ_{z_i} of the LNP remains at zero and the standard deviation of the mean increases. The mean σ_{z_i} converges to around 0.2 and the variance of the σ_{z_i}'s increases as it captures the stochasticity of the data. In HNP, the mean μ_z and standard deviation σ_z converge towards 0 and the latent representation becomes less meaningful.

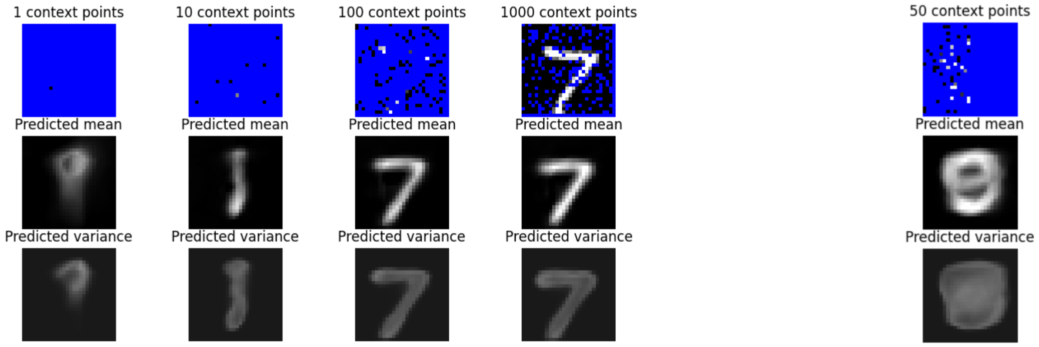
6.3 Constrained Hybrid Neural Processes

To force HNPs to learn a meaningful latent space, two additional terms are added to the loss function. We call this new model the constrained hybrid neural process (HNPC). These terms prevent the mean and standard deviation from falling below thresholds α and β. The cost is proportional to the $L2$ norm between the thresholds α and β and the measured mean and standard deviation. This is shown in Eq. 10 where γ is a hyperparameter used to weight the contribution of the two additional loss terms.

$$\begin{aligned}\mathcal{L}_{\text{HNPC}} = \mathcal{L}_{\text{LNP}} &+ \gamma \begin{cases} \|\alpha - \sqrt{Var(z|C)}\|^2 & \sqrt{Var(z|C)} < \alpha \\ 0 & otherwise \end{cases} \\ &+ \gamma \begin{cases} \|\beta - \mathbb{E}(z|C)\|^2 & \mathbb{E}(z|C) < \beta \\ 0 & otherwise \end{cases}\end{aligned} \tag{10}$$

Equation 10: HNPC loss where $\mathcal{L}_{\text{LNP}}$ is the LNP loss (Eq. 9) and the remaining terms discourage small standard deviations of the mean and variance parameters in the latent multivariate Gaussian distribution

Now the HNPC mean and standard deviation no longer converge to 0 (Figure 18). However, the HNPC μ_z and σ_z standard deviations remain relatively constant, suggesting that the model is possibly cheating by setting the weights of the first layer in the decoder to be zero for those inputs that are from the latent representation z.



(a) Varying numbers of context points.

(b) Context points on half the image.

Figure 19: Running MNIST experiments using constrained HNP (HNPC).

6.3.1 HNPC Experiments

First, we could already observe the versatility of HNPC on section 5.1.2 where the model was better than CNP and LNP in capturing the distribution variance.

Moreover, we run the basic MNIST image completion experiment using the HNPC to compare performance with CNP and LNP. Figure 19a demonstrates that HNPC has some characteristics of both

CNPs and LNPs. With one uninformative context point, the output is some version of an average number. With 10 context points, HNPC guesses a number, 1 in this case, which is characteristic of LNP. The final prediction given 1000 context points is sharp, much like the images produced by CNP Figure 9. However, in the the half image completion task (Figure 19b) the predicted mean is blurry across the entire image with high variance, showing that HNPC can sometimes perform worse than both CNP and LNP. In section 5.1.2, we found similar results on 1D-regression, that the HNPC mean and uncertainty predictions are between those from CNP and LNP. This reinforced our conclusion that HNPC gives predictions between CNP and LNP in terms of performance and characteristics.

HNPs attempt to address underfitting issues with inference in $\mathcal{O}(n+m)$ time, but it is not successful, instead ending somewhere between a CNP and an LNP in performance. This suggests that further improvements must be made to HNPs. In fact, another model from the NP Family, Attentive Neural Processes (ANP) proposed by Kim et al. [5], follows from the idea of passing deterministic information to the decoder. ANPs are slightly slower, performing inference in $\mathcal{O}(n(n+m))$ time, but are able to reconstruct input functions and overcome underfitting. Rather than passing additional information about the context set to the decoder as in HNPCs, ANPs directly pass information about the target signal through the encoder using an attention mechanism, applying the idea that target points close to a context points are likely to have similar output values.

7 Evaluation of Our Work

We faced some difficulty replicating the code written in the original Conditional Neural Process paper. We had access to an official repository for CNPs created by the authors of the CNP paper, but this codebase was written in Tensorflow v1. Thus, we decided to replicate the architecture of the model in Tensorflow v2, but still ran into many difficulties building the dataloader and optimising model training performance. We implemented LNPs from scratch while considering some third-party PyTorch repositories. Overall, we developed an in-depth understanding of Tensorflow functions, autograd, optimisation and data pipelines, and learnt how to optimise the hyperparameters for Neural Processes and their loss functions.

The results of our experiments matched the original paper to the best of our knowledge. It is difficult to critique our results further given that the original paper did not provide quantitative results.

Finally, the only experiment from the original CNP paper we did not reproduce is the one-shot classification using the Omniglot dataset, otherwise, we provide a comprehensive review of the work in CNP[1]. Additionally, we extend much further by replicating many Latent Neural Processes experiments.

8 Conclusion

In this work, we replicated the results from Garnelo et al. for CNPs and LNPs [1] [2]. As was presented in the original paper, we demonstrated how NPs are able to tackle several challenges that cannot be solved by neural networks nor GPs alone. In particular, NPs learn to represent distributions over functions and are able to make fast and flexible predictions conditioned on some context input. On top of that, NPs are able to learn an implicit prior from the data directly, which - in high-dimensional problems - is more effective than empirically engineering the prior.

Through our experiments, we observed that although CNPs and LNPs are “data-efficient” at test time, as they only need a few context points to make inferences over a given function, they still require millions of training examples to provide reasonable predictions, so are much less efficient than GPs with respect to their training complexity. Note that this latter observation was not mentioned in the original paper, but we deemed it worth mentioning.

We applied NPs to a range of regression tasks, including 1D-regression and image completion, to assess their flexibility and showed equal performance to the original paper on these tasks. We replicated the main conclusion of the papers, which is that CNPs and LNPs do not model variance perfectly and often underfit to context points due to the encoder information bottleneck. For this reason, it is not possible to use LNPs or CNPs as generative functions as in diffusion models, as the latent space is too large of a bottleneck to generate reconstructions.

9 Further work

Given more time, we would have shown an experiment that demonstrates the difference between CNP's point-wise sample generation and LNP's coherent sample generation. For example, for the 1D-regression case, functions drawn from a CNP will have a great deal of oscillation as each sample will be unrelated from its neighbours, and the function will not directly look like a function from the underlying distribution. However, functions drawn from a LNP will be globally coherent. This experiment is helpful to compare LNPs and CNPs, but we did not have time to complete it.

Current research about the effects of including the covariance between context and target points also sparked our interest. Yoo et al. explored learning covariance efficiently in NPs by enforcing it through the loss function [13]. With more time, we would have liked to consider the differences between the model from Yoo et al. and CNPs and LNPs.

Finally, constraining the loss function for HNPCs in the manner that we chose was just one attempt to improve prediction performance. The HNPC model could have been iteratively improved with more time.

References

[1] Marta Garnelo et al. *Conditional Neural Processes*. 2018. DOI: 10.48550/ARXIV.1807.01613. URL: https://arxiv.org/abs/1807.01613.

[2] Marta Garnelo et al. *Neural Processes*. 2018. DOI: 10.48550/ARXIV.1807.01622. URL: https://arxiv.org/abs/1807.01622.

[3] Jonathan Gordon et al. *Convolutional Conditional Neural Processes*. 2020. arXiv: 1910.13556 [stat.ML].

[4] Saurav Jha et al. *The neural process family: Survey, applications and Perspectives*. 2022. URL: https://arxiv.org/abs/2209.00517.

[5] Hyunjik Kim et al. *Attentive Neural Processes*. 2019. arXiv: 1901.05761 [cs.LG].

[6] Diederik P Kingma and Max Welling. *Auto-encoding variational Bayes*. 2022. URL: https://arxiv.org/abs/1312.6114.

[7] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: (2014). DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

[8] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. "Human-level concept learning through probabilistic program induction". In: *Science* 350.6266 (2015), pp. 1332–1338. DOI: 10.1126/science.aab3050. eprint: https://www.science.org/doi/pdf/10.1126/science.aab3050. URL: https://www.science.org/doi/abs/10.1126/science.aab3050.

[9] Tuan Anh Le. "Empirical Evaluation of Neural Process Objectives". In: 2018.

[10] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[11] Ziwei Liu et al. "Deep learning face attributes in the wild". In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015). DOI: 10.1109/iccv.2015.425.

[12] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning.* Adaptive computation and machine learning. MIT Press, 2006, pp. I–XVIII, 1–248. ISBN: 026218253X.

[13] Boseon Yoo et al. “Conditional Temporal Neural Processes with Covariance Loss”. In: *Proceedings of the 38th International Conference on Machine Learning.* Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, July 2021, pp. 12051–12061. URL: `https://proceedings.mlr.press/v139/yoo21b.html`.