
Test Time Scaling for Neural Processes

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 Uncertainty-aware meta-learning aims not only for rapid adaptation to new tasks
2 but also for reliable uncertainty estimation under limited supervision. Neural Pro-
3 cesses (NPs) offer a flexible solution by learning implicit stochastic processes
4 directly from data, often using a global latent variable to capture functional un-
5 certainty. However, we empirically find that variational posteriors for this global
6 latent variable are frequently miscalibrated, limiting both predictive accuracy and
7 the reliability of uncertainty estimates. To address this issue, we propose Test
8 Time Scaling for Neural Processes (TTSNPs), a sequential inference framework
9 based on Sequential Monte Carlo Sampler (SMCS) that refines latent samples at
10 test time without modifying the pre-trained NP model. TTSNPs iteratively trans-
11 form variational samples into better approximations of the true posterior using
12 neural transition kernels, significantly improving both prediction quality and un-
13 certainty calibration. This makes NPs more robust and trustworthy, extending
14 applicability to various scenarios requiring well-calibrated uncertainty estimates.

15 1 Introduction

16 In many real-world scenarios, tasks naturally vary in several aspects, such as the number of avail-
17 able data points and the degree of correlation between training and evaluation data. Despite
18 the diversity, a common challenge remains: each task typically offers only limited supervision,
19 which makes it difficult to train models that generalize reliably within individual tasks [2, 25].
20 To address this issue, meta-learning [48, 14] has emerged as a powerful paradigm. While meta-
21 learning [14, 23, 21] has demonstrated significant success in improving adaptability in various fields
22 via extracting transferable knowledge, reliable uncertainty estimation along with accurate predic-
23 tions becomes important, as it allows models to express appropriate levels of confidence in their pre-
24 dictions and make trustworthy decisions given limited information. To this end, uncertainty-aware
25 meta-learning [46, 1, 31, 47] have been proposed to integrate uncertainty quantification directly into
26 meta-training and evaluation processes, promoting both rapid adaptation and calibrated prediction.

27 Among various frameworks, Neural Processes [NPs; 17, 18] have emerged as a flexible and data-
28 driven approach for uncertainty-aware meta-learning. By training parametric neural networks to
29 maximize predictive likelihoods across meta-task datasets, NPs can capture the data-generating
30 mechanisms underlying diverse tasks without relying on explicit prior assumptions. As a result,
31 they offer a powerful balance between flexibility and uncertainty estimation, positioning them as a
32 foundational model for uncertainty-aware meta-learning [31]. A common approach to modeling un-
33 certainty in NPs involves introducing a global latent variable, which encodes functional uncertainty
34 or epistemic uncertainty across tasks.

35 However, in this work, we empirically demonstrate that previous approaches on NP mod-
36 els [18, 26, 31] leveraging a global latent variable to model functional uncertainty often suffer from
37 miscalibrated variational posteriors for the global latent variable. Moreover, we find that such mis-
38 calibration issues consistently arise regardless of the specific loss functions or model structure used

during the training of NPs. This miscalibration presents a fundamental challenge for NPs, where the global latent variable serves as the core mechanism for conveying contextual information to target predictions. When the variational posterior fails to faithfully reflect the uncertainty in the context set, the resulting latent representations may become either overly deterministic or excessively diffuse. Consequently, the model’s predictions on target points can suffer from poor calibration and reduced expressiveness, particularly when the target inputs deviate from those observed in the context set. These limitations undermine the generalization ability and reliability of NPs, especially in scenarios where accurate uncertainty estimation is crucial.

To overcome this limitation, we propose a principled sequential inference framework based on Sequential Monte Carlo Sampler (SMCS) [10, 9], designed to refine latent variable samples at test time without modifying the pre-trained NP model. Specifically, our approach, Test Time Scaling for Neural Processes (TTSNPs), iteratively transforms latent samples from the variational posterior into more accurate approximations of the true posterior using SMCS, with following two key components: (1) learned neural network transition kernels, and (2) learned intermediate distributions, achieved by generating pseudo context points via a neural network. This refinement improves predictive performance and enhances uncertainty calibration, enabling NPs to produce more reliable and trustworthy predictions. As a result, TTSNPs significantly extend the applicability of NPs to various tasks that demand robust and well-calibrated uncertainty estimation.

2 Preliminaries

2.1 Problem Setup

In typical NP settings, the dimensionality of datasets is assumed to be fixed [17, 18, 26, 30]. In contrast, following the direction explored in Lee et al. [31], we consider a more general setting in which the data for each task may vary in dimensionality. Specifically, we assume potentially infinite collection of tasks $\{\tau_j\}_{j \in \mathbb{N}}$, each drawn i.i.d. from a task distribution $p_\tau(\tau)$. Every task τ_j uses an input space $\mathcal{X}_{d_x(j)} \subseteq \mathbb{R}^{d_x(j)}$ and output space $\mathcal{Y}_{d_y(j)} \subseteq \mathbb{R}^{d_y(j)}$, where $d_x, d_y: \mathbb{N} \rightarrow \mathbb{N}$ specify each task’s input-output dimensionality. Concretely, τ_j provides a dataset $\mathcal{D}_j = \{\mathbf{d}_{j,k}\}_{k=1}^{n_j}$ in which each $\mathbf{d}_{j,k} = (\mathbf{x}_{j,k}, \mathbf{y}_{j,k})$ belongs to $\mathcal{X}_{d_x(j)} \times \mathcal{Y}_{d_y(j)}$. A subset $c_j \subsetneq [n_j]$ of indices defines the *context set* $\mathcal{D}_{j,c} := \{\mathbf{d}_{j,k}\}_{k \in c_j}$, while the complement $t_j = [n_j] \setminus c_j$ gives the *target set* $\mathcal{D}_{j,t}$. Each dataset \mathcal{D}_j is considered i.i.d. from some unknown function $f_j: \mathcal{X}_{d_x(j)} \rightarrow \mathcal{Y}_{d_y(j)}$. Meta-learning aims to learn a *shared* representation capturing how $\mathbf{x}_{j,k}$ and $\mathbf{y}_{j,k}$ are related, using $\mathcal{D}_{j,c}$ (training) and $\mathcal{D}_{j,t}$ (validation) across many tasks.

2.2 Neural Processes

NPs address the meta-learning problem by defining a distribution over functions f_j conditioned on each context set $\mathcal{D}_{j,c}$. The predictive distribution can be viewed as follows [31]:

$$p(\mathbf{Y}_j | \mathbf{X}_j, \mathcal{D}_{j,c}) = \int \left[\prod_{k \in [n_j]} p(\mathbf{y}_{j,k} | f_j, \mathbf{x}_{j,k}) \right] p(f_j | \mathcal{D}_{j,c}) df_j, \quad (1)$$

where \mathbf{X}_j and \mathbf{Y}_j collect all inputs $\{\mathbf{x}_{j,k}\}$ and outputs $\{\mathbf{y}_{j,k}\}$. When adopting a Gaussian likelihood for $p(\mathbf{y}|f, \mathbf{x})$ and introducing a latent variable $\mathbf{r}_j \in \mathbb{R}^{d_r}$ that parameterizes f_j , one might write

$$p(\mathbf{Y}_j | \mathbf{X}_j, \mathcal{D}_{j,c}) = \int \prod_{k \in [n_j]} \mathcal{N}(\mathbf{y}_{j,k} | \mu_{\mathbf{r}_j}(\mathbf{x}_{j,k}), \text{diag}(\sigma_{\mathbf{r}_j}^2(\mathbf{x}_{j,k}))) q(\mathbf{r}_j | \mathcal{D}_{j,c}; \phi) d\mathbf{r}_j, \quad (2)$$

where ϕ is the parameter set of the NPs encoder module. NP variants differ primarily in how they encode $\mathcal{D}_{j,c}$ into \mathbf{r}_j . Conditional NPs (CNPs) [17, 20, 46] use a deterministic encoder that maps $\mathcal{D}_{j,c}$ to a point estimate $\bar{\mathbf{r}}_j$. Latent NPs [18, 16, 30] instead model uncertainty by placing a variational Gaussian over \mathbf{r}_j . Both approaches generate predictive means and variances via a decoder:

$$(\mu_{\mathbf{r}_j}, \sigma_{\mathbf{r}_j}^2) = f_{\text{dec}}(\mathbf{x}_{j,k}, \mathbf{r}_j; \psi), \quad (3)$$

where ψ is the parameter set of the NPs decoder module. CNPs maximize the predictive log-likelihood over meta-training tasks, while latent NPs optimize the predictive log-likelihood or an

81 Evidence Lower Bound (ELBO) to balance data fit and Kullback-Leibler (KL) regularization. Here,
 82 ELBO is usually approximated by the following form:

$$\mathbb{E}_{\tau_j}[\log p(\mathbf{Y}_j | \mathbf{X}_j, \mathcal{D}_{j,c})] \geq \mathbb{E}_{\tau_j} \left[\sum_{k \in [n_j]} \mathbb{E}_{q(r_j | \mathcal{D}_j)} [\log \mathcal{N}_{j,k}] - \text{KL}[q(r_j | \mathcal{D}_j) \parallel q(r_j | \mathcal{D}_{j,c})] \right], \quad (4)$$

83 where $\mathcal{N}_{j,k}$ is shorthand for $\mathcal{N}(\mathbf{y}_{j,k} | \mu_{r_j}(\mathbf{x}_{j,k}), \text{diag}(\sigma_{r_j}^2(\mathbf{x}_{j,k})))$. In this paper, we are focusing on
 84 latent NPs, where we perform test-time scaling to sample from the latent variable more accurately.

85 2.3 Sequential Monte Carlo Samplers

86 In this section, we briefly review Sequential Monte Carlo Samplers (SMCS) [10], which form the
 87 foundational framework for our test-time scaling approach. For a more comprehensive overview,
 88 we refer the reader to Del Moral et al. [10], Dai et al. [9].

89 Let $\pi(x) := \gamma(x)/Z$ denote a target distribution, where the unnormalized density $\gamma(x)$ is known
 90 pointwise but the normalization constant Z is unknown. SMCS aims to approximate expectations
 91 with respect to π , such as $\mathbb{E}\pi[f]$ for a test function f , while simultaneously providing an estimate
 92 of Z . Rather than sampling directly from the target π , SMCS constructs a sequence of intermediate
 93 distributions $\{\pi_t\}_{t=0}^T$, beginning with a tractable initial distribution π_0 and gradually transporting it
 94 toward the target $\pi_T := \pi$. These intermediate distributions are designed to interpolate between π_0
 95 and π , typically becoming increasingly complex as t increases. Since the intermediate distributions
 96 are generally not analytically tractable, SMCS employs sequential importance sampling to adjust
 97 the samples accordingly.

98 In detail, let γ_t be the unnormalized density of π_t , i.e., $\pi_t(x) = \gamma_t(x)/Z_t$ for $t = 0, \dots, T$ (note
 99 that $Z_0 = 1$ as π_0 is assumed to be fully known). SMCS introduces *forward transition kernels*
 100 $\{F_t(x_t | x_{t-1})\}_{t=1}^T$ which propagates a sample x_{t-1} to x_t . Starting with $x_0 \sim \pi_0$, a sample path
 101 $x_{0:T} := (x_0, \dots, x_T)$ is generated from the joint proposal with density,

$$Q(x_{0:T}) := \pi_0(x_0) \prod_{t=1}^T F_t(x_t | x_{t-1}). \quad (5)$$

102 In principle, one would like to use the marginal $\int Q(x_{0:T}) dx_{0:T-1}$ as a proposal density for π ,
 103 but this is generally intractable. Instead, SMCS augments the target to the path space $x_{0:T}$ with a
 104 sequence of *backward transitions* $\{B_{t-1}(x_{t-1} | x_t)\}_{t=1}^T$ and defines the unnormalized path density

$$\Pi(x_{0:T}) := \frac{\Gamma(x_{0:T})}{Z} \text{ where } \Gamma(x_{0:T}) = \gamma(x_T) \prod_{t=1}^T B_{t-1}(x_{t-1} | x_t). \quad (6)$$

105 A sample path $x_{0:T}$ drawn from Q is then corrected with the importance weights,

$$w_T(x_{0:T}) := \frac{\Gamma(x_{0:T})}{Q(x_{0:T})} = \frac{\gamma(x_T) \prod_{t=1}^T B_{t-1}(x_{t-1} | x_t)}{\pi_0(x_0) \prod_{t=1}^T F_t(x_t | x_{t-1})}. \quad (7)$$

106 Given the forward and backward kernels, SMCS starts by drawing N i.i.d. *particles* $\{x_0^i\}_{i=1}^N$ from
 107 π_0 . At each iteration, the particles are sequentially extended by drawing samples from the forward
 108 kernel F_t , and the extended particles are re-weighted through the importance weights, which are
 109 computed in recursive fashion as follows:

$$w_0(x_0^i) = 1, \quad w_t(x_{0:t}^i) = w_{t-1}(x_{0:t-1}^i) \times \frac{\gamma_t(x_t^i) B_{t-1}(x_{t-1}^i | x_t^i)}{\gamma_{t-1}(x_{t-1}^i) F_t(x_t^i | x_{t-1}^i)} \text{ for } i = 1, \dots, N. \quad (8)$$

110 After completing the sequential update up to the step T , we have an estimator for the expectation
 111 $\mathbb{E}_\pi[f]$ and the normalization constant Z ,

$$\mathbb{E}_\pi[f] \approx \frac{1}{N} \sum_{i=1}^N \frac{w_T(x_{0:T}^i)}{\sum_{j=1}^N w_T(x_{0:T}^j)} f(x_T^i), \quad Z \approx \frac{1}{N} \sum_{i=1}^N w_T(x_{0:T}^i). \quad (9)$$

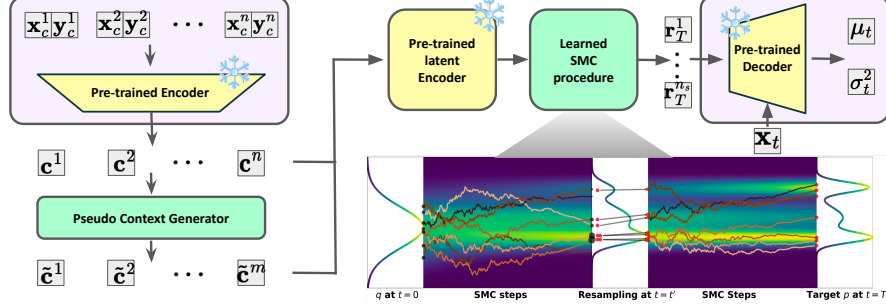


Figure 2: Schematic for TTSNP. TTSNP refines the latent variables from the variational posterior into more accurate samples from the true posterior using the learned transition kernels.

Learning for SMCS. Since both the intermediate distributions and the forward/backward kernels are crucial design choices in SMCS, it is beneficial to learn them when possible. Learning for SMCS can be formulated as a variational inference problem, where the ELBO provides a lower bound on the marginal likelihood defined by the SMCS procedure. A common approach is to begin with unadjusted Langevin diffusions and modify the drift term (involving the score functions) to maximize the ELBO [11, 19, 51, 7]. Among these, Chen et al. [7] introduced an optimal control framework for tuning a continuous-time extension of SMCS with resampling, which forms the foundation of our proposed algorithm. Refer to Appendix B to see additional explanations for SMCS.

3 Test Time Scaling for Neural Processes

3.1 Observation

As discussed in § 2.2, latent NP models are typically trained by maximizing the predictive log-likelihood or ELBO. Ideally, this results in a variational posterior $q(\mathbf{r}|\mathcal{D}_c)$ that approximates the true posterior $p(\mathbf{r}|\mathcal{D}_c) \propto \prod_{(x', y') \in \mathcal{D}_c} p(y'|x', \mathbf{r})p(\mathbf{r})$. However, due to training limitations and overfitting to specific context distributions, the learned posterior often deviates from the true posterior, especially at test time.

To validate this issue, we conducted experiments using a simple NP model [18] trained on 1D GP data with an RBF kernel. The same trained model was evaluated on unseen test datasets sampled under identical conditions. We compared the target marginal likelihoods using latent samples drawn from the variational posterior, as well as from importance sampling [IS; 50], Hamiltonian Monte Carlo [HMC; 44], and SMCS [10] with Unadjusted Langevin Algorithm (ULA) transitions. Since all methods use the same encoder and decoder, differences in log-likelihood stem solely from the quality of latent samples. Fig. 1 shows that the variational posterior underperforms compared to exact inference methods, rapidly saturating after 15–25 samples. This highlights its lack of diversity and poor calibration, indicating a failure to capture epistemic uncertainty. Based on these findings, we introduce a test-time scaling method using SMCS in the next section. We choose SMCS over other samplers due to its strong sample efficiency and fast convergence in high-dimensional regimes [40, 34, 57], which is also reflected in our empirical results.

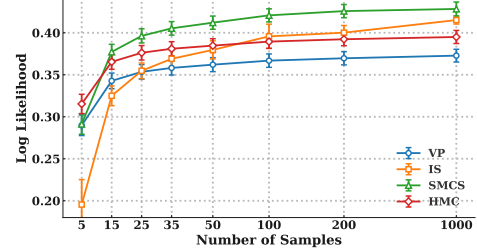


Figure 1: Marginal log likelihood comparison across variational posterior (VP), IS, HMC, and SMCS using the NP model [18]. The x-axis indicates the number of latent samples used for estimation.

3.2 Test Time Scaling with Sequential Monte Carlo Samplers

As mentioned in § 3.1, even after being trained on large-scale datasets, the latent encoder in latent NP models often fails to fully capture functional uncertainty—particularly when faced with structured or out-of-distribution test data, such as those arising from unseen kernels. To mitigate this

issue, we propose a test-time scaling approach that enhances uncertainty estimation without requiring any modification to the model’s pre-trained parameters. This design choice allows us to retain the representational power learned from the large-scale training data while refining the posterior inference process during evaluation.

Our key idea is to resample the global latent variable \mathbf{r} , which governs functional uncertainty in latent NP variants. Since the quality of uncertainty estimation is directly influenced by how well this latent distribution aligns with the test context, we aim to improve it via a principled sampling-based method. Specifically, we adopt the SMCS framework [10, 7] to transform an initial set of samples drawn from the variational distribution $q(\mathbf{r} \mid \mathcal{D}_c)$ into a set that better approximates the true posterior. To implement this, we first construct a sequence of intermediate distributions that gradually interpolate between the initial variational posterior and the target posterior. Then, particle transitions are guided by a parameterized forward kernel and backward kernel, both modeled using a neural network. The forward kernel nudges samples toward high-probability regions of the target distribution, while the backward kernel helps estimate gradients and improve learning stability during training. We also define an SMCS-based training objective derived from KL divergence [39, 7] and the marginal log-likelihood, which facilitates the learning of efficient and effective transitions for posterior refinement. To define an SMCS procedure, we must specify the initial distribution, intermediate distributions, and transition kernels. We detail our choices for each below.

Initial distribution. While the initial distribution is typically chosen to be simple, in our case with a pre-trained NP model, the variational posterior $q(\mathbf{r} \mid \mathcal{D}_c) = \mathcal{N}(\mathbf{r} \mid \mu_q, \sigma_q^2 I)$ serves as an effective starting point. Although it may not fully capture the uncertainty of the true posterior, it provides an informative and computationally tractable initialization that can be further refined through SMCS.

Intermediate distributions. We construct intermediate distributions via a geometric annealing path, enriched with a learned component. Specifically, for $t = 1, \dots, T$, we define

$$\pi_t(\mathbf{r}) \propto p(\mathbf{r} \mid \mathcal{D}_c)^{\beta_t} \tilde{q}(\mathbf{r} \mid \mathcal{D}_c)^{1-\beta_t} \cdot \tilde{q}(\mathbf{r} \mid \mathcal{D}_c \cup \mathcal{D}_p)^{1-\beta_t}, \quad 0 < \beta_1 < \dots < \beta_T = 1. \quad (10)$$

Here, the first term $p(\mathbf{r} \mid \mathcal{D}_c)$ is the true posterior. The second term includes a recalibrated variational posterior $\tilde{q}(\mathbf{r} \mid \mathcal{D}_c) := \mathcal{N}(\mathbf{r} \mid \mu_q, \sigma^2 I)$, retaining the original mean μ_q but using a fixed variance σ^2 . Empirically, setting $\sigma^2 = 1$ provides stable and robust performance. Together, these two factors form a slight modification of the standard geometric annealing path without requiring additional learning. The third factor introduces our core contribution, where the additional factor is defined as a variational posterior conditioning on the context appended with a *pseudo context set* \mathcal{D}_p —a synthetic future context predicted from the current estimate. Concretely, a permutation-invariant neural network $h(\mathcal{D}_c, \varepsilon)$ maps the original context \mathcal{D}_c and noise $\varepsilon \sim p(\varepsilon)$ to a pseudo context \mathcal{D}_p , defining an implicit generative model for the future data. Our design is inspired by Martingale posteriors [15, 30], where the predicted future data drive posterior uncertainty. In our case, the pseudo context serves as a form of “lookahead”, encouraging the model to maintain hypotheses that remain robust under plausible extensions of the observed context. This mechanism mitigates the overconfidence often observed in amortized inference by injecting uncertainty from possible yet-unseen data. Analogous to “chain-of-thought” reasoning in language models, the pseudo context enables the model to simulate and prepare for potential future scenarios, thereby enriching the intermediate distributions with anticipatory structure and improving posterior coverage. For implementation, we adopt the induced self-attention block [33] for h .

Forward and backward kernels. Given the intermediate distribution based on pseudo contexts, we define forward and backward kernels based on ULA. Following Chen et al. [7], we formulate our SMCS procedure as a time-discretizations of the continuous-time dynamics governed by the following Stochastic Differential Equation (SDE) evolving over $\tau \in [0, 1]$,

$$d\mathbf{r}_\tau^u = u(\mathbf{r}_\tau^u, \tau)d\tau + \sigma dW_\tau, \quad \mathbf{r}_0^u \sim \pi_0, \quad (11)$$

where $u : \mathbb{R}^{d_r} \times [0, 1] \rightarrow \mathbb{R}^{d_r}$ is a drift, σ is a diffusion coefficient, and W_τ is a standard Brownian motion. In parallel, we define the reverse-time SDE,

$$d\mathbf{s}_\tau^v = v(\mathbf{s}_\tau^v, \tau)d\tau + \sigma \bar{d}W_\tau, \quad \mathbf{s}_1^v \sim \pi, \quad (12)$$

with a drift v , diffusion σ , and $d\bar{W}_\tau$ denoting time-reversed Brownian motion. We expect these two SDEs to be time-reversals of each other, and their time-marginals at $\tau_t := t/T$ for $t = 0, \dots, T$ match the intermediate distributions we set, that is,

$$\mathbb{P}_{\tau_t}^u = \mathbb{P}_{\tau_t}^v = \pi_t \text{ for } t = 0, \dots, T, \quad (13)$$

where $\mathbb{P}_{\tau_t}^u$ and $\mathbb{P}_{\tau_t}^v$ denote time-marginals of the forward and reverse SDEs at time τ_t . There are potentially infinitely many choices of the pair (u, v) , but they must satisfy Nelson’s identity [45],

$$u - v = \sigma^2 \nabla \log \pi_\tau \text{ for } \tau \in [0, 1]. \quad (14)$$

Based on this, we choose the structural forms of u and v at $\{\tau_t\}_{t=0}^T$ as

$$\begin{aligned} u(\mathbf{r}_t, \tau_t) &:= \frac{\sigma^2}{2} (\nabla \log \tilde{\pi}_t(\mathbf{r}_t) + (1 - \beta_t) \text{NN}(\mathbf{r}_t, t, \mathcal{D}_c, \mathcal{D}_p)), \\ v(\mathbf{r}_t, \tau_t) &:= -\frac{\sigma^2}{2} (\nabla \log \tilde{\pi}_t(\mathbf{r}_t) + (1 - \beta_t) \text{NN}(\mathbf{r}_t, t, \mathcal{D}_c, \mathcal{D}_p)), \end{aligned} \quad (15)$$

where $\text{NN}(\cdot)$ is a neural network that takes as input the state \mathbf{r}_t , the step index t , and the representations of the context $\mathcal{D}_c \cup \mathcal{D}_p$ computed from a pre-trained encoder. If we were to directly use the score function of π_t , NN could simply output the Gaussian score. However, we instead wrap the inputs with a neural net for added flexibility. Note that NN encompasses this score function as a special case, ensuring compatibility with Nelson’s identity. Based on the choice of the drifts, we set the forward and backward transitions for SMCS as Euler-discretizations of the SDEs at $\{\tau_t\}_{t=0}^T$.

$$\begin{aligned} F_t(\mathbf{r}_t | \mathbf{r}_{t-1}) &= \mathcal{N}(\mathbf{r}_t | \mathbf{r}_{t-1} + h_t u(\mathbf{r}_{t-1}, \tau_{t-1}), 2h_t I), \\ B_{t-1}(\mathbf{r}_{t-1} | \mathbf{r}_t) &= \mathcal{N}(\mathbf{r}_{t-1} | \mathbf{r}_t + h_t v(\mathbf{r}_t, \tau_t), 2h_t I). \end{aligned} \quad (16)$$

While it is in principle possible to model the entire drifts using a neural network, recent empirical studies [22] indicate that approaches without explicit score functions often fail to work for even simple target distributions. By combining score-based gradients with neural proposals, our method stabilizes particle transport and enables efficient and expressive test-time inference. See Appendix B for details on the neural network architecture and modeling choices.

3.3 Training Objective

To ensure effective sampling with the SMCS framework, we must carefully design both the training objective and the data used to train F_t and B_t . If not properly designed, the importance weights may become overly concentrated on a few samples, leading to a severe particle degeneracy, which can negatively impact the diversity and effectiveness of the sampling process. To promote well-balanced importance weights and obtain high-quality posterior samples, along with the resampling, we employ two complementary loss terms: (1) the Kullback-Leibler (KL) divergence loss between two path measures, denoted as \mathcal{L}_{KL} , and (2) the log-likelihood maximization loss, denoted as \mathcal{L}_{LL} .

KL Divergence Loss. The KL divergence loss, $\mathcal{L}_{\text{path}}$, is formulated using two path measures, \mathbb{P}^u and \mathbb{P}^v , which correspond to the forward and backward SDEs, respectively. Intuitively, the forward path measure \mathbb{P}^u can be interpreted as the joint distribution $Q(\mathbf{r}_{0:T}^u)$ in the limit as $T \rightarrow \infty$ [5]. If this divergence is minimized to zero, it implies that the forward and backward transitions are perfectly time-reversible, thereby ensuring ideal sampling [7]. Consequently, training the transition kernels with this objective directly enhances the accuracy and quality of samples generated by the SMCS procedure. Then, given the sequential nature of our approach, the KL divergence is evaluated independently on each subinterval $[\tau_{t-1}, \tau_t]$, yielding:

$$\mathcal{L}_{\text{KL}} = \mathbb{E}_{\mathcal{D}_p} \left[\sum_{t=1}^T D_{\text{KL}} \left[\mathbb{P}_{[\tau_{t-1}, \tau_t]}^u || \mathbb{P}_{[\tau_{t-1}, \tau_t]}^v \right] \right], \quad (17)$$

where KL divergence is averaged by pseudo context due to the randomness \mathcal{D}_p . This formulation can be rewritten as a simplified objective using importance weights, following Chen et al. [7]. For detailed definitions and the full loss derivation, refer to Appendix B.

Marginal Log-likelihood Loss. In addition to the KL divergence loss, we introduce a log-likelihood maximization loss, \mathcal{L}_{LL} , which encourages the generated samples to capture diverse modes of the target distribution. This objective is designed to enhance the representation of epistemic uncertainty while reducing the number of samples required for reliable estimation. The loss is defined as $\mathcal{L}_{\text{LL}} = \sum_{(x,y) \in \mathcal{D}} \log \frac{1}{N} \sum_{i=1}^N \mathcal{N}(y | x, \mathbf{r}_T^i, \mathcal{D}_c)$, where \mathbf{r}_T^i denotes samples obtained from the final step of the SMCS procedure. Thus our final training objective \mathcal{L} becomes $\mathcal{L} := \mathcal{L}_{\text{KL}} + \mathcal{L}_{\text{LL}}$.

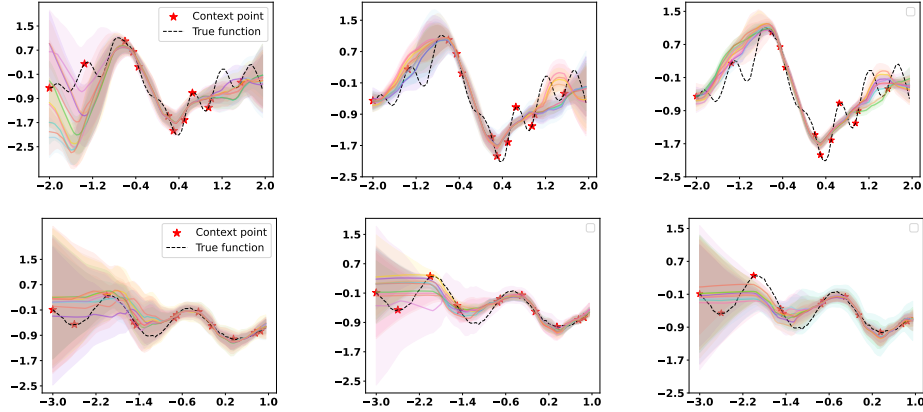


Figure 3: Visualization of posterior samples. The black dashed line shows the ground-truth function sampled from GP, and red stars denote context points. Colored lines and areas represent the mean and standard deviation decoded from each latent sample.

Table 1: Log-likelihood results for Sample Quality (left) and Matern to RQ (right) scenarios.

Model	RBF		Model	Matern		RQ	
	context	target		context	target	context	target
Pre-train	0.791 \pm 0.003	0.352 \pm 0.004	Pre-train	0.654 \pm 0.008	0.150 \pm 0.010	1.045 \pm 0.005	0.711 \pm 0.005
Fine-tune	0.833 \pm 0.005	0.382 \pm 0.010	Fine-tune	1.650 \pm 0.005	0.178 \pm 0.008	1.094 \pm 0.001	0.718 \pm 0.002
SMCS	0.868 \pm 0.001	0.405 \pm 0.001	SMCS	0.757 \pm 0.004	0.186 \pm 0.006	1.122 \pm 0.005	0.735 \pm 0.002
TTSNP (ours)	0.893 \pm 0.001	0.430 \pm 0.001	TTSNP (ours)	0.753 \pm 0.004	0.219 \pm 0.006	1.122 \pm 0.001	0.755 \pm 0.006

4 Experiments

In this section, we conduct a series of experiments to empirically validate the effectiveness of TTSNPs across a variety of settings, with a particular focus on regression tasks. We utilize two representative models from the latent NP family: simple NP [18], which is the earliest model in this line of work, and DANP [31], a recent model that has demonstrated strong performance and broad applicability across various tasks. Since our TTSNP is designed to be compatible with existing model architectures, we consider the following three baseline approaches: 1) *Pre-train*: a model that has been pre-trained on the training dataset is directly used for inference on the test dataset in a zero-shot manner, relying solely on the pre-trained latent variational posterior without any additional training; 2) *Fine-tune*: the latent path is further adapted for inference by training the transition kernel using the available data at test time; 3) *SMCS*: the samples from variational posterior transported by SMCS with the ULA transition kernels which does not require training. Unless otherwise specified, we fix the number of latent variable samples to 50 across both our method and all baselines to ensure fair comparison. In every result table, the best-performing metric is highlighted in **bold**, and all results are averaged over five different random seeds represented with 1-sigma error bars. In the experiment tables, gray box denotes that the evaluation dataset is the same as the training dataset (seen), while yellow box indicates a different evaluation dataset (unseen) for the TTSNP.

4.1 Simple NP

Sample Quality. We first conducted an experiment to demonstrate that our method, TTSNP, effectively and efficiently transforms samples drawn from the variational posterior to be better aligned with the true posterior. To this end, we pre-trained an NP model on one-dimensional GP data with RBF kernels. Using additional data from the same kernel, we trained both the ‘Fine-tune’ baseline and our TTSNP method. We then performed inference on an evaluation set and compared the quality of posterior samples produced by each method. Table 1 clearly demonstrates that TTSNP generates higher-quality latent variables compared to other baselines, *even with the same number of samples*, resulting in improved log-likelihoods for both context and target points. This result shows that our learned intermediate distributions and transition kernels generalize effectively not only to additional training data but also to the evaluation data set.

Table 2: Results of the context and target log-likelihoods for the GP regression task using the DANP models. " n D A " in the first column denotes that the n -D GP dataset was used to train for method A .

Model	1D RBF		Input range shift		Hyperparameter range shift	
	context	target	context	target	context	target
Pre-train	1.0488 \pm 0.001	0.4188 \pm 0.002	1.0921 \pm 0.000	0.2345 \pm 0.007	0.8136 \pm 0.000	-0.1971 \pm 0.015
SMCS	1.2725 \pm 0.001	0.4836 \pm 0.004	1.1874 \pm 0.001	0.4226 \pm 0.003	1.2051 \pm 0.001	0.1700 \pm 0.014
1D Fine-tune	1.1863 \pm 0.005	0.5095 \pm 0.004	1.1270 \pm 0.001	0.4663 \pm 0.004	0.8594 \pm 0.000	0.1084 \pm 0.004
1D TTSNP (Ours)	1.2729 \pm 0.001	0.6612 \pm 0.004	1.2639 \pm 0.001	0.6220 \pm 0.005	1.0890 \pm 0.001	0.3615 \pm 0.001
2D Fine-tune	1.0840 \pm 0.001	0.4150 \pm 0.003	1.0859 \pm 0.001	0.3809 \pm 0.001	0.9853 \pm 0.000	0.1485 \pm 0.001
2D TTSNP (Ours)	1.3099 \pm 0.001	0.6026 \pm 0.003	1.2429 \pm 0.000	0.5913 \pm 0.003	1.0414 \pm 0.000	0.3075 \pm 0.000

Analysis on generated pseudo context Furthermore, Fig. 3 illustrates how TTSNP benefits from the generated pseudo representations to obtain improved latent posterior samples. Specifically, the left, middle, and right sub-figures in Fig. 3 respectively show: (left) inference results using latent samples drawn from a variational posterior constructed solely from pseudo representations without access to true context points, (middle) inference using latent samples refined via TTSNP’s SMCS procedure, and (right) inference results from the Fine-tune baseline. As observed in the left figure, latent samples drawn from the variational posterior built on pseudo representations produce predictions and uncertainty estimates that differ from those of the original pre-trained variational posterior. This alternative perspective, introduced by the pseudo representations, becomes part of the intermediate distributions used in TTSNP. As a result, during the transition toward the true posterior, such diverse possibilities are continually integrated—allowing TTSNP to capture a broader range of posterior modes more sample-efficiently than standard SMCS.

Analysis on trade off between inference cost and the performance We conducted experiments to analyze the trade-off between inference cost and performance at test time. Specifically, we compared performance across different numbers of SMC steps, rather than using the default setting of $T = 10$ from previous experiments. As shown in Fig. 4, both SMCS and TTSNP exhibit improved performance as the number of inference steps increases. Notably, TTSNP achieves faster performance gains compared to SMCS, demonstrating its greater sample efficiency during inference. This behavior is similar to the trend observed in LLMs, where increasing inference cost—such as through more sampling or decoding steps—often leads to improved model performance.

Matern to RQ. In this scenario, we assume a pre-trained NP model that has been trained on GP data generated using the RBF kernel. We aim to perform inference on GP data generated with either a Matern or Rational Quadratic (RQ) kernel. Here, we assume that a small amount of additional training data is available for the Matern kernel, whereas no extra training data is provided for the RQ kernel. It is important to note that both the Matern and RQ kernels can be seen as different generalizations of the RBF kernel. Our expectation is that, if the SMCS framework is well trained to transform latent samples from the variational posterior into the true posterior using the Matern kernel data, the improved inference quality will also transfer to the RQ kernel, which is another generalization of the RBF kernel, despite the absence of direct training data. Table 1 shows the expected results, demonstrating that when the training and validation datasets share underlying characteristics, our learned intermediate distributions and transition kernels enable the NP to generalize effectively to the validation dataset.

4.2 Varying dimensional tasks with DANP

In this section, we design our experimental setup as follows. Leveraging the ability of the DANP model to handle data with varying input dimensions, we first pre-train the model using data generated from 3D and 4D GP with RBF kernels. Then, we generate training datasets from previously unseen 1D and 2D GP data with RBF kernels to train both the fine-tuned baseline and our TTSNP model. After training, we evaluate both models on 1D GP data generated in two different settings: (1) data within the same distribution as the training set, and (2) data with covariate shifts, such as a

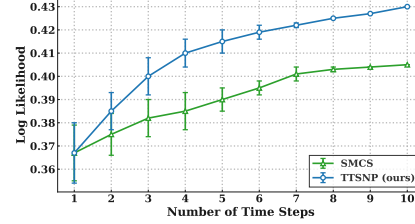


Figure 4: Marginal log likelihood comparison between SMCS and TTSNP using NP model. The x-axis indicates the number of time steps.

Table 3: Log-likelihood results for context and target values were obtained for image completion tasks using the EMNIST and Corrupted EMNIST datasets with 3 different corruptions.

Model	EMNIST		Corrupted	
	context	target	context	target
Pre-train	1.334 ± 0.003	0.574 ± 0.003	1.241 ± 0.003	0.551 ± 0.004
SMCS	1.349 ± 0.005	0.591 ± 0.010	1.351 ± 0.002	0.573 ± 0.005
Fine-tune	1.352 ± 0.004	0.601 ± 0.004	1.351 ± 0.002	0.583 ± 0.005
TTSNP (ours)	1.350 ± 0.002	0.620 ± 0.003	1.351 ± 0.001	0.613 ± 0.004

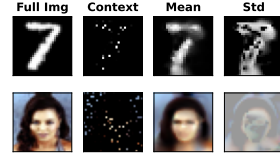


Figure 5: Visualization of the generated mean and covariance from TTSNP for the EMNIST and CelebA image completion tasks. Each column corresponds to the full image, the given context, the predicted mean, and the predicted standard deviation, respectively.

shifted input range or modified RBF kernel hyperparameters. This setup allows us to assess model performance both on in-distribution inference and under covariate shift conditions. The results in Table 2 confirm that the learned intermediate distributions and transition kernels remain effective even when the input dimension changes, as long as there is an underlying shared structure between the training and evaluation data—such as being generated from the same RBF kernel, or involving shifts in input range or kernel hyperparameters. This demonstrates the robustness of TTSNP under various covariate shift scenarios.

4.3 Image Completion

In the image completion scenario as well, we adopt a setup based on the DANP module. First, leveraging the capability of DANP to handle varying output dimensions, we pre-train the DANP model on the CelebA [35] image completion task. After this pre-training phase, we perform few-shot fine-tuning of the pre-trained DANP model on image completion tasks constructed from the EMNIST [8] dataset. Then we evaluate TTSNP with the EMNIST validation set and the Corrupted EMNIST dataset. Here, we used three different image corruption methods and report the averaged results from these corruptions: 1) ‘snow’, 2) ‘flip’, and 3) ‘brightness’. Table 3 clearly shows that, compared to other baselines, TTSNP more effectively adapts to new tasks with changing output dimensions, resulting in improved probabilistic inference. This shows that TTSNP is effective not only in scenarios where the input dimension varies but also in cases where the output dimension changes. As long as the task used to train the NP model shares some underlying features with the target task, TTSNP can leverage a small amount of additional data to successfully enhance inference performance. Also, similar to previous experiments, the learned SMCS procedure generalizes well to the unseen shifted evaluation tasks. Fig. 5 shows the predictive mean and standard deviation of EMNIST and CelebA data from TTSNP. Refer to Appendix D for additional experiments, including ablation studies on the number of training samples and training objectives.

5 Conclusion

In this paper, we first observed that the latent variational posteriors of existing NP models are often miscalibrated. To address this issue, we proposed TTSNP, a test-time scaling method based on a learned SMCS framework that transforms samples from the variational posterior into ones that better match the true posterior. TTSNP achieves this by learning both the construction of intermediate distributions through pseudo context representations conditioned on the input context and the approximation of score functions required for transition kernels. This enables TTSNP to guide samples more efficiently and effectively toward the true posterior by leveraging diverse and informative intermediate distributions. Our experiments demonstrate that TTSNP significantly improves posterior inference quality across a wide range of settings, including changes in input and output dimensions, covariate shifts, and cross-task generalization scenarios.

Nonetheless, TTSNP comes with some limitations. Training the model requires additional data and introduces extra memory and computational costs. Moreover, unlike standard pre-trained NPs, TTSNP requires additional computation at inference time, such as gradient evaluations for the SMCS procedure. Although this overhead can be mitigated by reducing the number of SMCS steps—allowing for a trade-off between efficiency and sample quality—future work may focus on minimizing this computational burden while maintaining strong performance.

References

- [1] Cesar Almecija, Apoorva Sharma, and Navid Azizan. Uncertainty-aware meta-learning for multimodal task distributions. *arXiv preprint arXiv:2210.01881*, 2022.
- [2] Han Altae-Tran, Bharath Ramsundar, Aneesh S Pappu, and Vijay Pande. Low data drug discovery with one-shot learning. *ACS central science*, 3(4):283–293, 2017.
- [3] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhersch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, April 2024. doi: 10.1145/3620665.3640366. URL <https://pytorch.org/assets/pytorch2-2.pdf>.
- [4] Matthew Ashman, Cristiana Diaconu, Junhyuck Kim, Lakee Sivaraya, Stratis Markou, James Requeima, Wessel P Bruinsma, and Richard E Turner. Translation equivariant transformer neural processes. *arXiv preprint arXiv:2406.12409*, 2024.
- [5] Paolo Baldi. *Stochastic calculus*. Springer, 2017.
- [6] Wessel P Bruinsma, Stratis Markou, James Requeima, Andrew YK Foong, Tom R Andersson, Anna Vaughan, Anthony Buonomo, J Scott Hosking, and Richard E Turner. Autoregressive conditional neural processes. *arXiv preprint arXiv:2303.14468*, 2023.
- [7] Junhua Chen, Lorenz Richter, Julius Berner, Denis Blessing, Gerhard Neumann, and Anima Anandkumar. Sequential controlled langevin diffusions. *arXiv preprint arXiv:2412.07081*, 2024.
- [8] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017.
- [9] C. Dai, J. Heng, P. E. Jacob, and N. Whiteley. An invitation to sequential Monte Carlo samplers. *Journal of the American Statistical Association*, 117:1587–1600, 2022.
- [10] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(3):411–436, 2006.
- [11] A. Doucet, W. Grathwohl, A. G. D. G. Matthews, and H. Strathmann. Score-based diffusion meets annealed importance sampling. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, 2022.
- [12] Leo Feng, Hossein Hajimirsadeghi, Yoshua Bengio, and Mohamed Osama Ahmed. Latent bottlenecked attentive neural processes. *arXiv preprint arXiv:2211.08458*, 2022.
- [13] Leo Feng, Frederick Tung, Hossein Hajimirsadeghi, Yoshua Bengio, and Mohamed Osama Ahmed. Memory efficient neural processes via constant memory attention block. *arXiv preprint arXiv:2305.14567*, 2023.
- [14] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.
- [15] Edwin Fong, Chris Holmes, and Stephen G Walker. Martingale posterior distributions. *arXiv preprint arXiv:2103.15671*, 2021.

- [16] A. Y. K. Foong, W. P. Bruinsma, J. Gordon, Y. Dubois, J. Requeima, and R. E. Turner. Meta-learning stationary stochastic process prediction with convolutional neural processes. In *Conference on Neural Information Processing Systems*, 2020.
- [17] M. Garnelo, D. Rosenbaum, C. J. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. J. Rezende, and S. M. A. Eslami. Conditional neural processes. In *International Conference on Machine Learning*, 2018.
- [18] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. A. Eslami, and Y. W. Teh. Neural processes. *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.
- [19] T. Geffner and J. Domke. Langevin diffusion variational inference. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics (AISTATS 2023)*, 2023.
- [20] J. Gordon, W. P. Bruinsma, A. Y. K. Foong, J. Requeima, Y. Dubois, and R. E. Turner. Convolutional conditional neural processes. In *International Conference on Learning Representations*, 2020.
- [21] Ido Greenberg, Shie Mannor, Gal Chechik, and Eli Meir. Train hard, fight easy: Robust meta reinforcement learning. *Advances in Neural Information Processing Systems*, 36:68276–68299, 2023.
- [22] Jiajun He, Yuanqi Du, Francisco Vargas, Dinghui Zhang, Shreyas Padhy, RuiKang OuYang, Carla Gomes, and José Miguel Hernández-Lobato. No trick, no treat: Pursuits and challenges towards simulation-free training of neural samplers. *arXiv preprint arXiv:2502.06685*, 2025.
- [23] Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-based systems*, 212:106622, 2021.
- [24] J. Heng, A. N. Bishop, G. Deligiannidis, and A. Doucet. Controlled sequential Monte Carlo. *The Annals of Statistics*, 48(5):2904–2929, 2020.
- [25] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.
- [26] H. Kim, A. Mnih, J. Schwarz, M. Garnelo, S. M. A. Eslami, D. Rosenbaum, and V. Oriol. Attentive neural processes. In *International Conference on Learning Representations*, 2018.
- [27] Sunwoo Kim, Minkyu Kim, and Dongmin Park. Test-time alignment of diffusion models without reward over-optimization, 2025. URL <https://arxiv.org/abs/2501.05803>.
- [28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.
- [29] Tuan Anh Le, Hyunjik Kim, Marta Garnelo, Dan Rosenbaum, Jonathan Schwarz, and Yee Whye Teh. Empirical evaluation of neural process objectives. In *NeurIPS workshop on Bayesian Deep Learning*, page 71, 2018.
- [30] Hyungi Lee, Eunggu Yun, Giung Nam, Edwin Fong, and Juho Lee. Martingale posterior neural processes. In *International Conference on Learning Representations*, 2023.
- [31] Hyungi Lee, Chaeyun Jang, Dongbok Lee, and Juho Lee. Dimension agnostic neural processes. *arXiv preprint arXiv:2502.20661*, 2025.
- [32] J. Lee, Y. Lee, J. Kim, E. Yang, S. J. Hwang, and Y. W. Teh. Bootstrapping neural processes. In *Conference on Neural Information Processing Systems*, 2020.
- [33] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019.

- [34] Xinzhu Liang, Joseph M Lukens, Sanjaya Lohani, Brian T Kirby, Thomas A Searles, and Kody JH Law. Smc is all you need: Parallel strong scaling. *arXiv preprint arXiv:2402.06173*, 2024.
- [35] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.
- [36] C. Louizos, X. Shi, K. Schutte, and M. Welling. The functional neural process. In *Conference on Neural Information Processing Systems*, 2019.
- [37] Stratis Markou, James Requeima, Wessel P Bruinsma, Anna Vaughan, and Richard E Turner. Practical conditional neural processes via tractable dependent predictions. *arXiv preprint arXiv:2203.08775*, 2022.
- [38] Alessandro Mastrototaro and Jimmy Olsson. Online variational sequential monte carlo. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL <https://openreview.net/forum?id=jbPc3pW6sC>.
- [39] Alex Matthews, Michael Arbel, Danilo Jimenez Rezende, and Arnaud Doucet. Continual repeated annealed flow transport monte carlo. In *International Conference on Machine Learning*, 2022.
- [40] Cory McCartan and Kosuke Imai. Sequential monte carlo for sampling balanced and compact redistricting plans. *The Annals of Applied Statistics*, 17(4):3300–3323, 2023.
- [41] Marko Mlikota and Frank Schorfheide. Sequential monte carlo with model tempering. *Studies in Nonlinear Dynamics and Econometrics*, 28(2):249–269, May 2023. ISSN 1558-3708. doi: 10.1515/snde-2022-0103. URL <http://dx.doi.org/10.1515/snde-2022-0103>.
- [42] Christian A. Naesseth, Scott W. Linderman, Rajesh Ranganath, and David M. Blei. Variational sequential monte carlo. In Amos J. Storkey and Fernando Pérez-Cruz, editors, *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, volume 84 of *Proceedings of Machine Learning Research*, pages 968–977. PMLR, 2018. URL <http://proceedings.mlr.press/v84/naesseth18a.html>.
- [43] Radford M Neal. Annealed importance sampling. *Statistics and computing*, 11:125–139, 2001.
- [44] Radford M Neal and Radford M Neal. Monte carlo implementation. *Bayesian learning for neural networks*, pages 55–98, 1996.
- [45] Edward Nelson. *Dynamical theories of Brownian motion*. Princeton University Press, 1967.
- [46] Tung Nguyen and Aditya Grover. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling. In *International Conference on Machine Learning*, 2022.
- [47] Deep Shankar Pandey. *Uncertainty-Aware Meta-Learning for Learning from Limited Data*. PhD thesis, Rochester Institute of Technology, 2025.
- [48] Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.
- [49] A. Thin, N. Kotelevskii, A. Doucet, A. Durmus, E. Moulines, and M. Panov. Monte Carlo variational auto-encoders. In *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)*, 2021.
- [50] Surya T Tokdar and Robert E Kass. Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(1):54–60, 2010.
- [51] Francisco Vargas, Shreyas Padhy, Denis Blessing, and Nikolas Nusken. Transport meets variational inference: Controlled Monte Carlo diffusions. In *International Conference on Learning Representations*, 2024.

- 493 [52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez,
494 Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Conference on Neural*
495 *Information Processing Systems*, 2017.
- 496 [53] Qi Wang and Herke Van Hoof. Learning expressive meta-representations with mixture of
497 expert neural processes. *Advances in neural information processing systems*, 35:26242–26255,
498 2022.
- 499 [54] Qi Wang, Marco Federici, and Herke van Hoof. Bridge the inference gaps of neural processes
500 via expectation maximization. In *The Eleventh International Conference on Learning Repre-*
501 *sentations*, 2023.
- 502 [55] H. Wu, J. Köhler, and F. Noé. Stochastic normalizing flows. In *Advances in Neural Information*
503 *Processing Systems 33 (NeurIPS 2020)*, 2020.
- 504 [56] Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Baker Grosse. Probabilistic
505 inference in language models via twisted sequential monte carlo. In *Forty-first International*
506 *Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenRe-
507 view.net, 2024. URL <https://openreview.net/forum?id=frA0NNBS1n>.
- 508 [57] Zheng Zhao, Sebastian Mair, Thomas B Schön, and Jens Sjölund. On feynman-kac training
509 of partial bayesian neural networks. In *International Conference on Artificial Intelligence and*
510 *Statistics*, pages 3223–3231. PMLR, 2024.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: Our main contribution is to fix the samples from the uncalibrated latent variational posterior using a learned SMC framework. And we propose a method which clearly aligned with this objective.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: In the § 5, we discuss about the limitation of TTSNP.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: We don't have any theoretical results for this paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We fully disclose all necessary information for reproducing the main experimental results, including dataset descriptions, experimental setup, evaluation metrics, and hyperparameters, clearly presented in [Appendix C](#).

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide detailed experimental setting in [Appendix C](#), and provide the original codebase within the supplementary material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Yes, we provide such details in [Appendix C](#).

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: All the experiments listed are averaged over five different random seeds, and we provide standard deviation as well.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Computational resource details are provided in [Appendix C](#).

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: Our research fully complies with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Yes, we provide full broader impacts in [Appendix E](#).

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [\[Yes\]](#)

Justification: We describe appropriate safeguards for responsible use and release of our research, including controlled access and clear usage guidelines, detailed in [Appendix E](#).

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [\[Yes\]](#)

Justification: All datasets and models utilized are clearly listed in [§ 4](#) and [Appendix C](#).

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Assets of our TTSNP framework (including code) are clearly provided in § 4, Appendix C, and supplementary materials.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our study does not include crowdsourcing or human-subject experiments.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: We do not have any issues regarding IRB approvals.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- 821 • We recognize that the procedures for this may vary significantly between institutions
822 and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the
823 guidelines for their institution.
- 824 • For initial submissions, do not include any information that would break anonymity
825 (if applicable), such as the institution conducting the review.

826 16. **Declaration of LLM usage**

827 Question: Does the paper describe the usage of LLMs if it is an important, original, or
828 non-standard component of the core methods in this research? Note that if the LLM is used
829 only for writing, editing, or formatting purposes and does not impact the core methodology,
830 scientific rigorousness, or originality of the research, declaration is not required.

831 Answer: [NA]

832 Justification: We used LLM only for grammar checking purposes.

833 Guidelines:

- 834 • The answer NA means that the core method development in this research does not
835 involve LLMs as any important, original, or non-standard components.
- 836 • Please refer to our LLM policy ([https://neurips.cc/Conferences/](https://neurips.cc/Conferences/2025/LLM)
837 [2025/LLM](https://neurips.cc/Conferences/2025/LLM)) for what should or should not be described.

A Related Works

Neural Processes The original Conditional Neural Process (CNP) [17] introduced a simple encoder-decoder architecture based on multilayer perceptrons. Its extension, the Neural Process (NP) [18], incorporated a global latent variable to model epistemic uncertainty. In this paper, they first suggest the ELBO objective to train the NP models with a latent variable. This design was further improved by Attentive Neural Processes (ANP) [26], which integrated attention mechanisms into the encoder for more expressive context representations. On the other hand, Gordon et al. [20] and Foong et al. [16] incorporate a convolutional deep set module to introduce translation equivariance into NP modules. However, since these models rely on convolutional computations, they are structurally limited and difficult to extend to data beyond three dimensions. Recently, Transformer Neural Processes (TNP) [46] replaced MLPs or attention layers with masked transformer layers to enhance scalability and long-range interaction modeling. Building on TNP, the Dimension-Agnostic Neural Process (DANP) [31] introduced the Dimension Aggregator Block (DAB), enabling a single model to handle tasks with varying input-output dimensions. Our proposed test-time scaling via SMC is developed on top of latent-path-based models such as NP and DANP.

Beyond these, many works have explored improved uncertainty quantification, expressiveness, and inference strategies. For instance, Functional Neural Processes (FNP) [36] used local latent variables to better capture function-specific variations. Bootstrapping Neural Processes (BNP) [32] introduced a residual bootstrapping mechanism to address model misspecification, while Martingale Posterior Neural Processes (MPNP) [30] proposed martingale posterior distribution [15] theory based Bayesian inference for more principled uncertainty estimation. Recent variants like Translation-Equivariant Transformer Neural Processes (TE-TNP) [4], Latent Bottlenecked Attentive Neural Processes (LBANP) [12], and Mixture-of-Experts Neural Processes (MoE-NP) [53] improve generalization and scalability through architectural innovations such as equivariant design, latent bottlenecks, and mixture modeling. Other contributions include Autoregressive Conditional Neural Processes (AR-CNP) [6] for sequential prediction, Self-normalized Importance-weighted Neural Processes (SI-NP) [54] for refined inference, Constant Memory Attentive Neural Processes (CMANP) [13] for memory-efficient modeling, and Gaussian Neural Processes (GNP) [37] for tractable predictive covariance modeling.

Sequential Monte Carlo Our work draws upon foundational concepts in SMC methods and their integration with learned stochastic dynamics. While SMC is designed to efficiently sample from sequential distributions, it has also been extensively explored for sampling complicated posterior distributions through sequential tempering [41], introducing learnable variational distributions [42, 38], or incorporating Langevin dynamics to replace sequential transitions [7]. In parallel, recent works have focused on inference-time scaling in foundation models such as LLMs and diffusion models, leveraging SMC to facilitate this scaling. Zhao et al. [56] treat the autoregressive prediction of LLMs as an SMC process and define a conditional target distribution with a learnable twisted function, enabling the guiding of pretrained LLMs to handle new tasks. In the diffusion models, Kim et al. [27] enhance sample efficiency by using a tempered SMC sampler that balances exploration and exploitation during the sampling process, achieving superior performance in tasks such as aesthetic reward optimization and multi-objective optimization.

B Details about SMCS and Modeling Transition kernels

B.1 Sequential Monte Carlo Samplers

In this section, we briefly review Sequential Monte Carlo Samplers (SMCS) [10], which form the foundational framework for our test-time scaling approach. For a more comprehensive overview, we refer the reader to Del Moral et al. [10], Dai et al. [9].

Let $\pi(x) := \gamma(x)/Z$ denote a target distribution, where the unnormalized density $\gamma(x)$ is known pointwise but the normalization constant Z is unknown. SMCS aims to approximate expectations with respect to π , such as $\mathbb{E}\pi[f]$ for a test function f , while simultaneously providing an estimate of Z . Rather than sampling directly from the target π , SMCS constructs a sequence of intermediate distributions $\{\pi_t\}_{t=0}^T$, beginning with a tractable initial distribution π_0 and gradually transporting it toward the target $\pi_T := \pi$. These intermediate distributions are designed to interpolate between π_0

Algorithm 1 Multinomial Resampling

Require: Particles $\{x_t^i\}_{i=1}^N$ at time step t and corresponding importance weights $\{w_t^i\}_{i=1}^N$
Ensure: Resampled particles $\{\tilde{x}_t^i\}_{i=1}^N$ and normalized importance weights $\{\tilde{w}_t^i\}_{i=1}^N$.

- 1: Compute the normalized importance weights: $\tilde{w}_t^i = \frac{w_t^i}{\sum_{i=1}^N w_t^i}$ for $i \in \{1, \dots, N\}$
- 2: Compute the Effective Sample Size using \tilde{w}_t^i s: $\text{ESS} = \frac{1}{\sum_{i=1}^N (\tilde{w}_t^i)^2}$
- 3: **if** $\text{ESS} < 0.3N$ **then**
- 4: **for** $i = 1$ to N **do**
- 5: Sample j from $\{1, \dots, N\}$ using multinomial distribution with probability $\{\tilde{w}_t^i\}_{i=1}^N$
- 6: Replace particle x_t^i with $\tilde{x}_t^i = x_t^j$
- 7: **end for**
- 8: Replace normalized importance weights \tilde{w}_t^i s with $\tilde{w}_t^i = \frac{1}{N}$ for $i \in \{1, \dots, N\}$
- 9: **else**
- 10: Keep particles $\{\tilde{x}_t^i\}_{i=1}^N$ and normalized importance weights $\{\tilde{w}_t^i\}_{i=1}^N$
- 11: **end if**

890 and π , typically becoming increasingly complex as t increases. Since the intermediate distributions
 891 are generally not analytically tractable, SMCS employs sequential importance sampling to adjust
 892 the samples accordingly.

893 In detail, let γ_t be the unnormalized density of π_t , i.e., $\pi_t(x) = \gamma_t(x)/Z_t$ for $t = 0, \dots, T$ (note
 894 that $Z_0 = 1$ as π_0 is assumed to be fully known). SMCS introduce *forward transition kernels*
 895 $\{F_t(x_t | x_{t-1})\}_{t=1}^T$ which propagates a sample x_{t-1} to x_t . Starting with $x_0 \sim \pi_0$, a sample path
 896 $x_{0:T} := (x_0, \dots, x_T)$ is generated from the joint proposal with density,

$$Q(x_{0:T}) := \pi_0(x_0) \prod_{t=1}^T F_t(x_t | x_{t-1}). \quad (18)$$

897 In principle, one would like to use the marginal $\int Q(x_{0:T}) dx_{0:T-1}$ as a proposal density for the
 898 target π , but this is generally intractable. Instead, SMCS augments the target to the path space $x_{0:T}$
 899 with a sequence of *backward transition kernels* $\{B_{t-1}(x_{t-1} | x_t)\}_{t=1}^T$ and defines the unnormalized
 900 path density

$$\Pi(x_{0:T}) := \frac{\Gamma(x_{0:T})}{Z} \text{ where } \Gamma(x_{0:T}) = \gamma(x_T) \prod_{t=1}^T B_{t-1}(x_{t-1} | x_t). \quad (19)$$

901 A sample path $x_{0:T}$ drawn from Q is then corrected with the importance weights,

$$w_T(x_{0:T}) := \frac{\Gamma(x_{0:T})}{Q(x_{0:T})} = \frac{\gamma(x_T) \prod_{t=1}^T B_{t-1}(x_{t-1} | x_t)}{\pi_0(x_0) \prod_{t=1}^T F_t(x_t | x_{t-1})}. \quad (20)$$

902 Given the forward and backward kernels, SMCS starts by drawing N i.i.d. *particles* $\{x_0^i\}_{i=1}^N$ from
 903 π_0 . At each iteration, the particles are sequentially extended by drawing samples from the forward
 904 kernel F_t , and the extended particles are re-weighted through the importance weights, which are
 905 computed in recursive fashion as follows:

$$w_0(x_0^i) = 1, \quad w_t(x_{0:t}^i) = w_{t-1}(x_{0:t-1}^i) \times \frac{\gamma_t(x_t^i) B_{t-1}(x_{t-1}^i | x_t^i)}{\gamma_{t-1}(x_{t-1}^i) F_t(x_t^i | x_{t-1}^i)} \text{ for } i = 1, \dots, N. \quad (21)$$

906 After completing the sequential update up to the step T , we have an estimator for the expectation
 907 $\mathbb{E}_\pi[f]$ and the normalization constant Z ,

$$\mathbb{E}_\pi[f] \approx \frac{1}{N} \sum_{i=1}^N \frac{w_T(x_{0:T}^i)}{\sum_{j=1}^N w_T(x_{0:T}^j)} f(x_{0:T}^i), \quad Z \approx \frac{1}{N} \sum_{i=1}^N w_T(x_{0:T}^i). \quad (22)$$

908 Vanilla SMCS can suffer from particle degeneracy: after several iterations, only a handful of par-
 909 ticles carry nearly all the weight while the rest contribute little. A standard remedy is resampling,
 910 in which a new particle set is drawn from the current importance-weighted empirical distribution.

911 In practice, resampling is triggered adaptively, for instance, whenever the effective sample size falls
 912 below a chosen threshold. When the resampling is applied, the recursive importance weight updates,
 913 and the estimators for expectations and the normalizing constant require minor adjustments. Refer
 914 to [Algorithm 1](#) to see how we conducted resampling in this paper. For a full treatment, see Del Moral
 915 et al. [10].

916 **Choice of intermediate distributions.** There are various options for designing intermediate dis-
 917 tributions; see Del Moral et al. [10], Dai et al. [9] for examples. A common choice is the geometric
 918 annealing path setting $\gamma_t(x) = \pi_0(x)^{1-\beta_t}\gamma(x)^{\beta_t}$ with coefficients $0 = \beta_0 < \beta_1 < \dots < \beta_T = 1$.

919 **Choice of transition kernels.** The design of forward and backward kernels is a critical component
 920 of SMCS. A widely used approach is to set F_t as a transition invariant to γ_t , such as Metropolis-
 921 Hastings steps, as proposed in Annealed Importance Sampling (AIS) [43]. While there exist optimal
 922 backward kernels for given forward kernels [10], they are often intractable in practice. Hence, back-
 923 ward kernels are often chosen for tractability. Notably, in AIS, the forward and backward kernels
 924 are coupled in a specific way, allowing simplified importance weight computation that does not even
 925 require evaluating the forward transition densities. Alternatively, one may consider proposals based
 926 on the Unadjusted Langevin Algorithm (ULA), where the forward transition kernels are defined via
 927 a discretization of Langevin diffusion [24, 55, 49],

$$F_t(x_t | x_{t-1}) = \mathcal{N}(x_t | x_{t-1} + h_t \nabla \log \pi_t(x_{t-1}), 2h_t I), \quad (23)$$

928 with h_t denoting the step size. When h_t is small, the forward proposal is approximately time-
 929 reversible, making the following a natural choice for the backward kernel:

$$B_{t-1}(x_{t-1} | x_t) = F_t(x_{t-1} | x_t) = \mathcal{N}(x_{t-1} | x_t + h_t \nabla \log \pi_t(x_t), 2h_t I). \quad (24)$$

930 One may also consider underdamped Langevin that incorporates auxiliary momentums [19].

931 **Learning for SMCS.** Since both the intermediate distributions and the forward/backward kernels
 932 are crucial design choices in SMCS, it is beneficial to learn them when possible. Learning for SMCS
 933 can be formulated as a variational inference problem, where the ELBO provides a lower bound on
 934 the marginal likelihood defined by the SMCS procedure. A common approach is to begin with un-
 935 adjusted Langevin diffusions and modify the drift term (involving the score functions) to maximize
 936 the ELBO [11, 19, 51, 7]. Among these, Chen et al. [7] introduced an optimal control framework
 937 for tuning a continuous-time extension of SMCS with resampling, which forms the foundation of
 938 our proposed algorithm.

939 B.2 Model structures

940 As discussed in § 3.2, our TTSP framework models both the pseudo representation generator h
 941 and the forward/backward transitions u and v using neural transition kernels. In this section, we
 942 provide detailed descriptions of how these two neural modules are implemented.

943 We begin with the pseudo representation generator h . As noted in § 3.2, h is designed to be
 944 permutation-invariant to ensure exchangeability of the generated context representations. Specif-
 945 ically, it takes the context set representations R_c and a set of random noise variables $\epsilon \sim p(\epsilon)$ (sam-
 946 pled from a standard Gaussian distribution in our experiments) as input and produces randomized
 947 pseudo context representations.

Following Lee et al. [30], we implement h using the induced self-attention block (ISAB) [33], a
 permutation-invariant attention module. The pseudo representation R_p generated by ISAB is defined
 as:

$$\text{ISAB}(\epsilon) = \text{MAB}(\epsilon, \eta), \quad \text{where } \eta = \text{MAB}(R_c, \epsilon),$$

948 where MAB denotes Multihead Attention Blocks [52], and R_c is the encoded context representation.

949 Next, we describe the structure of the drift functions u and v , which follow the formulation in [Eq. 15](#):

$$u(\mathbf{r}_t, \tau_t) := \frac{\sigma^2}{2} (\nabla \log \tilde{\pi}_t(\mathbf{r}_t) + (1 - \beta_t) \text{NN}(\mathbf{r}_t, t, \mathcal{D}_c, \mathcal{D}_p)), \quad (25)$$

$$v(\mathbf{r}_t, \tau_t) := -\frac{\sigma^2}{2} (\nabla \log \tilde{\pi}_t(\mathbf{r}_t) + (1 - \beta_t) \text{NN}(\mathbf{r}_t, t, \mathcal{D}_c, \mathcal{D}_p)), \quad (26)$$

where $\text{NN}(\cdot)$ is a neural network that takes as input the latent state \mathbf{r}_t , the time step t , and the combined representations of the context and pseudo context sets $\mathcal{D}_c \cup \mathcal{D}_p$, obtained via the pre-trained encoder. Including t as input to NN helps model the adaptive variance of the intermediate posterior $\tilde{q}(\mathbf{r}|\mathcal{D}_c \cup \mathcal{D}_p)$.

The structure of NN is implemented as a 3-layer Multi-Layer Perceptron (MLP), with the time step t embedded using a sinusoidal positional encoding (PE) as in Vaswani et al. [52]. The final form of the network is:

$$\text{NN}(\mathbf{r}_t, t, \mathcal{D}_c, \mathcal{D}_p) = \text{MLP}(\text{concat}(\mathbf{r}_t, R_c, R_p) + \text{PE}(t/T)),$$

where MLP refers to a 3-layer feedforward network, and $\text{PE}(\cdot)$ is the sinusoidal positional embedding.

B.3 Details on KL divergence loss

Path measure and Radon-Nikodym Derivative The KL divergence loss, $\mathcal{L}_{\text{path}}$, is formulated using two path measures, \mathbb{P}^u and \mathbb{P}^v , which correspond to the forward and backward SDEs, respectively. Intuitively, the forward path measure \mathbb{P}^u can be interpreted as the joint distribution $Q(\mathbf{r}_{0:T}^u)$ in the limit as $T \rightarrow \infty$ [5]. If this divergence is minimized to zero, it implies that the forward and backward transitions are perfectly time-reversible, thereby ensuring ideal sampling [7]. Consequently, training the transition kernels with this objective directly enhances the accuracy and quality of samples generated by the SMCS procedure. A key requirement for enabling principled training and inference is the ability to compute the likelihood ratio w between the forward and reverse-time processes, known as the Radon-Nikodym derivative (RND) [51, 7]. This quantity serves as the backbone for defining meaningful objectives and ensuring the correctness of sample-based approximations. Without a tractable form of the RND, learning reliable transition dynamics becomes fundamentally intractable. Fortunately, recent advances [51, 7] provide a computable expression for this crucial quantity as in the following lemma, which we leverage directly in our framework.

Lemma B.1 (Radon-Nikodym Derivative [51]). *Let $\mathbb{P}_{[a,b]}^u$ and $\mathbb{P}_{[a,b]}^v$ denote the path space measures of the solutions to the forward and backward SDEs defined in Eq. 11 and Eq. 12, respectively, over the time interval $[a, b] \subset [0, 1]$, where the drift functions u and v satisfy Nelson’s identity. Assume that the marginal distributions satisfy $\mathbf{r}_a^u \sim \pi_a$ and $\mathbf{s}_b^v \sim \pi_b$. Then, $\mathbb{P}_{[a,b]}^u$ -almost surely, the RND between these two measures can be computed as:*

$$\begin{aligned} \ln w_{[a,b]} = \ln \frac{d\mathbb{P}_{[a,b]}^v}{d\mathbb{P}_{[a,b]}^u}(\mathbf{r}) &= \ln \pi_b(\mathbf{r}_b) - \ln \pi_a(\mathbf{r}_a) + \int_a^b \frac{\|u\|^2 - \|u - \sigma^2 \nabla \log \pi\|^2}{2\sigma^2}(\mathbf{r}_t, t) dt \\ &\quad + \int_a^b \frac{u - \sigma^2 \nabla \log \pi}{\sigma^2}(\mathbf{r}_t, t) \cdot d\mathbf{r}_t^v - \int_a^b \frac{u}{\sigma^2}(\mathbf{r}_t, t) \cdot d\mathbf{r}_t^u. \end{aligned}$$

As we can see in Lemma B.1, we can compute w by divide trajectories \mathbf{r}^u and \mathbf{s}^v into subtrajectories using multiple chunks as follows:

$$\ln w = \ln \frac{\mathbb{P}^v}{\mathbb{P}^u} = \ln \frac{\mathbb{P}_{[\tau_0, \tau_1]}^v}{\mathbb{P}_{[\tau_0, \tau_1]}^u} + \dots + \ln \frac{\mathbb{P}_{[\tau_{T-1}, \tau_T]}^v}{\mathbb{P}_{[\tau_{T-1}, \tau_T]}^u} = \ln w_{[\tau_0, \tau_1]} + \dots + \ln w_{[\tau_{T-1}, \tau_T]}. \quad (27)$$

Following Vargas et al. [51] and Chen et al. [7], we can further simplify the continuous-time RND expression from Lemma B.1 by applying a discretization approximation. This yields a practical discrete-time formulation suitable for implementation:

$$\ln w_{[\tau_{n-1}, \tau_n]}(\mathbf{r}) = \ln \frac{d\mathbb{P}_{[\tau_{n-1}, \tau_n]}^v}{d\mathbb{P}_{[\tau_{n-1}, \tau_n]}^u} \quad (28)$$

$$\approx \ln \pi_{\tau_n}(\mathbf{r}_{\tau_n}) - \ln \pi_{\tau_{n-1}}(\mathbf{r}_{\tau_{n-1}}) + \sum_{i=1}^L \frac{\ln p_{(n-1)L+i-1}^v(\mathbf{r}_{((n-1)L+i-1)h} | \mathbf{r}_{((n-1)L+i)h})}{\ln p_{(n-1)L+i}^u(\mathbf{r}_{((n-1)L+i)h} | \mathbf{r}_{((n-1)L+i-1)h})}, \quad (29)$$

where L is the number of steps per subtrajectory and h is the step size. Here, by the Euler-Maruyama discretization, forward and backward transition densities are computed as follows:

$$p_{(n-1)L+i-1}^u(\mathbf{r}_{\text{new}} | \mathbf{r}_{\text{pre}}) = \mathcal{N}(\mathbf{r}_{\text{new}} | \mathbf{r}_{\text{pre}} + hu(\mathbf{r}_{\text{pre}}, a'), h\sigma^2(a')) \quad (30)$$

$$p_{(n-1)L+i}^v(\mathbf{r}_{\text{pre}} | \mathbf{r}_{\text{new}}) = \mathcal{N}(\mathbf{r}_{\text{pre}} | \mathbf{r}_{\text{new}} + h(\sigma^2 \nabla \log \pi - u)(\mathbf{r}_{\text{new}}, b'), h\sigma^2(b')) \quad (31)$$

where a' and b' are $((n-1)L+i-1)h$ and $((n-1)L+i)h$, respectively.

Algorithm 2 Overall TTSNP inference algorithm

Require: Context dataset \mathcal{D}_c , trained pseudo context generator h , and learned forward and backward transitions u and v .

Ensure: Updated latent particles $\{\mathbf{r}_T^i\}_{i=1}^N$ and corresponding normalized importance weights $\{\tilde{w}_T^i\}_{i=1}^N$.

- 1: **First Stage: Generate pseudo Representation**
 - 2: Sample noise $\epsilon \sim p(\epsilon)$
 - 3: Generate pseudo representations R_p using $h(\mathcal{D}_c, \epsilon)$
 - 4: **Second Stage: Generate initial latent particles and importance weights**
 - 5: Sample N number of latent particles $\{\mathbf{r}_0^i\}_{i=1}^N$ from variational latent posterior $q(\mathbf{r}|\mathcal{D}_c)$
 - 6: Set corresponding initial importance weights $\{w_0^i\}_{i=1}^N$ where $w_0^i = \frac{1}{N}$ for $i \in \{1, \dots, N\}$
 - 7: **Third Stage: SMCS procedure**
 - 8: **for** $t = 1$ to T **do**
 - 9: **for** i to N **do**
 - 10: Sample $\mathbf{r}_t^i \sim \mathcal{N}(\mathbf{r}_t^i | \mathbf{r}_{t-1}^i + h_t u(\mathbf{r}_{t-1}^i, \tau_{t-1}), 2h_t I)$
 - 11: **end for**
 - 12: Update importance weight based on Eq. 8 and Eq. 29
 - 13: Resample the latent variables \mathbf{r}_t^i and update \tilde{w}_t^i following Algorithm 1
 - 14: **end for**
 - 15: **Forth Stage: Compute predictive distribution**
 - 16: With final $\{\mathbf{r}_T^i\}_{i=1}^N$ and $\{\tilde{w}_T^i\}_{i=1}^N$ compute the predictive distribution similar to Eq. 38
-

983 **KL divergence loss** Then, given the sequential nature of our approach, the KL divergence is
984 evaluated independently on each subinterval $[\tau_{t-1}, \tau_t]$, yielding:

$$\mathcal{L}_{\text{KL}} = \mathbb{E}_{\mathcal{D}_p} \left[\sum_{t=1}^T D_{\text{KL}} \left[\mathbb{P}_{[\tau_{t-1}, \tau_t]}^u || \mathbb{P}_{[\tau_{t-1}, \tau_t]}^v \right] \right], \quad (32)$$

985 where KL divergence is averaged by pseudo context due to the randomness \mathcal{D}_p . This formulation can
986 be rewritten as a simplified objective using importance weights, following Chen et al. [7]. Following
987 the approaches proposed in Matthews et al. [39] and Chen et al. [7], the KL divergence on each
988 interval can be calculated as:

$$D_{\text{KL}} \left[\mathbb{P}_{[\tau_{t-1}, \tau_t]}^u || \mathbb{P}_{[\tau_{t-1}, \tau_t]}^v \right] = -\mathbb{E}_{\mathbf{r}^u \sim \mathbb{P}_{[\tau_k, \tau_t]}^u} \left[\log w_{[\tau_{t-1}, \tau_t]}(\mathbf{r}^u) \cdot w_{[\tau_k, \tau_t]}(\mathbf{r}^u) \right], \quad (33)$$

989 where $\tau_k < \tau_{t-1}$ refers to the most recent resampling time prior to τ_{t-1} . Finally, the KL loss \mathcal{L}_{KL}
990 can be expressed in practice using importance weights as:

$$\mathcal{L}_{\text{KL}} = \mathbb{E}_{\mathcal{D}_p} \left[\sum_{t=1}^T \frac{1}{N} \sum_{i=1}^N \text{detach}(w_{\tau_k}^{(i)}) \log w_{[\tau_{t-1}, \tau_t]}^{(i)} \right], \quad (34)$$

991 where $\text{detach}(\cdot)$ indicates that gradients are not propagated through the resampled weights $w_{n-1}^{(i)}$. In
992 our experiment, for efficient training, we trained the model using a single set of \mathcal{D}_p .

993 B.4 Overall TTSNP latent variable sampling algorithm

994 In this section, we present the full inference procedure of TTSNP as an algorithm that uses an SMCS
995 procedure to refine latent samples toward the true posterior distribution. As outlined in Algorithm 2,
996 the overall method can be divided into four main stages: (1) ‘Generate pseudo representation’, (2)
997 ‘Generate initial latent particles and importance weights’, (3) ‘SMCS procedure’, and (4) Compute
998 predictive distribution’.

999 In the first stage, ‘Generate pseudo representation’, we use the pseudo context generator h to create
1000 pseudo representations conditioned on the context set. Next, in ‘Generate initial latent particles
1001 and importance weights’, we sample initial latent variables from the variational posterior $p(\mathbf{r} | \mathcal{D}_c)$
1002 and assign initial weights accordingly. These latent particles are then used to initialize the SMCS
1003 procedure.

Table 4: Model structure details of NP

CATEGORY	DETAILS
MODEL SPECIFICATIONS	
HIDDEN DIMENSION FOR DETERMINISTIC PATH	128
HIDDEN DIMENSION FOR LATENT PATH	128
MLP DEPTH FOR DETERMINISTIC PATH	4
MLP DEPTH FOR LATENT ENCODER PRE-MODULE	4
MLP DEPTH FOR LATENT ENCODER POST-MODULE	2
DECODER DEPTH	3
NUMBER OF PARAMETERS FOR 1D GP REGRESSION	232194

The third stage, ‘SMCS procedure’, iteratively updates the latent particles using the learned intermediate distributions—constructed using both the context and pseudo representations—and the learned forward and backward transition kernels u and v . Finally, in ‘Compute predictive distribution’, we use the updated latent particles and their associated importance weights to compute the final predictive distribution over target outputs.

C Experimental Details

To support reproducibility, we provide our full experimental code as part of the supplementary material. Our implementation is based on the official codebase¹ of DANP [31], and all experiments were performed using PyTorch [3]. Training and evaluation were carried out on either an NVIDIA GeForce RTX 3090 or an RTX A6000 GPU. We optimized all models using the Adam optimizer [28], combined with a cosine annealing schedule for the learning rate.

Unless stated otherwise, we selected hyperparameters based on validation log-likelihood across tasks, using the following search spaces: learning rates from $\{5 \times 10^{-5}, 7 \times 10^{-5}, 9 \times 10^{-5}, 1 \times 10^{-4}, 3 \times 10^{-4}, 5 \times 10^{-4}\}$, weight decay values of $\{0, 1 \times 10^{-5}\}$, and batch sizes of $\{16, 32\}$.

C.1 Model structural details

In this section, we summarize the architectural details of NP, DANP in Table 4 and Table 6, and their respective variants when integrated with the TTSNP method in Table 5 and Table 7. For the NP and NP+TTSNP models, the description is based on the 1D GP regression task. In contrast, both DANP and DANP+TTSNP are designed to maintain a consistent structure and parameter count, regardless of changes in input or output dimensionality.

A key aspect to note is that TTSNP extends the base latent NP architecture by introducing two additional components: a permutation-invariant pseudo representation generator using the ISAB module [33] and a gradient estimator using a Multi-Layer Perceptron (MLP) structure that approximates the score function corresponding to the variational posterior constructed from the pseudo representations.

¹<https://openreview.net/forum?id=uGJxl2odR0>

Table 5: Model structure details of NP with TTSNP

CATEGORY	DETAILS
MODEL SPECIFICATIONS	
HIDDEN DIMENSION FOR DETERMINISTIC PATH	128
HIDDEN DIMENSION FOR LATENT PATH	128
MLP DEPTH FOR DETERMINISTIC PATH	4
MLP DEPTH FOR LATENT ENCODER PRE-MODULE	4
MLP DEPTH FOR LATENT ENCODER POST-MODULE	2
DECODER DEPTH	3
NUMBER OF MULTIHEAD ATTENTION BLOCKS IN ISAB	2
OUTPUT DIMENSION FOR EACH BLOCK	128
HIDDEN DIMENSION FOR GRADIENT ESTIMATOR	256
MLP LAYER FOR GRADIENT ESTIMATOR	3
NUMBER OF PARAMETERS FOR 1D GP REGRESSION	562818

1029 C.2 Evaluation Metric for the tasks

1030 Following the approach of Le et al. [29], we evaluate baseline models, such as NP and DANP, using
 1031 the normalized predictive log-likelihood, which is estimated through Monte Carlo sampling as:

$$\frac{1}{|t|} \sum_{k \in t} \log p(y_{j,k} | x_{j,k}, \mathcal{D}_{j,c}) \approx \frac{1}{|t|} \sum_{k \in t} \log \left(\frac{1}{K} \sum_{k=1}^K p(y_{j,k} | x_{j,k}, \mathbf{r}_j^{(k)}) \right), \quad (35)$$

$$\frac{1}{|c|} \sum_{k \in c} \log p(y_{j,k} | x_{j,k}, \mathcal{D}_{j,c}) \approx \frac{1}{|c|} \sum_{k \in c} \log \left(\frac{1}{K} \sum_{k=1}^K p(y_{j,k} | x_{j,k}, \mathbf{r}_j^{(k)}) \right), \quad (36)$$

1032 where $\mathbf{r}_j^{(k)}$ are sampled from the variational posterior $q(\theta_j | \mathcal{D}_{j,c})$, and c and t denote the context and
 1033 target points, respectively. However, for the SMC and TTSNP, we utilize our importance weight w
 1034 when computing the predictive log-likelihood as follows:

$$\frac{1}{|t|} \sum_{k \in t} \log p(y_{j,k} | x_{j,k}, \mathcal{D}_{j,c}) \approx \frac{1}{|t|} \sum_{k \in t} \log \left(\sum_{k=1}^K \bar{w}_j^{(k)} p(y_{j,k} | x_{j,k}, \mathbf{r}_j^{(k)}) \right), \quad (37)$$

$$\frac{1}{|c|} \sum_{k \in c} \log p(y_{j,k} | x_{j,k}, \mathcal{D}_{j,c}) \approx \frac{1}{|c|} \sum_{k \in c} \log \left(\sum_{k=1}^K \bar{w}_j^{(k)} p(y_{j,k} | x_{j,k}, \mathbf{r}_j^{(k)}) \right), \quad (38)$$

1035 where $\bar{w}_j^{(k)}$ are the normalized importance weight for each sample $\mathbf{r}_j^{(k)}$ after the SMC procedures.

1036 C.3 Dataset details of n-dimensional GP Regression task

1037 To construct GP tasks for our experiments, we generate synthetic datasets using GPs equipped with
 1038 one of three commonly used kernels: the RBF kernel, the Matern 5/2 kernel, and the RQ kernel.
 1039 Each kernel introduces different inductive biases, allowing us to evaluate model robustness across a
 1040 diverse range of smoothness conditions and correlation structures.

The RBF kernel is defined as

$$k(\mathbf{x}, \mathbf{x}') = s^2 \exp \left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell^2} \right),$$

Table 6: Model structure details of DANP

CATEGORY	DETAILS
MODEL SPECIFICATIONS	
HIDDEN DIMENSION FOR LINEAR PROJECTION IN DIMENSION AGNOSTIC BLOCK	32
HIDDEN DIMENSION FOR SELF-ATTENTION IN DIMENSION AGNOSTIC BLOCK	32
HIDDEN DIMENSION FOR LATENT PATH TRANSFORMER	64
NUMBER OF LAYERS FOR LATENT PATH TRANSFORMER	2
HIDDEN DIMENSION FOR LATENT PATH SELF-ATTENTION	64
HIDDEN DIMENSION FOR LATENT PATH MLP LAYERS	128
NUMBER OF LAYERS FOR LATENT PATH MLP LAYERS	2
HIDDEN DIMENSION FOR DETERMINISTIC PATH TRANSFORMER	128
NUMBER OF LAYERS FOR DETERMINISTIC PATH TRANSFORMER LAYERS	6
NUMBER OF HEADS FOR DETERMINISTIC TRANSFORMER LAYERS	4
DECODER DEPTH	2
NUMBER OF PARAMETERS	334562

while the Matern 5/2 kernel is given by

$$k(\mathbf{x}, \mathbf{x}') = s^2 \left(1 + \frac{\sqrt{5}d}{\ell} + \frac{5d^2}{3\ell^2} \right), \quad \text{where } d = \|\mathbf{x} - \mathbf{x}'\|.$$

Both use randomly sampled output scales $s \sim \text{Unif}(0.1, 1.0)$ and lengthscales $\ell \sim \text{Unif}(0.1, 0.6)$, and additive Gaussian noise drawn from $p \sim \text{Unif}(0.1, 0.5)$ is applied to the outputs for numerical stability.

The Rational Quadratic kernel, which generalizes the RBF by integrating over a distribution of lengthscales, is defined as

$$k(\mathbf{x}, \mathbf{x}') = s^2 \left(1 + \frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\alpha\ell^2} \right)^{-\alpha},$$

where we fix the mixture parameter $\alpha = 1.0$. For its implementation, the lengthscale and output scale are sampled similarly from uniform distributions, specifically:

$$\ell = 0.1 + (0.6 - 0.1) \cdot \text{Uniform}(0, 1), \quad s = 0.1 + (1.0 - 0.1) \cdot \text{Uniform}(0, 1).$$

Given a batch of inputs x , the pairwise normalized distances are computed and scaled accordingly, and a small diagonal term $\sigma_\epsilon^2 I$ is added to the covariance matrix to ensure numerical stability.

In our n -dimensional GP regression setup, we design the data generation process to scale with input dimensionality, ensuring meaningful predictive inference across varying dimensions.

And for the number of context points $|c|$, we used the sampled number from the range $\text{Unif}(5n^2, 50n^2 - |c|)$ where n indicates the x dimension for the GP task. This quadratic scaling with n reflects the increased data requirements for higher-dimensional input spaces. Similarly, the number of target points $|t|$ is sampled from $\text{Unif}(5n^2, 50n^2 - |c|)$, maintaining a fixed upper limit on the total number of points per task. And, inputs $\mathbf{x} \in \mathbb{R}^n$ for both context and target sets are drawn independently from a uniform distribution over $[-2, 2]^n$.

This flexible GP data generation process allows us to simulate tasks with varying degrees of smoothness, structure, and input dimensionality, providing a rigorous testbed for evaluating uncertainty-aware inference methods.

Table 7: Model structure details of DANP with TTSNP

CATEGORY	DETAILS
MODEL SPECIFICATIONS	
HIDDEN DIMENSION FOR LINEAR PROJECTION IN DIMENSION AGNOSTIC BLOCK	32
HIDDEN DIMENSION FOR SELF-ATTENTION IN DIMENSION AGNOSTIC BLOCK	32
HIDDEN DIMENSION FOR LATENT PATH TRANSFORMER	64
NUMBER OF LAYERS FOR LATENT PATH TRANSFORMER	2
HIDDEN DIMENSION FOR LATENT PATH SELF-ATTENTION	64
HIDDEN DIMENSION FOR LATENT PATH MLP LAYERS	128
NUMBER OF LAYERS FOR LATENT PATH MLP LAYERS	2
HIDDEN DIMENSION FOR DETERMINISTIC PATH TRANSFORMER	128
NUMBER OF LAYERS FOR DETERMINISTIC PATH TRANSFORMER LAYERS	6
NUMBER OF HEADS FOR DETERMINISTIC TRANSFORMER LAYERS	4
DECODER DEPTH	2
NUMBER OF MULTIHEAD ATTENTION BLOCKS IN ISAB	2
OUTPUT DIMENSION FOR EACH BLOCK	64
HIDDEN DIMENSION FOR GRADIENT ESTIMATOR	192
MLP LAYER FOR GRADIENT ESTIMATOR	3
NUMBER OF PARAMETERS FOR 1D GP REGRESSION	475490

1057 **Input range shift** As mentioned earlier, we conducted experiments using inputs x sampled from
1058 the range $[-2, 2]$. To evaluate how well TTSNP and baseline methods generalize under covariate
1059 shift, we introduced an input range shift scenario. Specifically, we constructed a validation set by
1060 modifying only the input domain while keeping all other GP generation settings identical to those
1061 used with the RBF kernel. In this shifted setting, inputs x were sampled from $[-3, 1]$, creating a dis-
1062 tributional mismatch between training and evaluation. We then assessed the inference performance
1063 of each method on this shifted validation set to examine their robustness to covariate shifts in the
1064 input space.

1065 **Hyperparameter range shift** In addition to the input range shift, we also evaluated the general-
1066 ization ability of TTSNP and baseline methods under a kernel hyperparameter shift, another form of
1067 covariate shift where the statistical properties of the data generation process change between training
1068 and inference. Specifically, we designed a validation scenario in which only the range of hyperpa-
1069 rameters used in the RBF kernel was altered, while keeping other settings—such as the number of
1070 data points and the input range—identical to the original training setup.

1071 During training, RBF kernel parameters were sampled from $s \sim \text{Unif}(0.1, 1.0)$ and $\ell \sim$
1072 $\text{Unif}(0.1, 0.6)$. For the shifted validation set, we extended these ranges to $s \sim \text{Unif}(0.1, 2.0)$ and
1073 $\ell \sim \text{Unif}(0.1, 1.0)$, effectively increasing the variance and smoothness of the generated functions.
1074 We then performed inference on this new validation data to assess how well each method adapts
1075 when faced with unseen kernel configurations, thus testing robustness under structural distribution
1076 shifts.

1077 C.4 EMNIST Dataset

1078 For our experiments, we constructed a modified version of the EMNIST Balanced dataset ² [8], a
1079 widely used benchmark derived from the original NIST Special Database. The full dataset includes
1080 47 alphanumeric character classes, but we curated a 10-class subset for focused evaluation. From
1081 this selection, we sampled 24,000 images for training and 4,000 for testing.

1082 Each image is a grayscale digit or letter rendered in a 28×28 resolution, represented as a single-
1083 channel input. Pixel locations were linearly scaled to lie within the range $[-0.5, 0.5]$, and intensity
1084 values were normalized to the same interval. During training, for each episode, we randomly chose
1085 the number of context points $|c|$ from a uniform distribution over $[5, 45]$, and the number of target
1086 points $|t|$ was drawn from $\text{Unif}(5, 50 - |c|)$, ensuring a total of at most 50 points per task.

1087 **Corrupted EMNIST Dataset** For the Corrupted EMNIST dataset, we evaluated model robustness
1088 by applying three types of image corruptions to the EMNIST evaluation set: (1) *snow*, (2) *flip*, and
1089 (3) *brightness*.

1090 In the *snow* corruption, we randomly added white noise to the image by setting a subset of pix-
1091 els to the maximum intensity. Specifically, each pixel was independently selected to be corrupted
1092 with probability proportional to the corruption intensity. Formally, a binary mask was generated
1093 where each pixel had a probability $0.1 \times \text{intensity}$ of being set to 1, and the corrupted image was
1094 obtained by replacing the masked pixels with maximum brightness.

In the *brightness* corruption setting, we simulated intensity-based corruption by amplifying the
pixel values and clipping them to stay within valid bounds. Specifically, we applied the transforma-
tion

$$\text{data} = \text{clamp}(\text{data} \times (1 + \text{intensity}), 0, 1),$$

1095 which increases overall brightness proportionally to a given intensity factor, followed by clipping to
1096 ensure pixel values remain in the $[0, 1]$ range.

1097 For the *flip* corruption, to simulate geometric distortions, we randomly flipped the image along
1098 spatial dimensions. Each image has a 5% chance of being flipped horizontally and another indepen-
1099 dent 50% chance of being flipped vertically. This introduces structural variations while preserving
1100 pixel intensity distributions.

1101 C.5 CelebA Dataset

1102 For experiments involving natural image completion, we used the CelebA dataset ³ [35], a large-
1103 scale face dataset commonly used in generative modeling benchmarks. The dataset consists of
1104 162,770 training images, 19,867 for validation, and 19,962 for testing. All images were center-
1105 cropped and downsampled to 32×32 resolution with 3 RGB channels to reduce computational
1106 complexity while preserving key facial features.

1107 As in our EMNIST experiments, we transformed each image into a pixel-level function over spatial
1108 coordinates. Specifically, pixel coordinates were scaled to the range $[-0.5, 0.5]$ along both axes,
1109 and pixel intensity values in each RGB channel were normalized to the interval $[-0.5, 0.5]$. This
1110 setup enables the image completion task to be framed as a conditional regression problem: the
1111 model is provided with a subset of known pixel values (the context) and is tasked with predicting
1112 the remaining pixels (the targets).

1113 To simulate diverse conditioning scenarios, we sampled the number of context points $|c| \sim$
1114 $\text{Unif}(5, 45)$, and drew the number of target points $|t| \sim \text{Unif}(5, 50 - |c|)$, ensuring that each training
1115 episode contains a variable and realistic number of observed and queried pixels.

1116 D Additional experiments

1117 D.1 Ablation study

²<https://www.nist.gov/itl/products-and-services/emnist-dataset>

³<https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>

1118 **Number of training data** In this section, we eval-
 1119 uate the learning efficiency and convergence speed
 1120 of TTSNP compared to the Fine-tune baseline by
 1121 analyzing their target log-likelihood performance.
 1122 Both methods start from the same pre-trained NP
 1123 model trained on 1D GP data generated with an
 1124 RBF kernel. We vary the number of training data
 1125 batches from 10 to 1000 and measure performance
 1126 using the mean and standard deviation of the target
 1127 log-likelihood across five random seeds. As shown
 1128 in Fig. 6, TTSNP achieves significantly faster conver-
 1129 gence and consistently outperforms the Fine-tune
 1130 method, even with substantially fewer training sam-
 1131 ples. This result shows that our method not only
 1132 samples more improved latent variables but is also
 1133 more efficient to train compared to the Fine-tune.

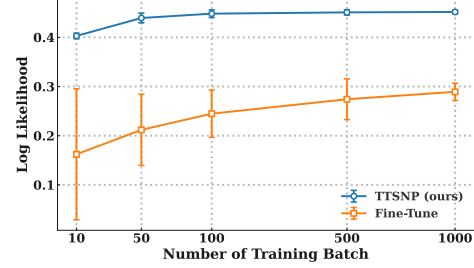


Figure 6: Comparison of target log-likelihood between TTSNP and the baseline ‘Fine-tune’ under varying amounts of training data, using the same number of samples.

1134 **Ablation on training objective** In this
 1135 section, we conduct an ablation study on
 1136 the training objectives used in our method.
 1137 As described in § 3.3, our full training loss
 1138 consists of two components: (1) the KL di-
 1139 vergence between two path measures, de-
 1140 noted as \mathcal{L}_{KL} , and (2) the log-likelihood
 1141 maximization loss, denoted as \mathcal{L}_{LL} . To
 1142 understand the individual contribution of
 1143 each objective, we evaluate models trained
 1144 with only one of the two loss terms.

1145 As shown in Table 8, both \mathcal{L}_{KL} and \mathcal{L}_{LL}
 1146 improve sample quality over the baseline
 1147 variational posterior from the pre-trained
 1148 model, highlighting their effectiveness in enhancing the efficiency and diversity of particles in the
 1149 learned SMCS procedure. However, the best performance is achieved when both objectives are
 1150 combined, as in TTSNP. This demonstrates that \mathcal{L}_{KL} and \mathcal{L}_{LL} are complementary, working together
 1151 to improve posterior approximation and overall inference quality.

Table 8: Log-likelihood results for the ablation study on training objectives. In the second and third rows, the notation $-A$ indicates that the objective A was excluded from the training loss during model training.

Model	RBF	
	context	target
TTSNP (ours)	0.893 \pm 0.001	0.430 \pm 0.001
- \mathcal{L}_{KL}	0.880 \pm 0.001	0.420 \pm 0.001
- \mathcal{L}_{LL}	0.872 \pm 0.001	0.415 \pm 0.001
SMCS	0.868 \pm 0.001	0.405 \pm 0.001

1152 **Number of samples** In this subsection, following
 1153 the setup in § 3.1, we perform an ablation study on
 1154 the number of samples used during inference. As
 1155 in other experiments described in § 4, we include
 1156 Pre-trained, Fine-tune, and SMCS as baselines for
 1157 comparison. Both TTSNP and SMCS are evaluated
 1158 with $T = 10$ inference steps.

1159 The experiment is conducted using a pre-trained NP
 1160 model trained on 1D Gaussian Process data with an
 1161 RBF kernel. For SMCS, we apply the Unadjusted
 1162 Langevin Algorithm as the transition kernel to sam-
 1163 ple from the posterior during inference. The re-
 1164 sults in Fig. 7 show that TTSNP achieves faster conver-
 1165 gence compared to other baselines as the num-
 1166 ber of samples increases. This demonstrates that
 1167 the learned intermediate distributions—constructed
 1168 using pseudo context representations—along with
 1169 the learned forward and backward transitions, effec-
 1170 tively guide latent samples toward the true posterior
 1171 while capturing diverse modes, resulting in improved sample efficiency.

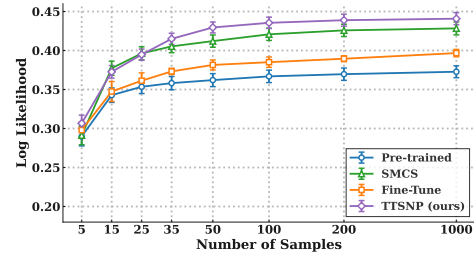


Figure 7: Comparison of target log-likelihood between TTSNP and the baselines ‘Pre-trained’, ‘SMCS’, and ‘Fine-tune’ using NP model trained with GP data generated by RBF kernel. Here, we compare the log-likelihood under the same varying amounts of latent samples.

Ablation on the Model structure In this section, we conduct an ablation study on the architectural design of the drift functions u and v . As detailed in [Appendix B.2](#), our default implementation defines the drift functions as follows:

$$u(\mathbf{r}_t, \tau_t) := \frac{\sigma^2}{2} (\nabla \log \tilde{\pi}_t(\mathbf{r}_t) + (1 - \beta_t) \text{NN}(\mathbf{r}_t, t, \mathcal{D}_c, \mathcal{D}_p)), \quad (39)$$

$$v(\mathbf{r}_t, \tau_t) := -\frac{\sigma^2}{2} (\nabla \log \tilde{\pi}_t(\mathbf{r}_t) + (1 - \beta_t) \text{NN}(\mathbf{r}_t, t, \mathcal{D}_c, \mathcal{D}_p)), \quad (40)$$

where $\text{NN}(\cdot)$ is implemented as a 3-layer MLP and is designed to approximate the score function $\nabla \log \tilde{q}(\mathbf{r} \mid \mathcal{D}_c \cup \mathcal{D}_p)$ of the intermediate distribution constructed from the context and pseudo context representations.

While this approach relies on a neural network to approximate the full score function, an alternative formulation can be derived by analytically decomposing the intermediate distribution using the Gaussian score form. In particular, we consider the following decomposition:

$$\nabla \log \pi_t(\mathbf{r}) := \beta_t \nabla \log p(\mathbf{r} \mid \mathcal{D}_c) + (1 - \beta_t) \nabla \log \tilde{q}(\mathbf{r} \mid \mathcal{D}_c) \quad (41)$$

$$+ (1 - \beta_t) \nabla \log \tilde{q}(\mathbf{r} \mid h(\mathcal{D}_c, t, \varepsilon_t) \cup \mathcal{D}_p), \quad (42)$$

$$= \nabla \log \tilde{\pi}_t(\mathbf{r}) + (1 - \beta_t) \frac{\mathbf{r} - \mu_{\text{NN}}(\mathcal{D}_c, \mathcal{D}_p, t)}{\sigma_{\text{NN}}^2(\mathcal{D}_c, \mathcal{D}_p, t)} \quad (43)$$

where $\tilde{\pi}_t(\mathbf{r})$ denotes the part of the intermediate distribution excluding contributions from the pseudo context, and the neural network models the additional gradient signal induced by the pseudo context generator h by modeling the mean μ_{NN} and the covariance σ_{NN}^2 of the Gaussian score. Also, more simply, we can design as follows:

$$\nabla \log \pi_t(\mathbf{r}) = \nabla \log \tilde{\pi}_t(\mathbf{r}) + (1 - \beta_t) \frac{\mathbf{r} - \mu_{\text{NN}}(\mathcal{D}_c, \mathcal{D}_p, t)}{\sigma^2}, \quad (44)$$

where σ^2 is the fixed variance, which we used for calibrating variational posteriors.

In this section, we compare the log-likelihood performance of TTSNP against the alternative drift modeling strategies described above, using 1D GP data generated with an RBF kernel. All methods share the same pre-trained NP base model, allowing for a fair evaluation of how different formulations of the drift function impact inference quality. We ensure a fair comparison by using the same training objectives, number of training samples, and number of inference steps across TTSNP and all other model variants—making the drift model structure the only varying factor. As shown in [Table 9](#), while alternative neural network designs also improve sample quality and log-likelihood performance compared to SMCS, TTSNP achieves the most significant improvement, highlighting the effectiveness of its architecture in modeling the score function and guiding posterior refinement.

D.2 Additional GP regression experiments

In [Table 10](#), we extend the GP regression experiments beyond those presented in [§ 4.1](#) and [§ 4.2](#) by exploring additional settings. Specifically, we first replicate the experiments originally conducted on 1D GP data in [§ 4.1](#), but now using 2D GP data. This allows us to empirically demonstrate that our method remains effective and directly applicable even when the input dimensionality increases. We first extend the **Sample Quality** experiment described in [§ 4.1](#) to a more challenging setting using 2D GP data generated with an RBF kernel. The results are reported in the left table of [Table 10](#). While the overall log-likelihood values are lower compared to the 1D case—due to the increased complexity of the 2D input space—the performance trends remain consistent. Specifically, TTSNP

Table 9: Log-likelihood results for the ablation study on drift model structures. In the second and third rows, the notations ‘Mean and Variance’ and ‘Mean’ indicate that the neural network is used to model both the mean and variance of the Gaussian score function, and only the mean, respectively.

Model	RBF	
	context	target
TTSNP (ours)	0.893 ± 0.001	0.430 ± 0.001
Mean and Variance	0.892 ± 0.002	0.420 ± 0.002
Mean	0.886 ± 0.002	0.421 ± 0.002
SMCS	0.868 ± 0.001	0.405 ± 0.001

Table 10: Log-likelihood results for Sample Quality (left) and Matern to RQ (right) scenarios.

Model	RBF		Model	Matern		RQ	
	context	target		context	target	context	target
Pre-train	-0.162 \pm 0.008	-0.392 \pm 0.010	Pre-train	-0.211 \pm 0.002	-0.462 \pm 0.003	0.299 \pm 0.004	0.084 \pm 0.005
Fine-tune	-0.156 \pm 0.005	-0.384 \pm 0.007	Fine-tune	-0.202 \pm 0.003	-0.440 \pm 0.005	0.310 \pm 0.004	0.092 \pm 0.005
SMCS	-0.154 \pm 0.004	-0.367 \pm 0.004	SMCS	-0.190 \pm 0.002	-0.432 \pm 0.002	0.318 \pm 0.003	0.104 \pm 0.004
TTSNP (ours)	-0.156 \pm 0.004	-0.351 \pm 0.005	TTSNP (ours)	-0.190 \pm 0.002	-0.420 \pm 0.001	0.316 \pm 0.003	0.110 \pm 0.003

continues to improve sample quality and enables efficient inference, confirming its effectiveness beyond simple 1D tasks and into higher-dimensional scenarios.

We also replicate the **Matern to RQ** experiment from § 4.1 in the 2D setting. The corresponding results are shown in the right table of Table 10. Again, we observe similar performance patterns to those seen in the 1D experiments, demonstrating that TTSNP enhances particle quality even in few-shot adaptation settings and generalizes effectively to unseen tasks. These results highlight the robustness of the learned intermediate distributions and transition kernels, even under high-dimensional covariate shift conditions.

E Broader Impact

This paper identifies a key limitation in latent Neural Process models: after pre-training, the global latent variables tend to be miscalibrated, resulting in low-quality samples from the variational posterior. To address this, we propose a method based on a Sequential Monte Carlo sampler that refines these latent samples to better match the true posterior distribution. While this approach introduces additional computational cost at test time compared to standard inference with pre-trained models, the cost is adjustable based on user preferences and is justified by the significant improvement in probabilistic inference quality. Aside from this computational trade-off, the proposed method poses no known negative societal impacts.