UNIVERSITY OF
CAMBRIDGE

Autoregressive Conditional Neural Processes

Advanced Machine Learning

Afiba Annor
Minh Dat Hong
James Xu
Paulina Körner

Word count: 4992

May 20, 2025

Contents

Introduction

In this report, we replicate the experiments from the paper *Autoregressive Conditional Neural Processes* [1], compare our results to those presented in the original work, and extend the study. The paper introduces an autoregressive (AR) inference procedure for Neural Processes (NP), with a particular focus on Conditional Neural Processes (CNP). This approach addresses a key limitation of standard NP architectures, which often struggle to capture dependencies between outputs in joint predictions.

The experiments we replicate from the original work include synthetic regression tasks and Sim-to-Real transfer using Lotka-Volterra dynamics.

Beyond replication, we explore several extensions of the autoregressive inference framework:

- **Application to image data:** We investigate the performance of Convolutional Conditional Neural Processes (ConvCNP) and Autoregressive Convolutional Conditional Neural Processes (AR ConvCNP) on image datasets, specifically the Extended Modified National Institute of Standards and Technology (EMNIST) dataset for image completion [2]. These experiments aim to evaluate whether data with strong local correlations, such as pixel intensities in images, benefit from AR inference.
- **AR inference in Latent Bottleneck Attentive Neural Processes (LBANP):** We adapt the LBANP to incorporate the AR inference mechanism. Our goal is to determine whether similar performance improvements are observed when applying the AR regime.
- **AR Inconsistency:** We explore a limitation in the AR inference strategy: the resulting joint distribution over targets depends on their ordering, violating permutation consistency. To address this, we investigate alternative approaches to obtain a more robust and order-agnostic estimate of the joint target distribution.

Before presenting the replication results and the extensions, we first provide background on NPs in general, the specific NP models used in our experiments and the concept of AR inference.

Background

2.1 Neural Processes

NPs [3] are models that use Neural Networks (NN) to approximate stochastic processes by predicting outputs from a limited set of observations (the context set). They operate in two stages: *conditioning*, where the model encodes context input-output pairs into a global representation, and *querying*, where this representation is used to predict outputs for new target inputs.

In contrast to classical stochastic processes such as Gaussian Processess (GPs) [4], which rely on analytical kernel functions, NPs utilise learned neural architectures. This provides greater flexibility and scalability, although it may compromise exact adherence to theoretical properties. A valid stochastic process model should satisfy *exchangeability* - predictions should be invariant to the order of target points - and *Kolmogorov consistency* - predictions should remain compatible when considering subsets of data. NPs approximate these properties through permutation-invariant architectures and set-based training methods.

NPs are predominantly applied in meta-learning, where the objective is to generalise across a distribution of tasks using few examples. Training involves sampling a function from a meta-dataset, splitting it into context and target sets, and updating the model parameters by maximising the log-likelihood of the targets under its predictive distribution. The typical NP architecture follows an encoder-decoder structure. The encoder independently processes each context point using a shared network and aggregates the representations via a commutative operation (e.g. summation or averaging) to ensure order invariance. The decoder then combines the global representation with each target input to produce predictive distributions. This report focuses on several NP variants, specifically the ConvCNP, Convolutional Gaussian Neural Processes (ConvGNP) and LBANP.

2.1.1 Conditional and Convolutional Neural Processes

To make predictions, CNPs [5] first encode the context set $(x_i^{(c)}, y_i^{(c)})$, for $i = 1, \ldots, M$, using a permutation-invariant function (often implemented as a Deep Set) that aggregates the information into a fixed-dimensional latent representation R. Then, they use a decoder which, given this aggregated representation and a target input, outputs a prediction (typically a Gaussian with a mean and variance). CNPs assume that, conditioned on R, predictions for target points are independent:

$$p_\theta(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)}; \mathcal{D}^{(c)}) = \prod_{i=1}^{N} p_\theta(y_i^{(t)}|x_i^{(t)}, R). \tag{2.1}$$

This factorisation simplifies training via maximum likelihood estimation and allows for efficient computation with complexity $O(T + C)$. However, it also introduces significant limitations:

- **Underfitting**: Relying on a single global representation R limits expressivity and fails to capture correlations between targets.
- **Incoherent Samples**: The factorised structure prevents the generation of consistent joint samples over the target set.

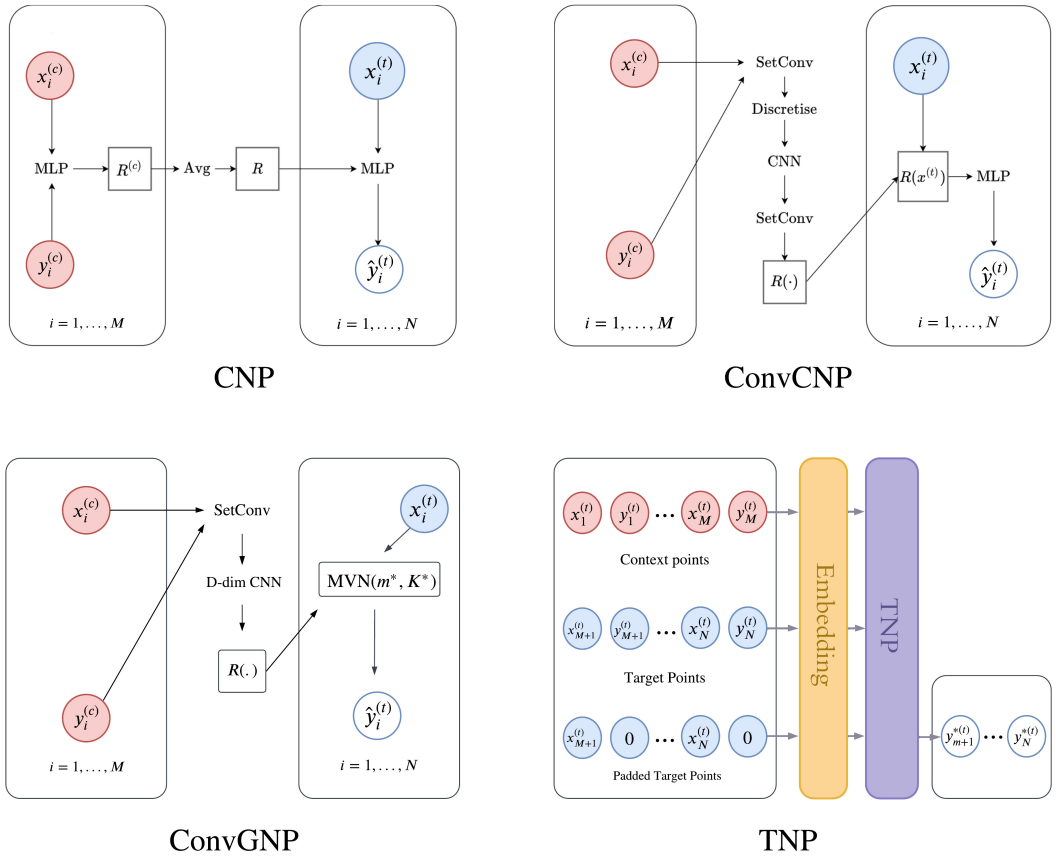

Figure 1: Comparison of four Neural Process (NP) architectures. (*Top-left*) CNP, which encodes the context set via an Multi-Layer Perceptron and then predicts each target point independently. (*Top-right*) ConvCNP, which introduces *translation equivariance* by encoding context points into a continuous representation that is processed by a convolutional backbone. (*Bottom-left*) ConvGNP produces a correlated Gaussian predictive distribution over the target set. (*Bottom-right*) TNP, which employs a transformer-based embedding to capture richer dependencies among context and target points. Figure adapted from [6].

Standard CNPs depend on absolute input positions, which limits their ability to generalise in applications such as time series and images where translation equivariance (TE) is crucial. ConvCNPs [7] address this shortcoming by encoding the context set C with convolution layers. This function is then queried at any target location $x_i^{(t)}$, ensuring that predictions shift in tandem with the inputs. To reconcile set-based inputs with grid-based convolutions, ConvCNPs employ the SetConv layer, a generalisation of standard convolutions to unordered sets and continuous queries. Despite having better performance than CNP, Gaussian and independent predictions assumption still limits ConvCNP's ability to model multimodal and correlated distributions, sometimes resulting in outputs that lack global coherence.

2.1.2 Gaussian Neural Process

Fully Convolutional Gaussian Neural Processes (FullConvGNP) [8] extend the CNP family by explicitly modelling correlations between target outputs through a full covariance structure. Unlike standard CNPs, which assume independent predictions for each target point, Gaussian Neural Processes (GNP) define a multivariate Gaussian predictive distribution over the target set:

$$p_\theta(\mathbf{y}^{(t)}|\mathbf{x}^{(t)}, \mathcal{D}^{(c)}) = \mathcal{N}(m^*, K^*), \tag{2.2}$$

where m^* and K^* denote the predictive mean vector and full covariance matrix over the target outputs.

The FullConvGNP computes the full covariance by mapping the context to a high-dimensional grid and applying convolutions in a 2D-dimensional space, leading to a computational cost that scales quadratically with the number of target points and incurs significant memory overhead. This renders it impractical for high-dimensional inputs. To alleviate this, the ConvGNP [9] reparameterises the covariance using standard D-dimensional convolutions to extract context-dependent features, reducing the complexity to linear (with a linear covariance kernel) while preserving TE and capturing output correlations. However, despite this efficiency gain, the ConvGNP remains computationally demanding for large target sets when using more complex kernels and is limited to Gaussian predictive distributions, restricting its ability to model multimodal or heavy-tailed data.

2.1.3 Transformer Neural Process

Transformer Neural Processes (TNP) [10] evolved from these earlier families by reformulating meta-learning as a sequence modelling problem. In TNPs, context and target datasets are jointly encoded using a transformer architecture with layers of masked self-attention that enforce permutation invariance and target equivariance, thereby capturing complex, higher-order interactions and producing coherent joint predictions with robust uncertainty estimates - albeit at a quadratic computational cost relative to the total number of points. To mitigate this inefficiency, latent bottleneck architectures, such as LBANPs [11], introduce a fixed set of latent vectors that summarise the context via cross-attention and subsequent self-attention among the latent vectors; this design reduces the conditioning phase complexity to $O(NL+L^2)$, where L is fixed, and allows each target query to interact only with these latent representations, yielding a query complexity that scales linearly with the number of targets. We consider the default setting of LBANP in this work, which only makes independent predictions.

2.2 Autoregressive Inference

2.2.1 Autoregressive Conditional Neural Processes

As discussed, a limitation of models under the CNP class is that they assume the target outputs are conditionally independent - they target points simultaneously, without considering the relationships between them. Consequently, CNPs cannot produce coherent predictions or model complex distributions with multiple modes [1].

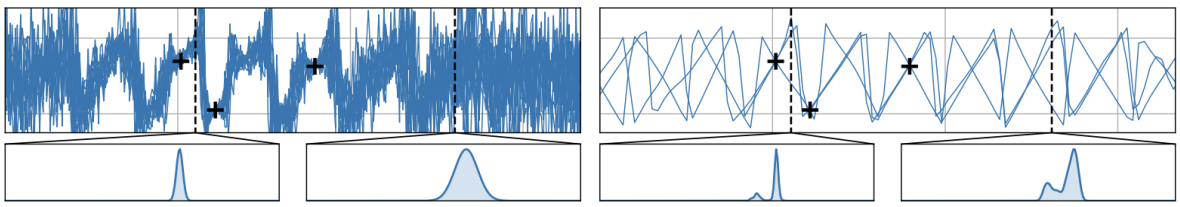

Figure 2: Comparison of ConvCNP sawtooth function predictions using standard inference (*left*) and autoregressive inference (*right*). The black crosses indicate the observations and the plots below indicate the marginal predictive distributions at the dashed vertical lines. Autoregressive inference results in more coherent samples and captures multimodality. Figure reproduced from [1].

Many problems require capturing dependencies between predictions, such as predicting heat waves or floods [9]. Standard NPs that do model dependencies can be more computationally expensive or

challenging to train [9]. The question arises: can we make dependent predictions without changing the model architecture or retraining?

The autoregressive process, proposed by [1], achieves this for CNPs by finding a joint distribution over the target dataset $\mathcal{D}^{(t)}$. This is done by sampling target outputs sequentially and leveraging the product rule. The process proposed can be extended to any NP.

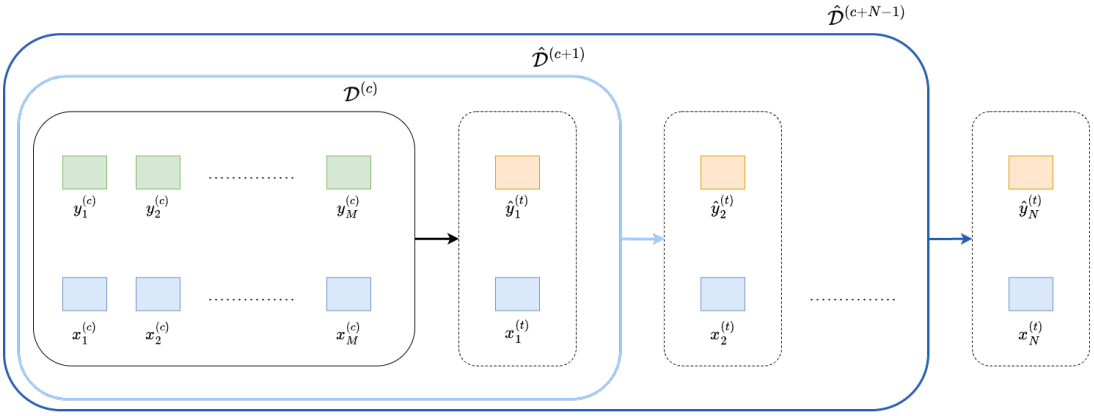

Figure 3: Diagram of autoregressive inference process for AR NPs.

Consider a NP π_θ with context dataset $\mathcal{D}^{(c)} = (\boldsymbol{x}^{(c)}, \boldsymbol{y}^{(c)})$ and target inputs $\boldsymbol{x}^{(t)}$. Figure 3 demonstrates the sampling procedure. Starting with the first target point $x_1^{(t)}$, we condition on the context dataset $\mathcal{D}^{(c)}$ which induces the following predictive distribution over $y_1^{(t)}$:

$$y_1^{(t)} \sim P_{x_1^{(t)}} \pi_\theta \Big(\boldsymbol{x}^{(c)},\ \boldsymbol{y}^{(c)} \Big) \tag{2.3}$$

From this distribution, a sample value is drawn for the target output, $\hat{y}_1^{(t)}$. To predict the next target point $x_2^{(t)}$, we condition on both the context set and the previous target points, $\hat{\mathcal{D}}^{(c+1)} = \mathcal{D}^{(c)} \cup \big(x_1^{(t)}, \hat{y}_1^{(t)}\big)$. The target output $\hat{y}_2^{(t)}$ is then sampled from the induced distribution. This process is conducted iteratively for the rest of the target points. The generalised predictive distribution for the target outputs is as follows:

$$\text{for } i = 1, ..., N, \quad y_i^{(t)} \sim P_{x_i^{(t)}} \pi_\theta \Big(\boldsymbol{x}^{(c)} \oplus \boldsymbol{x}_{1:(i-1)}^{(t)},\ \boldsymbol{y}^{(c)} \oplus \boldsymbol{y}_{1:(i-1)}^{(t)} \Big) \tag{2.4}$$

where $\mathbf{a} \oplus \mathbf{b}$ concatenates two vectors $\mathbf{a}$ and $\mathbf{b}$ [1].

Since each step is conditioned on the previous outputs, the product rule can be used to define the joint predictive distribution $q_\theta^{(AR)}$:

$$q_\theta^{(AR)}(y_1^{(t)}, \ldots, y_N^{(t)} \,|\, \boldsymbol{x}^{(t)}, \mathcal{D}^{(c)}) \ = \ \prod_{i=1}^{N} q_\theta \Big(y_i^{(t)} \,\Big|\, x_i^{(t)},\ \mathcal{D}^{(c)} \oplus (\boldsymbol{x}_{1:i-1}^{(t)}, \boldsymbol{y}_{1:i-1}^{(t)}) \Big) \tag{2.5}$$

This process allows correlations to be modelled in predictions, without changing the model itself.

2.2.2 Denoising Samples

To reduce the observation noise present in the samples, we apply a denoising step to smoothen samples. This is conducted by feeding the target inputs $\boldsymbol{x}_{1:N}^{(t)}$ along with their corresponding sampled outputs $\hat{\boldsymbol{y}}_{1:N}^{(t)}$

back into the NP model π_θ. Conditioning on this set, the model is queried at the same target input locations simultaneously. From the resulting predictive distribution,

$$\mathcal{N}(\boldsymbol{\mu}_{1:N}, \boldsymbol{D}) \;=\; P_{\boldsymbol{x}^{(t)}_{1:N}}\, \pi_\theta\Big(\mathcal{D}^{(c)}_M \oplus (\boldsymbol{x}^{(t)}_{1:N}, \hat{\boldsymbol{y}}^{(t)}_{1:N})\Big) \tag{2.6}$$

we obtain the predictive mean function $\boldsymbol{\mu}_{1:N}$ [1]. Proposition 2.2 in [1] states that, given an infinite number of points, this predictive mean converges to the true underlying function with the noise removed. Hence, by obtaining the predictive mean, we generate samples with less noise.

2.2.3 Model Trade-offs

NPs can exhibit trade-offs between consistency, capturing dependencies between target points and generating non-Gaussian predictive distributions. Models under the GNP class are able to capture dependencies. However, to maintain tractability, these models assume a joint predictive Gaussian distribution. As a result, they do not perform well on tasks with non-Gaussian distributions.

Model	Consistent	Dependencies	Non-Gaussian
AR ConvCNP [1]	✗	✓	✓
AR LBANP	✗	✓	✓
ConvCNP [7]	✓	✗	✓
LBANP [11]	✓	✗	✓
ConvGNP [9]	✓	✓	✗

Table 1: Comparison of different NP model classes in terms of consistency, ability to model target dependencies and capability to produce non-Gaussian predictions.

As discussed, models under the CNP class and LBANPs do not model dependencies. Applying AR inference allows for such dependencies to be easily modelled, requiring no changes to the architecture or training process [1]. Additionally, AR inference enables the model to capture non-Gaussian predictive distributions, as it only assumes Gaussian autoregressive conditionals [1].

However, AR inference also introduces new challenges. It breaks permutation invariance and thus Kolmogorov consistency [1]. Hence, the performance becomes sensitive to design choices such as the order of the target points. It also increases computational cost at inference time, since the model must perform one forward pass per target point. To alleviate this, N target inputs could be divided into K batches, assuming points within each batch is conditionally independent. The AR inference process can then be conducted on each batch with N/K forward passes. However, this trades-off faster and less expressive predictions [1]. Experimentation with this method was conducted on the LBANP for synthetic regression tasks, with results given in Appendix 5

Replication

In this section, we replicate experiments from the original *Autoregressive Conditional Neural Processes* paper [1]. We focus on the synthetic regression tasks for a GP with an Exponential Quadratic (EQ) kernel, a sawtooth process and a mixture process, as well as the Sim-to-Real transfer experiment using Lotka-Volterra dynamics. A subset of models, ConvCNP and ConvGNP, from the original paper are selected for replication in order to conserve computational resources. Note that results differ slightly from those reported in [1] due to the models converging to a different local minima during training.

For the replication, we use the official code provided by the authors, available on the project's GitHub repository [12].

3.1 Synthetic Data Generation

This experiment compares the performance of models on three stochastic processes with one-dimensional inputs and outputs. Synthetic data is generated by uniformly sampling at random, in the range $[-2, 2]$, from the following processes [1]:

- EQ: A Gaussian task where samples are drawn from a GP with an EQ kernel.
- Sawtooth: A non-Gaussian task with random frequency, direction and phase.
- Mixture: A non-Gaussian task where samples are drawn, with equal probability, from the EQ, Matérn$-\frac{5}{2}$, weekly periodic or sawtooth process.

Context points are uniformly sampled at random between 0 and 30 whilst target points are fixed at 50 for the EQ and mixture process and at 100 for the sawtooth process.

	EQ Norm. KL to truth (↓ better)	**Sawtooth** Norm. log-lik. (↑ better)	**Mixture** Norm. log-lik. (↑ better)
ConvGNP	**0.01 ± 0.00**	2.23 ± 0.16	-0.17 ± 0.02
ConvCNP	0.41± 0.01	2.53 ± 0.05	-0.21 ± 0.04
AR ConvCNP	**0.01 ± 0.00**	**3.32 ± 0.01**	**0.43 ± 0.04**

Table 2: Model perfomance on three synthetically generated 1D processes: GP with an EQ kernel, sawtooth and mixture. The results compare ConvGNP, standard ConvCNP and AR ConvCNP. Autoregressive inference improves performance across all tasks.

For the EQ task, the ground truth is known and can easily be computed, therefore, the normalised Kullback-Leibler (KL) divergence is used to measure how close the models' predictive distribution is to the true distribution. Given that the true distribution is not a simple known probabilistic model for the sawtooth and mixture tasks, they are evaluated using the normalised log-likelihood; maximising the log-likelihood is equivalent to minimising the KL divergence [1].

As in the original paper, Table 2 shows that AR ConvCNP improves the performance of the standard ConvCNP, outperforming ConvGNP on the non-Gaussian tasks. For EQ, AR ConvCNP enables performance on par with ConvGNP.

3.2 Predator-Prey: Sim-to-Real Transfer

This experiment evaluates different NP models on real hare-lynx population data [13] and simulated predator-prey data using the Lotka-Volterra equations with stochastic components. The outcome is two time series X_t or Y_t with non-Gaussian likelihood and complex dependencies. The Lotka-Volterra equations were simulated on a dense grid to construct meta-data sets and then split for three different tasks. For the interpolation task, the data was randomly partitioned into context and target sets; for forecasting, a random time point was selected to separate past observations (context) from future values (target); and for reconstruction, one of the series, either X_t or Y_t, was divided in a forecasting manner with the complementary series appended to the context.

	Int. (S)	For. (S)	Rec. (S)	Int. (R)	For. (R)	Rec. (R)
ConvGNP	-3.47 ± 0.02	-4.17 ± 0.02	-3.68 ± 0.02	-4.17 ± 0.04	-4.57 ± 0.01	-5.08 ± 0.16
ConvCNP	-3.47 ± 0.02	-4.85 ± 0.02	-4.06 ± 0.02	-4.21 ± 0.04	-4.96 ± 0.01	-4.75 ± 0.06
AR ConvCNP	**-3.29 ± 0.02**	**-3.58 ± 0.02**	**-3.45 ± 0.02**	**-4.16 ± 0.04**	**-4.32 ± 0.01**	**-4.35 ± 0.04**

Table 3: Normalised log-likelihoods (higher is better) for ConvGNP, ConvCNP and autoregressive ConvCNP across six Sim-to-Real Lotka-Volterra tasks. Tasks include interpolation (Int.), forward extrapolation (For.), and reconstruction (Rec.), evaluated on both simulated (S) and real (R) data. Autoregressive inference improves performance across all tasks.

The results indicate that the autoregressive model (AR ConvCNP) consistently outperforms both ConvGNP and ConvCNP across all tasks in both simulated and real settings. This suggests that incorporating autoregressive inference is beneficial for enhancing predictive performance in modelling predator-prey dynamics.

Extension

4.1 EMNIST Image Completion

EMNIST is a benchmark dataset that extends the Modified National Institute of Standards and Technology (MNIST) [2] dataset by including digits and handwritten letters. We extended the evaluation of the AR inference mechanism to the EMNIST image completion task. This task is well-suited for AR models due to strong spatial dependencies and complex, non-Gaussian pixel distributions. We used the *balanced* split, which provides uniformly sampled classes of 28×28 grayscale images, each representing a single character.

Although EMNIST is typically used for classification, we follow prior work [5, 10, 11] and use it for image completion. Our aim is not state-of-the-art performance but to assess whether the ConvCNP benefits from AR inference in capturing spatial structure and non-Gaussian outputs.

Each image is interpreted as a function from 2D coordinates to grayscale values. Input x is a pixel coordinate, and output y is the corresponding intensity, normalised to $[0, 1]$. This framing yields a 2D meta-regression task, where each image is a sample from a distribution over image functions. Context and target sets are random pixel subsets. As in [10, 11], $N \sim U[3, 197)$ and $M \sim U[3, 200 - N)$. Inputs are scaled to $[-1, 1]$ and outputs to $[-0.5, 0.5]$ for stability.

Models were trained on digit classes (0-9). For evaluation, two test sets of 4096 samples were used: the EMNIST Seen set (digits) and the Unseen set (letters), allowing assessment of generalisation to novel classes.

Initially, we trained the models using a setup similar to that of regression and predator-prey experiments. We used a U-Net architecture with seven layers and 64 hidden channels, trained for 200 epochs. We used model checkpointing based on the validation log-likelihood and used the best model at the end of training. Table 4 shows normalised log-likelihoods for ConvGNP, ConvCNP, and AR ConvCNP.

AR inference improved performance over ConvCNP on the Seen set. However, log-likelihoods showed high variance and unusually high values for the Seen set. For comparison, the best model from [10, 11], TNP-A, achieved 1.54 ± 0.01 (Seen) and 1.41 ± 0.01 (Unseen).

	EMNIST Seen (0-9)	**EMNIST Unseen (10-46)**
	Norm. log-lik. (↑ better)	Norm. log-lik. (↑ better)
ConvGNP	1.42 ± 0.08	0.72 ± 0.30
ConvCNP	1.78 ± 0.12	$\mathbf{1.09 \pm 0.27}$
ConvCNP (AR)	$\mathbf{1.92 \pm 0.12}$	0.87 ± 0.35

Table 4: Normalised log-likelihoods for the EMNIST image completion task, trained without regularization. The model was trained only on digits (classes 0-9). The Seen test set contains digits, while the Unseen test set includes letters (classes 10-46) that were not seen during training.

Careful inspection revealed that the models exploited the log-likelihood objective by predicting very low variances in regions with little variation, such as the dark corners and edges of the images. Since these areas are almost always black, overconfident predictions (i.e., low variance) yield high log-likelihoods.

In contrast, in central regions where pixel values vary and the character appears, the model assigns high variance to hedge against errors. This avoids penalties for incorrect predictions while achieving high log-likelihoods, despite failing to reconstruct meaningful structure. Figure 4 illustrates this in the ConvCNP model, which produces a generic blurry white blob with high uncertainty in the centre and confidently predicts black background pixels.

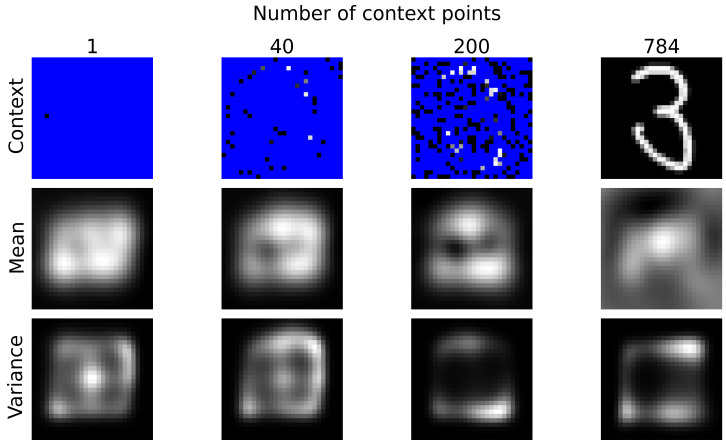

Figure 4: EMNIST image completion using ConvCNP (without autoregressive inference) on a character from the Unseen test set. Each column shows results for a different number of context pixels: 1, 40, 200, and 784. For each column, the context pixels (top row), predicted mean image (middle row), and predicted variance image (bottom row) are shown. The model fails to complete the character structure and instead predicts a generic white blob in the middle and the background with low uncertainty.

Formally, the ConvCNP is trained by maximising the log-likelihood under a multivariate Gaussian with diagonal covariance. Given predicted mean μ and variance σ^2, the log-likelihood of a target y is:

$$\log p(y \mid \mu, \sigma^2) = -\frac{1}{2} \sum_{i=1}^{D} \left[\log(2\pi\sigma_i^2) + \frac{(y_i - \mu_i)^2}{\sigma_i^2} \right], \tag{4.1}$$

where D is the number of output dimensions (pixels), and each dimension is treated independently.

This formulation allows the model to inflate log-likelihoods by predicting small variances in confident regions, even when predictions in the centre are poor. In those areas, it outputs blurry shapes with high variance to avoid penalties, resulting in misleadingly high scores despite the poor visual quality of the predictions.

To address this, we explored various training strategies to improve robustness. We applied curriculum learning - starting with many context points and few targets, then gradually reversing this - to increase difficulty. We also expanded model capacity by increasing U-Net depth and tested both uniform and pyramid-shaped hidden dimensions. Clamping predicted variances had little effect as the model continued to collapse the variances to the lower bound to artificially inflate log-likelihoods. Ultimately, we found that adding a regularisation term to the loss was the most effective strategy to prevent the model from exploiting the log-likelihood objective. This penalised the ConvCNP model for predicting extremely small variances, discouraging overconfidence. The modified loss is:

$$\mathcal{L}_{\text{total}} = -\log p(y \mid x) + \lambda \cdot \frac{1}{D} \sum_{i=1}^{D} \frac{1}{\sigma_i^2 + \epsilon}, \tag{4.2}$$

where D is the number of output dimensions (pixels), λ is the regularisation weight ($1e^{-3}$), and ϵ (10^{-4}) ensures numerical stability. The penalty increases as variances shrink, regularising the model's precision and discouraging overconfidence. Unlike harsher penalties (e.g., $1/\sigma^4$), which could cause exploding

gradients and unstable training, the $1/\sigma^2$ term is smoother and more stable. Although it limits the maximum attainable log-likelihood, it is a necessary trade-off to prevent variance collapse and overconfident predictions.

Regularisation was applied only to ConvCNP because its diagonal covariance enables simple variance regularisation. The ConvGNP model also exploited the log-likelihood (see Figure 11 in Appendix 5) but uses a full Gaussian covariance [8], which prevents independent variance regularisation and would require more complex penalties that we leave for future work.

Table 5 shows that regularisation produces more realistic log-likelihoods, consistent with prior findings [10, 11]. Figure 5 shows predictions from a regularised ConvCNP without AR inference. As context increases, predictions more closely match the true character. Variance is highest near character edges, where pixel values are uncertain, and lowest in background regions.

	EMNIST Seen (0-9)	**EMNIST Unseen (10-46)**
	Norm. log-lik. (↑ better)	Norm. log-lik. (↑ better)
ConvCNP	0.82 ± 0.01	0.72 ± 0.02
ConvCNP (AR)	**1.13 ± 0.01**	**1.04 ± 0.02**

Table 5: Normalised log-likelihoods on the EMNIST image completion task, trained with regularisation. Results are shown for both EMNIST Seen (digits 0-9) and Unseen (letters 10-46) test sets.

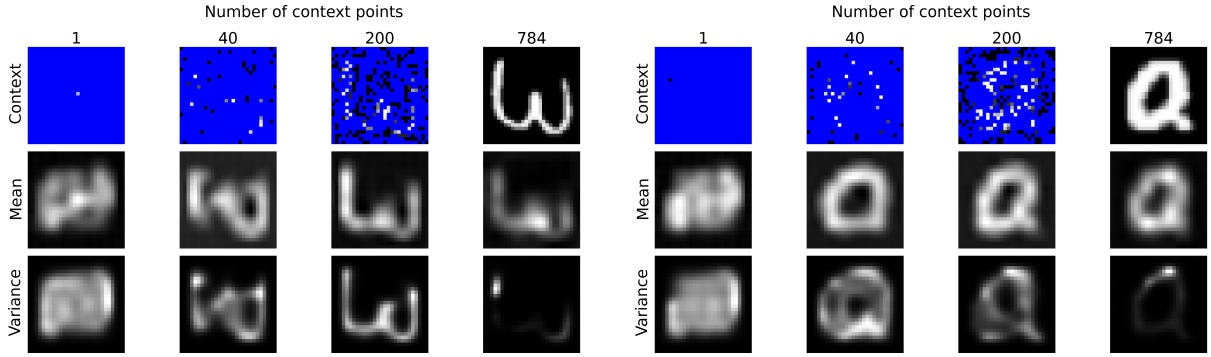

Figure 5: EMNIST image completion using ConvCNP with regularisation and without autoregressive inference. Left: digit from the Seen set (0–9). Right: letter from the Unseen set (10–46). With one context point, the model predicts a generic blob, as expected. As the number of context points increases, predictions become more detailed and resemble the true character more closely.

Table 5 shows that AR inference significantly improves log-likelihood on both Seen and Unseen test sets. This is expected, as AR inference explicitly models dependencies between target pixels, unlike the standard ConvCNP which assumes independent outputs.

Figure 6 shows outputs from the ConvCNP model with AR inference, using a single sample path. Despite higher log-likelihoods (Table 5), AR ConvCNP produces lower-quality visual outputs than the non-AR model (Figure 5). This is due to a discrepancy between training and testing: the model was trained with up to 197 context points ($N \sim U[3, 197)$), but to create Figure 6 it had to predict all 784 pixels using a growing context set. This represents an out-of-distribution scenario for the model.

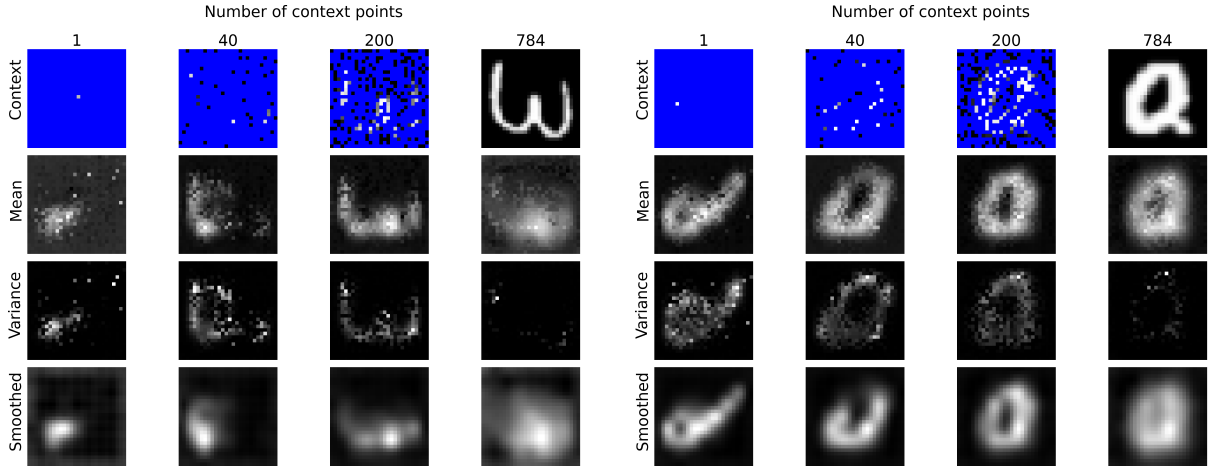

Figure 6: Predicted marginal means, predicted marginal variances, and smoothed means per pixel for the ConvCNP model with autoregressive inference on the Seen (left) and Unseen (right) test sets. Mean and variance are computed from marginal predictions. These reflect per-pixel uncertainty based on random prediction order and a single sample path. That means predictions reflect one possible target sequence, conditioned on the context and previous outputs. Random ordering was used because results depend on prediction order. The smoothed noiseless mean is obtained by passing the AR-generated samples through the model again.

Appendix D of [1] notes that AR models may fail when tested with more context points than seen during training. Although ConvCNPs are more robust due to their convolutional structure, performance still depends on the density of context points. Our results confirm this: the model loses coherence when completing the full image. Appendix D also notes that ordering has little impact during training but becomes important as context density increases. While our figures use random ordering, we found that left-to-right ordering led to worse results (see Appendix 5), further confirming the findings in [1].

The full implementation of the EMNIST image completion task is available on GitHub.

4.2 LBANP with Autoregressive Inference

As an extension, we incorporated the AR inference mechanism into the LBANP, proposed in [11]. Using the paper's original code base [14] we replicate the EQ[1] 1D regression task and extended it to the sawtooth task in [1]. This allowed us to run the LBANP model with AR inference and compare its performance to standard the LBANP model. The code can be found on GitHub.

Table 6 shows that AR inference significantly improves the performance of LBANP in both the Gaussian EQ task and non-Gaussian sawtooth task.

[1] In the LBANP paper [11] the EQ kernel is referred to as the RBF kernel, which is a common alternative name.

	EQ Norm. log-lik. (↑ better)	**Sawtooth** Norm. log-lik. (↑ better)
Ground Truth †	1.89 ± 0.00	—
LBANP †	1.18 ± 0.00	1.24 ± 0.03
AR LBANP †	**1.47 ± 0.01**	**2.70 ±0.04**

Table 6: Normalised log-likelihoods (higher is better) on two synthetically generated 1D processes: GP with an Exponential Quadratic (EQ) kernel and sawtooth. The results compare standard autoregressive inference for LBANP. The † indicates that LBANP results were obtained from slightly different testing sets - parameters for the kernels, as well as the noise distributions, are not exactly the same as those of ConvCNP and ConvGNP in Table 2. The ground truth indicates the normalised log-likelihood of the true target points.

4.3 Inconsistency of Autoregressive Rollout

Under the usage of the AR scheme for inference of more than one target point, the NP is not Kolmogorov consistent. We consider the consistency criteria for marginalisation and permutation.

Permutation Equivariance. While the AR scheme leaves the context set $\mathcal{D}^{(c)}$ permutation invariant, its joint distribution over target points is not permutation equivariant. For instance, the joint prediction of the target variables $(y_1^{(t)}, y_2^{(t)})$ can be calculated in two different orders under the AR scheme.

Consistency under permutation requires that the joint distribution of the target set is invariant to the order of the target points. Specifically, we require:

$$P(y_1^{(t)}, y_2^{(t)} \mid D^{(c)}) = P(y_1^{(t)} \mid D^{(c)})P(y_2^{(t)} \mid x_1^{(t)}, D^{(c)}) = P(y_2^{(t)} \mid D^{(c)})P(y_1^{(t)} \mid x_2^{(t)}, D^{(c)}) \tag{4.3}$$

Enforcing these constraints forces overly restrictive conditions on the model. For example, consider a CNP. Let f be the encoder, g the decoder, and assume aggregation is done by summation. For the CNP to be AR consistent, substituting its formulation into Equation 4.3 yields:

$$g(f(D^{(c)}), x_1^{(t)}) \cdot g(f(D^{(c)} \cup \{x_1^{(t)}\}), x_2^{(t)}) = g(f(D^{(c)}), x_2^{(t)}) \cdot g(f(D^{(c)} \cup \{x_2^{(t)}\}), x_1^{(t)}) \tag{4.4}$$

$$g(f_c, x_1^{(t)}) \cdot g(f_c + f_1, x_2^{(t)}) = g(f_c, x_2^{(t)}) \cdot g(f_c + f_2, x_1^{(t)}) \tag{4.5}$$

where $f_c = f(D^{(c)})$ and $f_i = f(\{x_i^{(t)}\})$. This requirement forces the decoder $g(a, b)$ into a very restrictive form. Meeting this constraint would require oversimplifying the model - for example, using $g(a, b) = a \cdot b$ - which would likely fail to capture complex data patterns.

This equation implies that the joint distribution should be order-invariant. However, due to the structural decisions made in contemporary NP architectures, this consistency does not hold in general.

Marginalisation Inconsistency. Consequently, marginalisation inconsistency arises from an inconsistent definition of the joint. For instance, we would require the assertion of

$$P(y_1^{(t)} \mid D^{(c)}) = \int P(y_1^{(t)}, y_2^{(t)} \mid D^{(c)}) \, \mathrm{d}y_2^{(t)} \tag{4.6}$$

Where in a neural process, the left-hand side of Equation 4.3 would be obtained from a single query at $x_1^{(t)}$, whereas from the previous discussion, the integrand on the right-hand side is not consistently predicted from the AR scheme.

4.3.1 Fitting through samples

Inconsistency of the AR samples is an issue because an AR prediction now depends on an assigned ordering of the target points, which may be arbitrary and hard to reproduce. This order dependence undermines confidence in the resulting intervals when the same inputs yield different outputs. To assess this issue, the authors analysed the standard deviation of the log-likelihood. They observed that with low context sizes, the standard deviation can reach up to 0.5 (see Figure 7), corresponding to a multiplicative factor of about 1.65 in the likelihood space. This high variance is expected since later samples are more influenced by earlier, less stable predictions when the function is not well constrained.

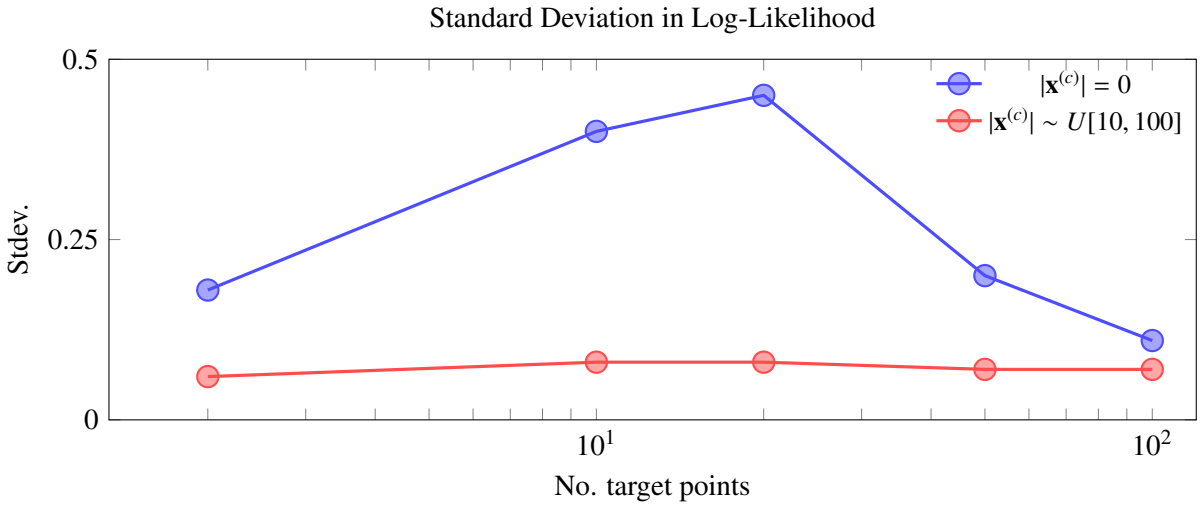

Figure 7: Standard deviation in log-likelihood as a function of the number of target points for different context sizes for 1-Dimensional Sawtooth Data, obtained from Figure 9 of [1].

For larger context sizes, the deviations between joint distributions become much smaller, but this fails to capture the complete picture. When the context is large, the advantage of using an autoregressive scheme over marginal predictions diminishes. As shown in Table 7 and Figure 8, the AR scheme offers significant gains in log-likelihood only for small context sizes; gains become marginal once the context exceeds about 25 points. This is concerning because AR is most beneficial where it also shows the greatest inconsistency.

Context Size	Average Difference (AR superiority)	Sample Count
1-5	4.3348	251
6-10	1.4958	557
11-15	0.5497	953
16-20	0.2489	1114
21-25	0.1202	1513
26-30	0.0603	2015
31-35	0.0316	2226
36-40	0.0144	2455
41-45	0.0068	3018
46-50	0.0022	3542

Table 7: Average difference between autoregressive and marginal log-likelihoods by context size intervals, from LBANP model on Gaussian Process regression.

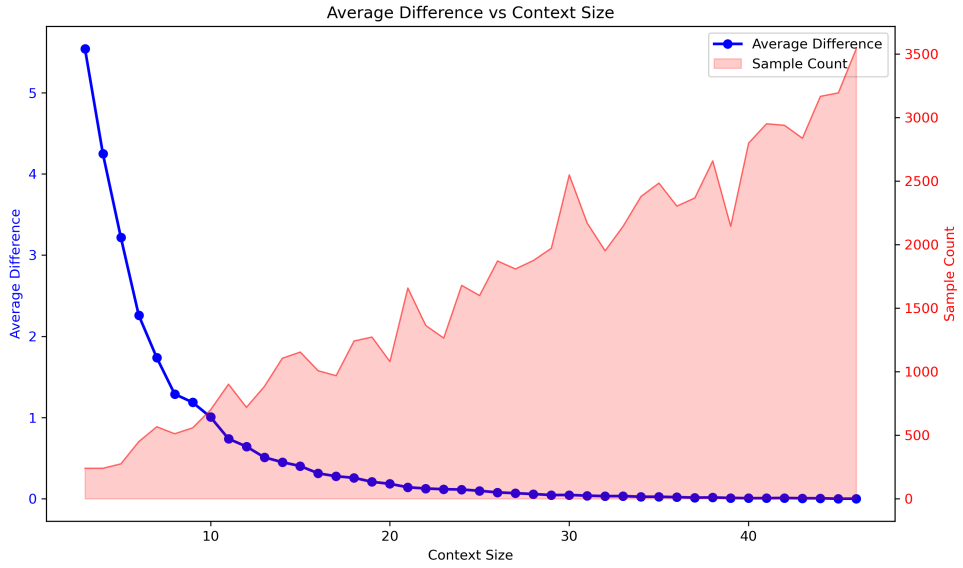

Figure 8: Average difference between autoregressive and marginal log-likelihood by context size intervals in synthetic Gaussian Process regression using LBANP model.

4.3.2 Alternative Quantification of Inconsistency

Furthermore, the metric of standard deviation in log-likelihood space is rather abstract and difficult to visualise. We propose a different way of quantifying this inconsistency by attempting to estimate the joint distribution of the target set from samples independent of their ordering. We do this by first sampling a random ordering of the target points, using this ordering to generate a sample of the joint distribution, then repeating for many samples. For data and target points, it is infeasible to obtain a good estimate of the joint distribution due to the curse of dimensionality. We therefore assume the joint distribution is multivariate Gaussian and proceed by fitting the mean and covariance matrix from the drawn samples. In the case of GP simulated datasets, this assumption indeed represents the ground truth - whether the joint distributions outputted from the AR scheme are Gaussian is another matter entirely.

	EQ Norm. log-lik. (↑ better)
Ground Truth	-0.25 ± 0.00
ConvGNP	**-0.25 ± 0.00**
ConvCNP	-0.65 ± 0.01
AR ConvCNP	**-0.25 ± 0.00**
AR ConvCNP - alt.	-0.64 ± 0.04

	EQ Norm. log-lik. (↑ better)
Ground Truth †	1.89 ± 0.00
LBANP †	1.18 ± 0.00
AR LBANP†	**1.47 ± 0.01**
AR LBANP - alt. †	1.21 ± 0.02

Table 8: Normalised log-likelihoods (higher is better) on three synthetically generated 1D GP with an Exponential Quadratic (EQ) kernel. The results compare standard and autoregressive inference for ConvGNP, ConvCNP and LBANP, as well as alternative autoregressive inference to investigate the inconsistent nature of the autoregressive procedure. The † indicates that LBANP results were obtained from slightly different testing sets - parameters for the kernels, as well as the noise distributions, are not exactly the same as that of ConvCNP and ConvGNP. The ground truth indicates the normalised log-likelihood of the true target points.

	Int. (S)	For. (S)	Rec. (S)	Int. (R)	For. (R)	Rec. (R)
ConvGNP	-3.47 ± 0.02	-4.17 ± 0.02	-3.68 ± 0.02	-4.17 ± 0.04	-4.57 ± 0.01	-5.08 ± 0.16
ConvCNP	-3.47 ± 0.02	-4.85 ± 0.02	-4.06 ± 0.02	-4.21 ± 0.04	-4.96 ± 0.01	-4.75 ± 0.06
AR ConvCNP	**-3.29 ± 0.02**	**-3.58 ± 0.02**	**-3.45 ± 0.02**	**-4.16 ± 0.04**	**-4.32 ± 0.01**	**-4.35 ± 0.04**
AR ConvCNP - alt.	-3.54 ± 0.03	-5.05 ± 0.02	-4.14 ± 0.03	-4.25 ± 0.05	-4.83 ± 0.02	-4.85 ± 0.07

Table 9: Normalised log-likelihoods (higher is better) across six Sim-to-Real Lotka-Volterra tasks. Tasks include interpolation (Int.), forward extrapolation (For.), and reconstruction (Rec.), evaluated on both simulated (S) and real (R) environments. The results compare standard and autoregressive inference for ConvCNP and ConvGNP as well as alternative autoregressive inference to investigate the inconsistent nature of the autoregressive procedure.

If the joint distributions from different orderings are similar and nearly Gaussian, then approximating their mixture with a single Gaussian should fit well. We can evaluate this fit by using the resulting Gaussian to compute the log-likelihood of the test data - especially for GP regression, where the ground truth is a multivariate Gaussian. The corresponding results are shown in Tables 8 and 9.

The performance of force-fitting a Gaussian joint distribution yields predictions that are only marginally better or worse than independent predictions - and far worse than those from the AR scheme. This implies that the joint distributions are either highly non-Gaussian, significantly inconsistent, or both. In the next section, we discuss how to address this consistency issue.

4.4 Addressing Inconsistency

4.4.1 Mixture-Based Formulation for Autoregressive Neural Processes

To ensure theoretical consistency, we propose abandoning the notion that the AR scheme yields deterministic outputs. Instead, we formalise a stochastic interpretation by modelling the output as a sample from a mixture distribution. Through the principle of permutation equivariance, no particular ordering of target points should be privileged; hence, we postulate a mixture model wherein for a set of N target points $\{x_1^{(t)}, x_2^{(t)}, \ldots, x_N^{(t)}\}$, the likelihood derived from the product rule in Equation (2.5) is drawn from the distribution:

$$q^{(AR)}\left(y_1^{(t)}, y_2^{(t)}, \ldots, y_N^{(t)} \mid D^{(c)}, x_{1:N}^{(t)}\right) = \frac{1}{|S_N|} \sum_{\pi_k \in S_N} q^{(AR)}\left(y_{\pi_k(1)}^{(t)}, y_{\pi_k(2)}^{(t)}, \ldots, y_{\pi_k(N)}^{(t)} \mid D^{(c)}, x_{1:N}^{(t)}\right) \tag{4.7}$$

where $q^{(AR)}\left(y_{\pi_k(1)}^{(t)}, y_{\pi_k(2)}^{(t)}, \ldots, y_{\pi_k(N)}^{(t)} \mid D_c, x_{1:N}^{(t)}\right)$ represents the likelihood estimate from the NP under the AR scheme computed via the product rule in Equation (2.1) and S_N denotes the set of all permutations of the target points. It is noteworthy that all permutations are ascribed equal probability for the latent variable K. This formulation ensures that the joint distribution is consistent under both marginalisation and permutation.

Under this interpretation, the conventional AR scheme is equivalent to a single-sample Monte Carlo estimate of the mixture model - it computes the likelihood using one permutation of target points. While evaluating all permutations is infeasible, using multiple samples and averaging can reduce the variance of the likelihood estimate, in line with the law of large numbers. Empirical results for this behaviour are exhibited in Appendix for the LBANP in both GP and Sawtooth regression tasks.

A key assumption here is that the ordering of target points is independent and, ideally, uniformly distributed. However, by relaxing the joint distribution assumption to use a mixture model, we also

allow for non-uniform distributions across permutations, adding modelling flexibility. Consequently, our AR inference procedure is as follows:

1. Establish a fixed, reproducible ordering of the target points (e.g., sorted by L_2 norm). Denote this ordering as $(x_1^{(t)}, x_2^{(t)}, \ldots, x_N^{(t)})$.
2. Sample a permutation π_k from the discrete distribution $K = \{1, \ldots, N!\}$ spanning all permutations of the target points.
3. Generate predictions for $y_{\pi_k(1)}^{(t)}, y_{\pi_k(2)}^{(t)}, \ldots, y_{\pi_k(N)}^{(t)}$ from the neural process under the AR scheme.

In general, the distribution of K can be learned using latent variable techniques such as Expectation Maximisation, Variational Inference, or gradient descent with a pointer-like loss function (i.e., a classification loss where targets are trained to identify their predecessor in the permutation). However, these methods are beyond the scope of this study and will be explored in future research.

4.4.2 Heuristic Ordering Strategies

Here we explore deterministic distributions for K - specifically, cases where the ordering is non-stochastic. These approaches are derived from heuristic principles, and we demonstrate empirically that in certain scenarios, they **outperform** the standard AR scheme.

In Section D.3 of their appendix [1], the authors argued that an effective ordering should faithfully mirror the evolving conditional distributions of the NP throughout the AR sampling procedure. The authors noted that points sampled early in the sequence often have poor distributional characteristics - reflecting a blend of epistemic and aleatoric uncertainty - which leads to deviations from Gaussian behaviour. As the effective context grows, the underlying function becomes more accurately constrained, with the uncertainty shifting mainly to its aleatoric component. However, the way in which the function is constrained can be flawed, as it may include errors that propagate from earlier sampling stages. Thus, selecting the initial samples carefully is essential for effectively guiding the joint distribution. Motivated by these considerations, we propose the following heuristic ordering strategies:

1. **Greedy Minimal Uncertainty (GMinU)**. An intuitive approach to ordering target points involves prioritising those for which we possess the highest predictive confidence. By constructing the effective context set with reliable samples, we enhance the AR scheme's performance for subsequent predictions with enlarged effective context. This concept is illustrated in Figure 9. We designate this as the *Greedy Minimal Uncertainty* (GMinU) ordering. Model-based variance estimators or simple Euclidean distances of target points $x_i^{(t)}$ can serve as proxies for predictive uncertainty.
2. **Greedy Maximal Uncertainty (GMaxU)**. This represents the inverse of the GMinU ordering, wherein we initially sample from points associated with minimal predictive confidence. The rationale becomes apparent when considering the exemplar provided in Figure 10. We term this the *Greedy Maximal Uncertainty* (GMaxU) ordering.

Table 10 shows the empirical performance of our heuristic ordering strategies for autoregressive inference using the LBANP model on both EQ and Sawtooth regression tasks, revealing that performance varies with the task and ordering strategy.

For EQ GP regression - wherein the ground truth conditional distributions are Gaussian - we observe that the GMaxU ordering strategy yields statistically significant improvements relative to both the random (canonical AR) ordering and the GMinU approach. Notably, the GMinU strategy exhibits inferior performance compared to the random ordering baseline, suggesting that the LBANP model demonstrates excessive confidence in its predictions for target points proximal to the original context

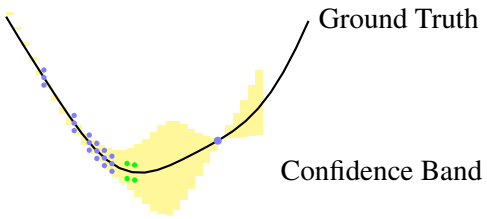

Figure 9: A scenario when GMinU is suitable. By first sampling from the green area where the model has high confidence, we build up a reliable context set. This allows the model to make better predictions in the more uncertain middle region.

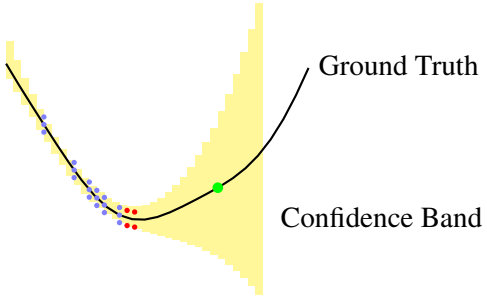

Figure 10: A scenario when GMaxU may be suitable. By first sampling from the green area, the model has less certainty in this area and is more likely to be able to capture the large-scale trend. If we were to sample at the red points first, the model becomes overconfident in its beliefs of a continuing downward trend from an augmented context set, leading to very bad losses in the green areas.

set. This phenomenon is particularly pronounced when the underlying aleatoric noise deviates from Gaussian structure. For predictions in closer proximity to context labels, deviations are predominantly aleatoric; consequently, by imposing conditional Gaussian marginals that fail to capture the true nature of the aleatoric noise, the neural process demonstrates significantly degraded performance. This is in fact the case for the GP task for LBANP - noise variables are drawn from Student-t distributions, with significantly larger tails than Gaussians. This overconfidence propagates sequentially through the sampling procedure and compounds errors cumulatively, reducing predictive accuracy.

Method	**EQ** Norm. log-lik. (↑ better)	**Sawtooth** Norm. log-lik. (↑ better)
Random (Original AR)	1.467 ± 0.006	2.705 ± 0.041
GMaxU	**1.478 ± 0.005**	2.692 ± 0.036
GMinU	1.465 ± 0.005	**2.773 ± 0.032**

Table 10: Normalised log-likelihood measurements (higher is better) for different ordering strategies on GP and Sawtooth regression tasks using LBANP with autoregressive inference. Results are presented with standard error estimates derived from multiple experimental runs.

For the Sawtooth regression task, with its non-Gaussian conditional distributions, an opposite performance trend emerges. Specifically, the GMaxU strategy underperforms relative to the random ordering baseline, while the GMinU strategy achieves statistically significant improvements. This can be explained by noting that the greatest uncertainty in the Sawtooth function is consistently concentrated near its discontinuities. Yet, sampling early from these uncertain regions provides little insight into the function's behaviour in other areas. Thus, by deferring sampling in high-uncertainty regions, the GMinU strategy enables a more robust overall characterisation of the function before addressing the challenging

discontinuities.

These results indicate that neither GMaxU nor GMinU is universally superior. Both rely on the variance structure of target points, but GMaxU samples high-variance regions early while GMinU defers them, leading to task-specific performance differences that can be leveraged with appropriate ordering strategy selection using prior knowledge of the target domain's distribution. We explore these methods in more depth in Appendix 5.

Conclusion

Our investigation into AR inference for NPs has revealed its strategic advantages in modelling complex dependencies and capturing non-Gaussian predictive distributions. Our replication experiments on synthetic regression tasks, sim-to-real predator-prey dynamics, and image completion on EMNIST demonstrated that introducing an AR test-time procedure improves log-likelihoods compared to standard factorised models. Furthermore, our extension of AR inference to LBANP models confirms its broad applicability across different NP architectures. We finally introduce and address AR inconsistency by proposing a mixture-based formulation that averages over all target orderings for permutation invariance and by introducing heuristic ordering strategies (GMinU and GMaxU) that dynamically prioritise target points based on predictive uncertainty to improve performance.

Potential directions for future work include tuning the regularisation weight λ for EMNIST image completion to balance the trade-off between limiting the maximum log-likelihood and preventing the model from gaming the metric. Regularisation strategies for the ConvGNP, such as trace or log-determinant terms, could be explored for comparison with ConvCNP and AR ConvCNP. Lastly, Expectation Maximisation, Variational Inference or pointer-style loss functions could be investigated to learn the distribution over permutations when addressing the AR inconsistency.

Bibliography

[1] W. P. Bruinsma, S. Markou, J. Requiema, A. Y. Foong, T. R. Andersson, A. Vaughan, A. Buonomo, J. S. Hosking, and R. E. Turner, "Autoregressive conditional neural processes," *arXiv preprint arXiv:2303.14468*, 2023.

[2] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in *2017 international joint conference on neural networks (IJCNN)*, pp. 2921–2926, IEEE, 2017.

[3] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. Eslami, and Y. W. Teh, "Neural processes," *arXiv preprint arXiv:1807.01622*, 2018.

[4] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer school on machine learning*, pp. 63–71, Springer, 2003.

[5] M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende, and S. A. Eslami, "Conditional neural processes," in *International conference on machine learning*, pp. 1704–1713, PMLR, 2018.

[6] Y. Dubois, J. Gordon, and A. Y. Foong, "Neural process family." http://yanndubs.github.io/Neural-Process-Family/, September 2020.

[7] J. Gordon, W. P. Bruinsma, A. Y. K. Foong, J. Requeima, Y. Dubois, and R. E. Turner, "Convolutional conditional neural processes," 2020.

[8] W. P. Bruinsma, J. Requeima, A. Y. Foong, J. Gordon, and R. E. Turner, "The gaussian neural process," *arXiv preprint arXiv:2101.03606*, 2021.

[9] S. Markou, J. Requeima, W. P. Bruinsma, A. Vaughan, and R. E. Turner, "Practical conditional neural processes via tractable dependent predictions," *arXiv preprint arXiv:2203.08775*, 2022.

[10] T. Nguyen and A. Grover, "Transformer neural processes: Uncertainty-aware meta learning via sequence modeling," *arXiv preprint arXiv:2207.04179*, 2022.

[11] L. Feng, H. Hajimirsadeghi, Y. Bengio, and M. O. Ahmed, "Latent bottlenecked attentive neural processes," *arXiv preprint arXiv:2211.08458*, 2022.

[12] W. Bruinsma, "Neural processes: A framework for composing neural processes in python," 2021. Accessed: 2025-03-27.

[13] D. A. MacLulich, *Fluctuations in the numbers of the varying hare (Lepus americanus)*, vol. 189. University of Toronto Press Toronto, 1937.

[14] L. Feng, H. Hajimirsadeghi, Y. Bengio, and M. O. Ahmed, "Latent bottlenecked attentive neural processes," 2023. Accessed: 2025-03-27.

Appendix

EMNIST image completion task

For the EMNIST image completion task, we applied variance regularisation only to the ConvCNP model. This is because ConvCNP uses a diagonal Gaussian covariance, which allows a simple regularisation term to penalise small predicted variances. In contrast, the ConvGNP model uses a full Gaussian covariance structure [8], which makes independent variance regularisation infeasible. Applying regularisation in this case would require more complex terms, such as the trace or log-determinant of the covariance matrix, which we leave for future work.

As shown in Figure 11, ConvCNP exhibits similar behaviour to ConvCNP when completing full images without regularisation. This suggests that the model, like ConvCNP, is also prone to exploiting the log-likelihood objective when not properly constrained.

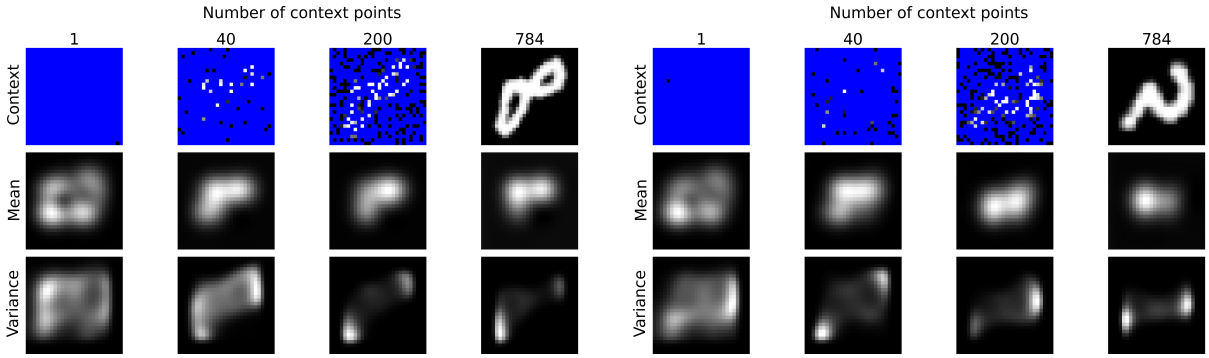

Figure 11: EMNIST image completion using ConvGNP without regularisation and without autoregressive inference. Left: digit from the Seen set (0-9). Right: letter from the Unseen set (10-46).

Effect of ordering

Appendix D of [1] suggests that ordering of the context points has minimal effect during training but becomes more important when context density increases. Our experiments support this: while we used random context ordering for most of our results, we also tested a left-to-right ordering scheme. As shown in Figure 12, this ordering led to worse performance and noisier predictions. These results confirm the findings of [1] and highlight the sensitivity of autoregressive inference to context ordering.

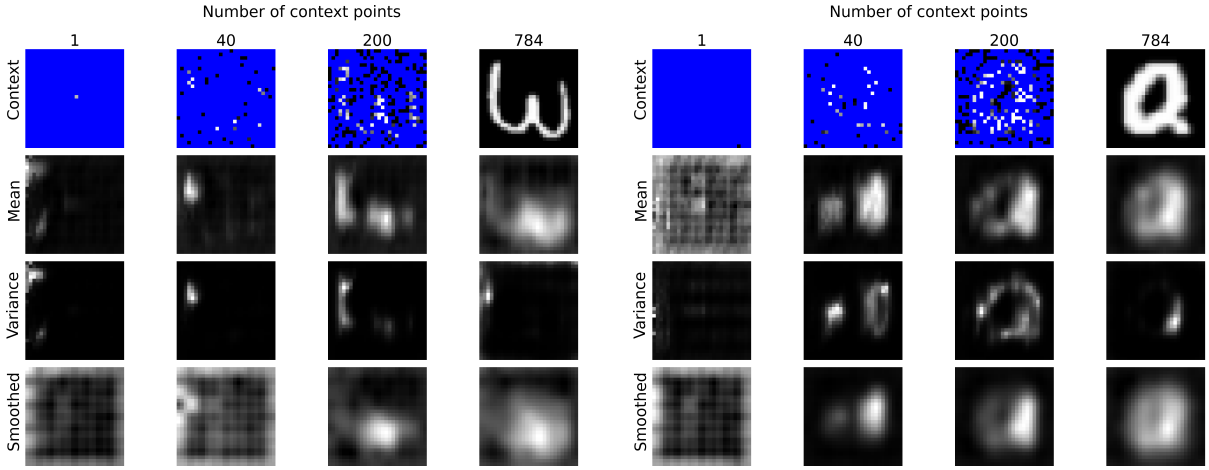

Figure 12: Predicted mean, variance, and smoothed mean for the ConvCNP model with autoregressive inference on the Seen (left) and Unseen (right) test sets, with predictions made left to right. Mean and variance are computed from marginal predictions. These reflect per-pixel uncertainty, based on a single sample path. The smoothed noiseless mean is obtained by passing the AR-generated samples through the model again.

Batched Autoregressive Inference for LBANP

To expedite auto-regressive (AR) inference, one may concurrently process multiple target points during the inference phase. Whilst this approach forgoes the modelling of dependencies amongst target points within the same batch, it significantly accelerates inference by reducing the requisite number of forward passes through the computational model.

The objective of the Latent Bottleneck Attentive Neural Process (LBANP) was to reduce both training and inference complexity to scale linearly with respect to the cardinality of context and target point sets. However, employing auto-regressive inference rollout methodologies increases the computational complexity during inference to quadratic scaling in both aforementioned dimensions. Therefore, in accordance with the theoretical underpinnings of LBANP, it becomes mathematically prudent to implement batched inference strategies that preserve linear complexity scaling. This may be accomplished by constraining the number of auto-regressive rollouts to a predetermined constant for each instance, effectively partitioning the target point set into a fixed number of batches.

A straightforward implementation would involve the uniform random distribution of target points amongst these predefined groups. Empirical evaluations of this methodology applied to LBANP on synthetic EQ Gaussian Process regression and Sawtooth function approximation tasks are presented in Table 11. The results demonstrate that performance enhancements achieved through auto-regressive rollouts with as few as 2 batches already exhibit statistically significant improvements compared to the non-auto-regressive baseline (i.e., batches = 1).

The results presented in Table 11 indicate that even minimal application of auto-regressive batching methodologies yields statistically significant performance enhancements vis-à-vis non-auto-regressive computational paradigms. This behaviour suggests the existence of an optimal equilibrium point k^* that effectuates a balance between computational efficiency and predictive performance tailored for a specific task.

AR Batches	EQ	Sawtooth
1 (No AR)	1.183 ± 0.005	1.238 ± 0.022
2	1.313 ± 0.004	1.892 ± 0.027
4	1.426 ± 0.003	2.039 ± 0.025
8	1.444 ± 0.004	2.555 ± 0.039
N (Full AR)	1.467 ± 0.004	2.705 ± 0.041

Table 11: Performance comparison of LBANP with different numbers of autoregressive batches. Values shown are average negative log-likelihood (lower is better) on test sets. The inference time is shown relative to the non-autoregressive case. N represents the number of target points, which varies per test instance. If the number of batches is larger than the number of target points, then we just set the number of batches to be the number of target points.

Smart Batching

The stochastic methodology employed for batch construction need not be constrained to uniform. From a statistical inference perspective, target points within an identical batch are treated as conditionally independent by the neural process, suggesting an optimal partitioning strategy wherein points within each batch exhibit minimal statistical dependency. For instance, in the context of Gaussian Process regression tasks, the covariance function $k(x, x')$ induces a natural metric whereby spatial proximity correlates with statistical dependency, with $\lim_{\|x-x'\|_2 \to \infty} k(x, x') \to 0$ for many common kernels. Consequently, a theoretically justified partitioning scheme would seek to minimise the mutual information $I(X_i; X_j)$ between random variables corresponding to points x_i, x_j within the same batch, or equivalently, to minimise the within-batch entropy $H(X_B) = H(X_1, X_2, \ldots, X_{|B|})$ for each batch B. Such methods need additional $O(1)$ computational complexity per datapoint, but still must remain algorithmically efficient to avoid becoming the dominant computational bottleneck in the inference pipeline.

One instantiation of an information-theoretic greedy algorithm would involve the partitioning of target points into similarity-based clusters $C = \{C_1, C_2, \ldots, C_k\}$ (constructed via k-means clustering with respect to an appropriate metric space) followed by the construction of batches $\mathcal{B} = \{B_1, B_2, \ldots, B_m\}$ through strategic sampling across these clusters. Specifically, each batch B_j is populated by selecting points from disparate clusters to maximise the pairwise dissimilarity metric $d(x_i, x_{i'}) = \|x_i - x_{i'}\|_2$ for all $(x_i, x_{i'}) \in B_j \times B_j$. Empirical evaluation is applied to EQ Gaussian Process regression tasks is presented in Table 12.

AR Batches	Clustered	EQ
1 (No AR)	✗	1.183 ± 0.005
2	✗	1.313 ± 0.004
2	✓	**1.329 ± 0.003**
4	✗	1.426 ± 0.003
4	✓	**1.438 ± 0.004**
8	✗	1.444 ± 0.004
8	✓	**1.452 ± 0.004**
N (Full AR)	✗	1.467 ± 0.004

Table 12: Performance comparison of LBANP with different numbers of autoregressive batches, comparing random versus clustered batching strategies. Values shown are average negative log-likelihood (lower is better) on test sets. Clustered batching consistently outperforms random batching by grouping dissimilar points together in each batch.

For alternative tasks exhibiting distinct dependency structures, such as the sawtooth function where correlation patterns manifest non-stationary periodicities, one must employ task-specific dissimilarity metrics that appropriately characterise the underlying statistical relationships.

MC sampling for Mixture Models

One may employ larger sample sizes to estimate the joint distribution across the target set. Table 13 presents LBANP performance with varying Monte Carlo (MC) sample quantities, alongside empirical variance estimates of these approximations. For instances with sufficiently small target sets, exhaustive permutation evaluation becomes computationally feasible, enabling exact joint distribution estimation under the mixture model framework. Comparative results between exact calculation and MC approximation methods are presented in Table 13.

No. MC Samples	EQ	Sawtooth	Variance (EQ)	Variance (Sawtooth)
1 (Original AR)	1.467 ± 0.004	2.705 ± 0.041	0.072	0.212
2	1.462 ± 0.003	2.615 ± 0.039	0.032	0.136
5	1.472 ± 0.003	2.695 ± 0.033	0.014	0.045
10	1.489 ± 0.002	2.725 ± 0.034	0.009	0.028

Table 13: Performance comparison of LBANP with different numbers of Monte Carlo samples for estimating the joint distribution. Values shown are average log-likelihood (higher is better) on test sets.

Monte Carlo (MC) estimators cannot be expected to surpass single-sample auto-regressive schemes in terms of predictive accuracy, as both methodologies approximate the same underlying joint distribution. The empirical variance of MC estimators exhibits an inverse proportionality relationship with sample cardinality, decreasing in a manner consistent with theoretical expectations - specifically, following a linear decay pattern in accordance with the central limit theorem and the law of large numbers as the sample size increases.

Method	< 5 Target Points	Variance
Exact (All Permutations)	1.584 ± 0.000	0.000
MC (10 samples)	1.592 ± 0.013	0.040
MC (5 samples)	1.584 ± 0.009	0.091
MC (2 samples)	1.573 ± 0.011	0.156
Original AR (1 sample)	1.568 ± 0.018	0.222

Table 14: Comparison between exact permutation-based estimation and Monte Carlo sampling for small numbers of target points. For < 5 target points, we can compute the exact mixture by evaluating all possible permutations.

GMaxU and GMinU Heuristic Ordering

We propose two heuristic ordering strategies, GMaxU and GMinU, which enhance AR scheme performance by strategically evaluating points of higher uncertainty either earlier or later in the sequence. This establishes a joint distribution wherein the permutation of target points is non-uniformly distributed, preferentially favouring specific orderings.

In our experimentation, we employ deterministic orderings that ensure consistency of the augmented AR scheme without necessitating Monte Carlo evaluations. This consistency is achieved in the trivial

sense, as any initial permutation of target points will be transformed to a fixed ordering via the respective GMaxU or GMinU methodology. We investigate two uncertainty notions, the selection of which has been primarily motivated by computational constraints.

Autoregressive Variance The first notion of uncertainty utilises the trained model's autoregressive variance statistics. During the kth step of the AR scheme, with augmented context set $D_c^{(k)} = D_c \cup \{(x_{t,1}, y_{t,1}), \ldots, (x_{t,k-1}, y_{t,k-1})\}$, we compute the marginal variances of the non-evaluated target points $y_{t,n}$ for $n = k, k+1, \ldots, N$ given $D_c^{(k)}$. We then select the subsequent target rollout point as that exhibiting the highest/lowest variance for GMaxU/GMinU respectively. A significant limitation of this approach is its $O(N)$ computational complexity per step, resulting in $O(N^2)$ complexity for the entire rollout, rendering it computationally prohibitive for large N.

To ameliorate this issue, we approximate autoregressive variance statistics with those calculated at step $k = 1$, thereby establishing a fixed ordering of all target points that persists throughout. This methodology necessitates the ordering calculation only once per datapoint, thus decoupling computational complexity from target set size. We empirically evaluate both approaches in table 15.

Method	**EQ** ↑	**Sawtooth** ↑
Autoregressive Variance at Each Step		
GMaxU	**1.478 ± 0.005**	2.692 ± 0.036
GMinU	1.465 ± 0.005	**2.743 ± 0.032**
Fixed Ordering Based on Initial Marginal Variance		
GMaxU	1.472 ± 0.006	2.685 ± 0.038
GMinU	1.460 ± 0.006	2.735 ± 0.034
Random (Original AR)	1.467 ± 0.006	2.705 ± 0.041

Table 15: Comparison of GMaxU and GMinU heuristic ordering strategies using both recalculated uncertainty at each step and fixed ordering based on initial uncertainty.

Euclidean Distance Uncertainty The second notion of uncertainty employs the Euclidean distance between target points as a proxy for the dependence of corresponding target labels. In practice, any valid kernel function may be substituted for the L_2 norm (indeed, for Gaussian Processes, the L_2 norm is equivalent by monotonicity of the kernel function). Herein, uncertainty is quantified as the minimum Euclidean distance to any other point within the target set.

We investigate both autoregressive distance uncertainty, which entails recalculating the minimum distance at each iterative step of the AR scheme, and a fixed ordering determined at step $k = 1$ based solely on minimum distance to context points. The comparative empirical results are presented in table 16. The empirical evidence suggests that for the Gaussian Process and Sawtooth tasks, autoregressive variance-based ordering yields optimal performance. However, this methodology incurs substantial computational overhead, rendering alternative approaches potentially more suitable for large-scale target sets whilst still demonstrating statistically significant improvements over stochastic ordering methodologies.

Method	**EQ ↑**	**Sawtooth ↑**
Autoregressive Distance at Each Step		
GMaxU	**1.475 ± 0.006**	2.688 ± 0.037
GMinU	1.462 ± 0.006	**2.720 ± 0.033**
Fixed Ordering Based on Initial Distance to Context		
GMaxU	1.470 ± 0.007	2.680 ± 0.039
GMinU	1.458 ± 0.007	2.710 ± 0.035
Random (Original AR)	1.467 ± 0.006	2.705 ± 0.041

Table 16: Comparison of GMaxU and GMinU heuristic ordering strategies using Euclidean distance as uncertainty proxy, with both recalculated distances at each step and fixed ordering based on initial distances to context points.