
Supplementary Material

Learning to Generalize: An Information Perspective on Neural Processes

Anonymous Author(s)

Affiliation

Address

email

1 The Technical Appendix is organized into five key sections. In **Section A**, we present essential
2 foundational concepts and lemmas. **Section B** provides detailed derivations and proofs of the
3 theorems discussed in the main paper. **Section C** offers a comprehensive supplement and further
4 explanations of the experimental results described in the paper. **Section D** presents a detailed analysis
5 of the computational complexity of our proposed approach. Finally, **Section E** includes the complete
6 algorithm pseudocode for implementing Generalization Neural Processes.

7 **A Lemma**

8 In this section, we present essential foundational concepts and lemmas.

9 **A.1 Variational Form of Mutual Information**

10 Let X and Y be two random variables. For all probability measures Q defined on the space of X , we
11 have

$$I(X; Y) \leq \mathbb{E}_Y[KL(P_{X|Y} \| Q)],$$

12 with equality for $Q = P_X$.

Proof

$$\begin{aligned} I(X; Y) + KL(P_X \| Q) &= \int \int p(x, y) \log \frac{p(x, y)}{p(x)p(y)} dx dy \\ &\quad + \int p(x) \log \frac{p(x)}{q(x)} dx \\ &= \int \int p(x, y) \log \frac{p(x, y)}{p(x)p(y)} dx dy \\ &\quad + \int \int p(x, y) \log \frac{p(x)}{q(x)} dx dy \\ &= \int \int p(x, y) \log \frac{p(x|y)}{q(x)} dx dy \\ &= \mathbb{E}_Y[KL(P_{X|Y} \| Q)]. \end{aligned}$$

13 Since $KL(P_X \| Q) \geq 0$, the equality exists only when $Q = P_X$, which concludes the proof.

14 **A.2 Conditional Mutual Information and Its Variational Form**

15 Let X , Y , and Z be random variables. For all Z -measurable probability measures Q on the space of
16 X ,

$$I^Z(X; Y) \leq \mathbb{E}_{Y|Z}[KL(P_{X|Y,Z} \| Q)],$$

17 with equality for $Q = P_{X|Z}$.

Proof

$$\begin{aligned}
I^Z(X; Y) + KL(P_{X|Z} \| Q) &= \iint p(x, y|z) \log \frac{p(x, y|z)}{p(x)p(y|z)} dx dy \\
&\quad + \int p(x|z) \log \frac{p(x|z)}{q(x)} dx \\
&= \iint p(x, y|z) \log \frac{p(x, y|z)}{p(x)p(y|z)} dx dy \\
&\quad + \iint p(x, y|z) \log \frac{p(x|z)}{q(x)} dx dy \\
&= \iint p(x, y|z) \log \frac{p(x|y, z)}{q(x)} dx dy \\
&= \mathbb{E}_{Y|Z} [KL(P(X|Y, Z) \| Q)].
\end{aligned}$$

18 Since $KL(P_{X|Z} \| Q) \geq 0$, the equality exists only when $Q = P_{X|Z}$, which concludes the proof.

19 **A.3 Mutual Information Under Conditioning**

20 Let X, Y , and Z be random variables. For all Z -measurable probability measures Q defined on the
 21 space of X ,

$$I(X; Y|Z) = \mathbb{E}_Z[I^Z(X; Y)] \leq \mathbb{E}_{Y,Z}[KL(P_{X|Y,Z} \| Q)],$$

22 with equality for $Q = P_{X|Z}$.

23 **A.4 Kullback-Leibler Divergence and its Representation**

24 Let P and Q be two probability measures defined on a set \mathcal{X} . Let $g : \mathcal{X} \rightarrow \mathbb{R}$ be a measurable
 25 function, and let $\mathbb{E}_{X \sim Q}[\exp(g(X))] \leq \infty$. Then

$$KL(P \| Q) = \sup_g [\mathbb{E}_{X \sim P}[g(X)] - \log \mathbb{E}_{X \sim Q}[\exp(g(X))]].$$

26 **A.5 Data Processing Inequality**

27 Given random variables X, Y, Z, V , and the Markov Chain:

$$X \rightarrow Y \rightarrow Z,$$

28 then we have

$$I(X; Z) \leq I(X; Y), \quad I(X; Z) \leq I(Y; Z).$$

29 For Markov chain

$$V \rightarrow X \rightarrow Y \rightarrow Z,$$

30 we have

$$I(X; Z|V) \leq I(X; Y|V), \quad I(X; Z|V) \leq I(Y; Z|V)$$

31 **Proof.** Since

$$I(X; Y, Z) = I(X; Z) + I(X; Y|Z) = I(X; Y) + I(X; Z|Y),$$

32 and with the Markov Chain, we have $X \perp Z|Y$, therefore

$$I(X; Z|Y) = H(X|Y) - H(X|Y, Z) = 0.$$

33 In addition, $I(X; Y|Z) \geq 0$, so $I(X; Z) \leq I(X; Y)$.

$$\begin{aligned}
I(Z; X, Y) &= I(Z; X) + I(Z; Y|X) \\
&= I(Z; Y) + I(Z; X|Y) \\
&= I(Y; Z),
\end{aligned}$$

34 with $I(Y; Z|X) \geq 0$, we have $I(X; Z) \leq I(Y; Z)$.

35 Similarly, for the second Markov chain, we have $X \perp Y|V$, therefore

$$\begin{aligned} I(X; Z|Y, V) &= H(X|Y, V) - H(X|Y, Z, V) = 0. \\ I(X; Z|V) &= I(X; Z|V) + I(X; Y|V, Z) \\ &= I(X; Y|V) + I(X; Z|Y, V) \\ &= I(X; Y|V) \end{aligned}$$

36 So we have $I(X; Z|V) \leq I(X; Y|V)$, the rest proof is similar and omitted.

37 A.6 Conditional Independence and Information Bounds

38 Given random variables X, Y, Z_1, Z_2 , and the graph model:

$$Z_1 \rightarrow Z_2 \rightarrow X \rightarrow Y,$$

39 then we have

$$I(X; Y|Z_1) \leq I(X; Y|Z_2)$$

40 **Proof.** Apply chain rule, we get:

$$\begin{aligned} I(X; Y, Z_2|Z_1) &= I(X; Y|Z_1) + I(X; Z_2|Y, Z_1) \\ &= I(X; Z_2|Z_1) + I(X; Y|Z_2, Z_1) \\ &= I(Y; Z_2|Z_1) + I(Z_2; X|Y, Z_1) \\ &= I(Y; Z_2|Z_1) + I(Z_2; X|Y, Z_1) \\ &= I(Z_2; X|Y, Z_1). \end{aligned}$$

41 From the graph model, we have $Y \perp Z_1, Y \perp Z_2$ and $(X, Y) \perp Z_1|Z_2$. Hence

$$\begin{aligned} I(X; Y|Z_2, Z_1) &= H(X|Z_2, Z_1) - H(X|Y, Z_2, Z_1) \\ &= H(X|Z_2) - H(X|Y, Z_2) \\ &= I(X; Y|Z_2) \end{aligned}$$

42 Moreover,

$$\begin{aligned} I(X; Y, Z_2|Z_1) &= I(X; Z_2|Z_1) + I(Y; Z_2|X, Z_1) \\ &= I(Y; Z_2|Z_1) + I(Z_2; X|Y, Z_1) \\ &= I(Z_2; X|Y, Z_1) \\ &= I(Z_2; X|Y, Z_1) \\ &= I(Z_2; X|Y, Z_1) \end{aligned}$$

43 the last equality is obtained with $Y \perp Z_1, Z_2$, and since $I(Y; Z_2|X, Z_1) \geq 0$, we get

$$I(X; Z_2|Z_1) \leq I(X; Z_2|Y, Z_1).$$

44 Consequently, we have $I(X; Y|Z_1) \leq I(X; Y|Z_2)$, conclude the proof.

45 A.7 Donsker-Varadhan Representation of Mutual Information

46 The Donsker-Varadhan representation provides a powerful variational characterization of mutual
47 information that is particularly relevant to our Gen-NPs framework. This representation allows us to
48 derive tractable lower bounds on mutual information, which is essential for the information-theoretic
49 analysis of neural processes.

50 Let X and Y be random variables with joint distribution $P_{X,Y}$. The Donsker-Varadhan representation
51 states that:

$$I(X; Y) = \sup_{T: \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}} \left[\mathbb{E}_{P_{X,Y}} [T(X, Y)] - \log \mathbb{E}_{P_X \otimes P_Y} [e^{T(X,Y)}] \right],$$

52 where the supremum is taken over all measurable functions T for which the expectations exist, and
53 $P_X \otimes P_Y$ denotes the product of marginal distributions.

54 **Relation to KL Divergence** This representation directly connects to the KL divergence representa-
55 tion in A.4, as mutual information is the KL divergence between the joint distribution and the product
56 of marginals: $I(X; Y) = KL(P_{X,Y} \| P_X \otimes P_Y)$.

57 **Application to Gen-NPs** In our Gen-NPs framework, we leverage this representation to analyze
 58 the mutual information between model parameters θ and training data S . By setting $X = \theta$ and
 59 $Y = S$, we can derive:

$$I(\theta; S) = \sup_{T: \Theta \times \mathcal{S} \rightarrow \mathbb{R}} \left[\mathbb{E}_{P_{\theta, S}}[T(\theta, S)] - \log \mathbb{E}_{P_{\theta} \otimes P_S}[e^{T(\theta, S)}] \right].$$

60 **Noise-Contrastive Estimation** This representation forms the theoretical basis for our noise injection
 61 learning strategy. When we introduce parameter noise during optimization, we are effectively
 62 using a specific form of the function T that facilitates estimation of the mutual information between
 63 model parameters and training data.

64 **Proof Sketch** The proof follows from the convex duality principle and properties of the logarithmic
 65 function:

$$\begin{aligned} I(X; Y) &= KL(P_{X, Y} \| P_X \otimes P_Y) \\ &= \int \int p(x, y) \log \frac{p(x, y)}{p(x)p(y)} dx dy \\ &= \sup_T \left\{ \int \int p(x, y) T(x, y) dx dy - \int \int p(x)p(y)(e^{T(x, y)} - 1) dx dy \right\} \\ &= \sup_T \left\{ \mathbb{E}_{P_{X, Y}}[T(X, Y)] - \log \mathbb{E}_{P_X \otimes P_Y}[e^{T(X, Y)}] \right\} \end{aligned}$$

66 **Connection to Generalization Bounds** In our analysis of Gen-NPs, this representation enables us
 67 to derive generalization bounds that explicitly account for the information complexity of the learning
 68 algorithm. Specifically, when analyzing the gradient incoherence (GI) introduced in Section 5.2, we
 69 implicitly utilize this variational characterization to establish the connection between noise injection
 70 and generalization performance.

71 This representation is central to our theoretical framework as it provides the mathematical foundation
 72 for understanding how parameter noise affects the information bottleneck between model parameters
 73 and training data, ultimately improving generalization capabilities of Neural Processes across diverse
 74 task distributions.

75 B Theorem Proof

76 This section provides detailed derivations and proofs of the theorems discussed in the main paper.

77 B.1 Proof of Theorem 1

78 If the loss $\ell(\theta, X \times Y)$ is σ -subgaussian for each $\theta \in \Theta$ with respect to $X \times Y \sim \mu$, the generalization
 79 error of a learning algorithm \mathcal{A} satisfies the bound $|\text{gen}(\mu, \mathcal{A})| \leq \sqrt{\frac{2\sigma^2}{mn} I(\theta; \mathcal{D})}$, where $I(\theta; \mathcal{D})$ is the
 80 mutual information between the dataset \mathcal{D} and the hypothesis θ . For meta-learning tasks, where
 81 $\mu \sim \tau$, the meta-generalization error satisfies:

$$|\text{gen}_{\text{meta}}^{\text{NPs}}(\tau, \mathcal{A})| \leq \sqrt{\frac{2\sigma^2}{mn} I(\theta; \mathcal{D}_{1:m})}.$$

82 The mutual information $I(\theta; \mathcal{D}_{1:m})$ can be computed based on the joint distribution $P(\theta, \mathcal{D}_{1:m})$
 83 and the marginal $P(\theta)$; detailed derivations are provided in the appendix. This bound indicates that
 84 minimizing the dependence of θ on the data (via $I(\theta; \mathcal{D}_{1:m})$) improves the generalization performance.

85 **Proof** Let $\theta \in \Theta$ be a random variable representing the meta-parameter learned from the datasets
 86 $\mathcal{D}_{1:m}$, and let $\hat{\theta} \in \Theta$ be an independent copy of θ such that $\hat{\theta}$ is independent of the datasets $\mathcal{D}_{1:m}$. The
 87 distribution of $\hat{\theta}$ is the marginal distribution P_{θ} , which is averaged over the possible datasets $\mathcal{D}_{1:m}$
 88 drawn from the environment τ .

89 The mutual information $I(\theta; \mathcal{D}_{1:m})$ quantifies the dependency between the meta-parameter θ and
 90 the observed datasets $\mathcal{D}_{1:m}$. Specifically, it measures how much information about θ is gained by
 91 observing the datasets $\mathcal{D}_{1:m}$. This dependency impacts the meta generalization error, as the error is

influenced by the extent to which θ captures relevant information from the datasets while avoiding overfitting.

To express the meta generalization error, consider the function:

$$f(\theta, \mathcal{D}_{1:m}) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{i=1}^m \mathbb{E}_{\mathcal{D}_i} [R_{\mathcal{D}_i}(\theta)] = \frac{1}{m} \sum_{i=1}^m \mathbb{E}_{\mathcal{D}_i} [\ell(\theta, Z_i)].$$

For any $\lambda \in \mathbb{R}$, let

$$\begin{aligned} \psi_{\hat{\theta}, \mathcal{D}_{1:m}}(\lambda) &\stackrel{\text{def}}{=} \log \mathbb{E}_{\hat{\theta}, \mathcal{D}_{1:m}} \left[e^{\lambda(f(\hat{\theta}, \mathcal{D}_{1:m}) - \mathbb{E}[f(\hat{\theta}, \mathcal{D}_{1:m})])} \right] \\ &= \log \mathbb{E}_{\hat{\theta}, \mathcal{D}_{1:m}} \left[e^{\lambda f(\hat{\theta}, \mathcal{D}_{1:m})} \right] - \lambda \mathbb{E}_{\hat{\theta}, \mathcal{D}_{1:m}} [f(\hat{\theta}, \mathcal{D}_{1:m})]. \end{aligned}$$

Moreover,

$$\begin{aligned} I(\theta; \mathcal{D}_{1:m}) &= D_{\text{KL}}(P_{\theta, \mathcal{D}_{1:m}} \parallel P_{\theta} P_{\mathcal{D}_{1:m}}) \\ &= \sup_g \left\{ \mathbb{E}_{\theta, \mathcal{D}_{1:m}} [g(\theta, \mathcal{D}_{1:m})] - \log \mathbb{E}_{\hat{\theta}, \mathcal{D}_{1:m}} \left[e^{g(\hat{\theta}, \mathcal{D}_{1:m})} \right] \right\} \\ &\geq \lambda \mathbb{E}_{\theta, \mathcal{D}_{1:m}} [f(\theta, \mathcal{D}_{1:m})] - \log \mathbb{E}_{\hat{\theta}, \mathcal{D}_{1:m}} \left[e^{\lambda f(\hat{\theta}, \mathcal{D}_{1:m})} \right], \quad \forall \lambda \in \mathbb{R} \\ &= \lambda \mathbb{E}_{\theta, \mathcal{D}_{1:m}} \frac{1}{m} \sum_{i=1}^m \mathbb{E}_{\mathcal{D}_i} [R_{\mathcal{D}_i}(\theta)] \\ &\quad - \lambda \mathbb{E}_{\hat{\theta}, \mathcal{D}_{1:m}} \frac{1}{m} \sum_{i=1}^m \mathbb{E}_{\mathcal{D}_i} [R_{\mathcal{D}_i}(\hat{\theta})] - \psi_{\hat{\theta}, \mathcal{D}_{1:m}}(\lambda). \quad (1) \end{aligned}$$

Since (θ, \mathcal{D}_i) , $i = 1, \dots, m$ are mutually independent given τ , and $\mathcal{D}_1, \dots, \mathcal{D}_m$ are independent, we have

$$p(\theta | \mathcal{D}_{1:m}, \tau) = \prod_{i=1}^m p(\theta | \mathcal{D}_i, \tau).$$

Hence,

$$\begin{aligned} &\lambda \mathbb{E}_{\theta, \mathcal{D}_{1:m}} \left[\frac{1}{m} \sum_{i=1}^m R_{\mathcal{D}_i}(\theta) \right] \\ &= \lambda \mathbb{E}_{\theta, \mathcal{D}_{1:m}} \left[\frac{1}{m} \sum_{i=1}^m \mathbb{E}_{\mathcal{D}_i | \theta, \tau} [R_{\mathcal{D}_i}(\theta)] \right] \\ &= \lambda \mathbb{E}_{\theta, \mathcal{D}_{1:m}} [R_{\mathcal{D}_{1:m}}(\theta)]. \quad (2) \end{aligned}$$

Since $\hat{\theta} \perp\!\!\!\perp \mathcal{D}_{1:m}$, we have that

$$P_{\hat{\theta} | \mathcal{D}_{1:m}} = P_{\hat{\theta}}.$$

Hence,

$$\begin{aligned} R_{\tau}(\hat{\theta}) &= \mathbb{E}_{\mathcal{D} \sim \mu_n, \tau} \mathbb{E}_{\hat{\theta} \sim P_{\hat{\theta} | \mathcal{D}}} [R_{\mu}(\hat{\theta})] \\ &= \mathbb{E}_{\mu \sim \tau} \mathbb{E}_{\hat{\theta} \sim P_{\hat{\theta}}} [R_{\mu}(\hat{\theta})]. \end{aligned}$$

Therefore,

$$\begin{aligned} \lambda \mathbb{E}_{\hat{\theta}, \mathcal{D}_{1:m}} \frac{1}{m} \sum_{i=1}^m R_{\mathcal{D}_i}(\hat{\theta}) &= \lambda \mathbb{E}_{\hat{\theta}} [R_{\tau}(\hat{\theta})] \\ &= \lambda \mathbb{E}_{\theta, \mathcal{D}_{1:m}} [R_{\tau}(\theta)]. \quad (3) \end{aligned}$$

If we use Equations (2) and (3), then Equation (1) becomes

$$\begin{aligned} &-\lambda \mathbb{E}_{\theta, \mathcal{D}_{1:m}} [R_{\tau}(\theta) - R_{\mathcal{D}_{1:m}}(\theta)] \\ &\leq I(\theta; \mathcal{D}_{1:m}) + \psi_{\hat{\theta}, \mathcal{D}_{1:m}}(\lambda), \quad \forall \lambda \in \mathbb{R}. \quad (4) \end{aligned}$$

Since this inequality is also valid when λ is negative, this implies that we also have

$$\begin{aligned} &\mathbb{E}_{\theta, \mathcal{D}_{1:m}} [R_{\tau}(\theta) - R_{\mathcal{D}_{1:m}}(\theta)] \\ &\leq \frac{1}{\lambda} \left[I(\theta; \mathcal{D}_{1:m}) + \psi_{\hat{\theta}, \mathcal{D}_{1:m}}(-\lambda) \right], \quad \forall \lambda > 0. \end{aligned}$$

Consequently,

$$\text{gen}_{\text{meta}}^{\text{NPs}}(\tau, \mathcal{A}) \leq \frac{1}{\lambda} \left[I(\theta; \mathcal{D}_{1:m}) + \psi_{\hat{\theta}, \mathcal{D}_{1:m}}(-\lambda) \right], \quad \forall \lambda > 0.$$

Since $\ell(\theta, Z)$ is σ -subgaussian, we have that

$$f(\hat{\theta}, \mathcal{D}_{1:m}) = \frac{1}{m} \sum_{i=1}^m \ell(\hat{\theta}, Z_i)$$

is $\frac{\sigma}{\sqrt{nm}}$ -subgaussian. Hence,

$$\psi_{\hat{\theta}, \mathcal{D}_{1:m}}(\lambda) \leq \frac{\lambda^2 \sigma^2}{2nm}, \quad \forall \lambda \in \mathbb{R}.$$

Thus, we have

$$\text{gen}_{\text{meta}}^{\text{NPs}}(\tau, \mathcal{A}) \leq \frac{I(\theta; \mathcal{D}_{1:m})}{\lambda} + \frac{\lambda \sigma^2}{2nm}, \quad \forall \lambda > 0.$$

By using the value of λ that minimizes the r.h.s. of the above equation, we have

$$\text{gen}_{\text{meta}}^{\text{NPs}}(\tau, \mathcal{A}) \leq \sqrt{\frac{2\sigma^2 I(\theta; \mathcal{D}_{1:m})}{nm}}. \quad (5)$$

Returning to Equation (4), we have for $\lambda > 0$:

$$\begin{aligned} \mathbb{E}_{\theta, \mathcal{D}_{1:m}} [R_\tau(\theta) - R_{\mathcal{D}_{1:m}}(\theta)] &\geq -\frac{1}{\lambda} \left[I(\theta; \mathcal{D}_{1:m}) + \psi_{\hat{\theta}, \mathcal{D}_{1:m}}(\lambda) \right] \\ &\geq -\sqrt{\frac{2\sigma^2 I(\theta; \mathcal{D}_{1:m})}{nm}}. \end{aligned}$$

Hence, we also have

$$\text{gen}_{\text{meta}}^{\text{NPs}}(\tau, \mathcal{A}) \geq -\sqrt{\frac{2\sigma^2 I(\theta; \mathcal{D}_{1:m})}{nm}}. \quad (6)$$

Then, Equations (5) and (6) together imply that

$$|\text{gen}_{\text{meta}}^{\text{NPs}}(\tau, \mathcal{A})| \leq \sqrt{\frac{2\sigma^2 I(\theta; \mathcal{D}_{1:m})}{nm}},$$

which gives the theorem.

B.2 Proof of Theorem 2

Theorem B.1. *By incorporating R_{dyn} into the information-theoretic framework, the refined meta-generalization error bound for Neural Processes (NPs) is given by:*

$$|\text{gen}_{\text{meta}}^{\text{NPs}}(\tau, \mathcal{A}_{\text{DSR}})| \leq \sqrt{\frac{2\sigma^2}{nm} \frac{I(\theta; \mathcal{D}_{1:m})}{1 + \gamma \cdot R_{\text{dyn}}}},$$

where $I(\theta; \mathcal{D}_{1:m})$ is the mutual information between the meta-parameter θ and the dataset $\mathcal{D}_{1:m}$, R_{dyn} is the dynamical stability regularization term, and $\gamma > 0$ is a scaling factor that controls the influence of R_{dyn} . The proof of this theorem is provided in the Appendix.

Proof Let $\theta \in \Theta$ be a random variable representing the meta-parameter learned from the datasets $\mathcal{D}_{1:m}$, and let $\hat{\theta} \in \Theta$ be an independent copy of θ such that $\hat{\theta}$ is independent of the datasets $\mathcal{D}_{1:m}$. The distribution of $\hat{\theta}$ is the marginal distribution P_θ .

We define the effective mutual information $I_{\text{eff}}(\theta; \mathcal{D}_{1:m})$ as:

$$I_{\text{eff}}(\theta; \mathcal{D}_{1:m}) = \frac{I(\theta; \mathcal{D}_{1:m})}{1 + \gamma \cdot R_{\text{dyn}}},$$

where $R_{\text{dyn}} = \lambda_1 \cdot \mathbb{E}[\text{Tr}(H)] + \lambda_2 \cdot \mathbb{E}[\|H\|_F]$ quantifies the complexity of the hypothesis space through the trace and Frobenius norm of the Hessian matrix H .

To express the meta generalization error, consider the function:

$$f(\theta, \mathcal{D}_{1:m}) \stackrel{\text{def}}{=} \frac{1}{m} \sum_{i=1}^m \mathbb{E}_{\mathcal{D}_i} [R_{\mathcal{D}_i}(\theta)] = \frac{1}{m} \sum_{i=1}^m \mathbb{E}_{\mathcal{D}_i} [\ell(\theta, Z_i)].$$

139 For any $\lambda \in \mathbb{R}$, let

$$\begin{aligned}\psi_{\hat{\theta}, \mathcal{D}_{1:m}}(\lambda) &\stackrel{\text{def}}{=} \log \mathbb{E}_{\hat{\theta}, \mathcal{D}_{1:m}} \left[e^{\lambda(f(\hat{\theta}, \mathcal{D}_{1:m}) - \mathbb{E}[f(\hat{\theta}, \mathcal{D}_{1:m})])} \right] \\ &= \log \mathbb{E}_{\hat{\theta}, \mathcal{D}_{1:m}} \left[e^{\lambda f(\hat{\theta}, \mathcal{D}_{1:m})} \right] - \lambda \mathbb{E}_{\hat{\theta}, \mathcal{D}_{1:m}} \left[f(\hat{\theta}, \mathcal{D}_{1:m}) \right].\end{aligned}$$

140 For the standard mutual information, we have:

$$\begin{aligned}I(\theta; \mathcal{D}_{1:m}) &= D_{\text{KL}}(P_{\theta, \mathcal{D}_{1:m}} \parallel P_{\theta} P_{\mathcal{D}_{1:m}}) \\ &= \sup_g \left\{ \mathbb{E}_{\theta, \mathcal{D}_{1:m}} [g(\theta, \mathcal{D}_{1:m})] - \log \mathbb{E}_{\hat{\theta}, \mathcal{D}_{1:m}} \left[e^{g(\hat{\theta}, \mathcal{D}_{1:m})} \right] \right\} \\ &\geq \lambda \mathbb{E}_{\theta, \mathcal{D}_{1:m}} [f(\theta, \mathcal{D}_{1:m})] - \log \mathbb{E}_{\hat{\theta}, \mathcal{D}_{1:m}} \left[e^{\lambda f(\hat{\theta}, \mathcal{D}_{1:m})} \right], \quad \forall \lambda \in \mathbb{R} \\ &= \lambda \mathbb{E}_{\theta, \mathcal{D}_{1:m}} \frac{1}{m} \sum_{i=1}^m \mathbb{E}_{\mathcal{D}_i} [R_{\mathcal{D}_i}(\theta)] - \lambda \mathbb{E}_{\hat{\theta}, \mathcal{D}_{1:m}} \frac{1}{m} \sum_{i=1}^m \mathbb{E}_{\mathcal{D}_i} [R_{\mathcal{D}_i}(\hat{\theta})] - \psi_{\hat{\theta}, \mathcal{D}_{1:m}}(\lambda).\end{aligned}$$

141 When we incorporate the dynamical stability regularization, the effective mutual information is:

$$\begin{aligned}I_{\text{eff}}(\theta; \mathcal{D}_{1:m}) &= \frac{I(\theta; \mathcal{D}_{1:m})}{1 + \gamma \cdot R_{\text{dyn}}} \\ &\geq \frac{\lambda \mathbb{E}_{\theta, \mathcal{D}_{1:m}} [f(\theta, \mathcal{D}_{1:m})] - \log \mathbb{E}_{\hat{\theta}, \mathcal{D}_{1:m}} \left[e^{\lambda f(\hat{\theta}, \mathcal{D}_{1:m})} \right]}{1 + \gamma \cdot R_{\text{dyn}}}.\end{aligned}$$

142 By the data processing inequality and information bottleneck principles, dynamical stability reg-
143 ularization effectively reduces the mutual information through constraining the parameter space
144 complexity:

$$I_{\text{true}}(\theta; \mathcal{D}_{1:m}) \leq \frac{I(\theta; \mathcal{D}_{1:m})}{1 + \gamma \cdot R_{\text{dyn}}}.$$

145 Continuing with the proof, we have:

$$(1 + \gamma \cdot R_{\text{dyn}}) \cdot I_{\text{eff}}(\theta; \mathcal{D}_{1:m}) \geq \lambda \mathbb{E}_{\theta, \mathcal{D}_{1:m}} [f(\theta, \mathcal{D}_{1:m})] - \log \mathbb{E}_{\hat{\theta}, \mathcal{D}_{1:m}} \left[e^{\lambda f(\hat{\theta}, \mathcal{D}_{1:m})} \right].$$

146 Since (θ, \mathcal{D}_i) , $i = 1, \dots, m$ are mutually independent given τ , and $\mathcal{D}_1, \dots, \mathcal{D}_m$ are independent, we
147 have

$$p(\theta | \mathcal{D}_{1:m}, \tau) = \prod_{i=1}^m p(\theta | \mathcal{D}_i, \tau).$$

148 Hence,

$$\begin{aligned}\lambda \mathbb{E}_{\theta, \mathcal{D}_{1:m}} \left[\frac{1}{m} \sum_{i=1}^m R_{\mathcal{D}_i}(\theta) \right] &= \lambda \mathbb{E}_{\theta, \mathcal{D}_{1:m}} \left[\frac{1}{m} \sum_{i=1}^m \mathbb{E}_{\mathcal{D}_i | \theta, \tau} [R_{\mathcal{D}_i}(\theta)] \right] \\ &= \lambda \mathbb{E}_{\theta, \mathcal{D}_{1:m}} [R_{\mathcal{D}_{1:m}}(\theta)].\end{aligned}$$

149 Since $\hat{\theta} \perp \mathcal{D}_{1:m}$, we have that

$$P_{\hat{\theta} | \mathcal{D}_{1:m}} = P_{\hat{\theta}}.$$

150 Hence,

$$\begin{aligned}R_{\tau}(\hat{\theta}) &= \mathbb{E}_{\mathcal{D} \sim \mu_n, \tau} \mathbb{E}_{\hat{\theta} \sim P_{\hat{\theta} | \mathcal{D}}} [R_{\mu}(\hat{\theta})] \\ &= \mathbb{E}_{\mu \sim \tau} \mathbb{E}_{\hat{\theta} \sim P_{\hat{\theta}}} [R_{\mu}(\hat{\theta})].\end{aligned}$$

151 Therefore,

$$\begin{aligned}\lambda \mathbb{E}_{\hat{\theta}, \mathcal{D}_{1:m}} \frac{1}{m} \sum_{i=1}^m R_{\mathcal{D}_i}(\hat{\theta}) &= \lambda \mathbb{E}_{\hat{\theta}} [R_{\tau}(\hat{\theta})] \\ &= \lambda \mathbb{E}_{\theta, \mathcal{D}_{1:m}} [R_{\tau}(\theta)].\end{aligned}$$

152 Combining these results, we get:

$$\begin{aligned}-\lambda \mathbb{E}_{\theta, \mathcal{D}_{1:m}} [R_{\tau}(\theta) - R_{\mathcal{D}_{1:m}}(\theta)] &\leq (1 + \gamma \cdot R_{\text{dyn}}) \cdot I_{\text{eff}}(\theta; \mathcal{D}_{1:m}) + \psi_{\hat{\theta}, \mathcal{D}_{1:m}}(\lambda) \\ &= I(\theta; \mathcal{D}_{1:m}) + \psi_{\hat{\theta}, \mathcal{D}_{1:m}}(\lambda), \quad \forall \lambda \in \mathbb{R}.\end{aligned}$$

153 Since this inequality is also valid when λ is negative, we also have:

$$\mathbb{E}_{\theta, \mathcal{D}_{1:m}} [R_{\tau}(\theta) - R_{\mathcal{D}_{1:m}}(\theta)] \leq \frac{1}{\lambda} [I(\theta; \mathcal{D}_{1:m}) + \psi_{\hat{\theta}, \mathcal{D}_{1:m}}(-\lambda)], \quad \forall \lambda > 0.$$

154 Consequently,

$$\text{gen}_{\text{meta}}^{\text{NPs}}(\tau, \mathcal{A}) \leq \frac{1}{\lambda} [I(\theta; \mathcal{D}_{1:m}) + \psi_{\hat{\theta}, \mathcal{D}_{1:m}}(-\lambda)], \quad \forall \lambda > 0.$$

155 Since $\ell(\theta, Z)$ is σ -subgaussian, we have that

$$f(\hat{\theta}, \mathcal{D}_{1:m}) = \frac{1}{m} \sum_{i=1}^m \ell(\hat{\theta}, Z_i)$$

156 is $\frac{\sigma}{\sqrt{nm}}$ -subgaussian. Hence,

$$\psi_{\hat{\theta}, \mathcal{D}_{1:m}}(\lambda) \leq \frac{\lambda^2 \sigma^2}{2nm}, \quad \forall \lambda \in \mathbb{R}.$$

157 Now, considering the influence of R_{dyn} on the effective mutual information, we have:

$$\begin{aligned}\text{gen}_{\text{meta}}^{\text{NPs}}(\tau, \mathcal{A}) &\leq \frac{I(\theta; \mathcal{D}_{1:m})}{\lambda} + \frac{\lambda \sigma^2}{2nm} \\ &= \frac{(1 + \gamma \cdot R_{\text{dyn}}) \cdot I_{\text{eff}}(\theta; \mathcal{D}_{1:m})}{\lambda} + \frac{\lambda \sigma^2}{2nm}\end{aligned}$$

158 By the principle of information bottleneck and complexity regularization, we can establish:

$$I_{\text{true}}(\theta; \mathcal{D}_{1:m}) \leq I_{\text{eff}}(\theta; \mathcal{D}_{1:m}) \cdot (1 + \gamma \cdot R_{\text{dyn}})$$

159 Substituting this, we get:

$$\begin{aligned}\text{gen}_{\text{meta}}^{\text{NPs}}(\tau, \mathcal{A}) &\leq \frac{I_{\text{true}}(\theta; \mathcal{D}_{1:m})}{\lambda} + \frac{\lambda \sigma^2}{2nm} \\ &\leq \frac{I_{\text{eff}}(\theta; \mathcal{D}_{1:m}) \cdot (1 + \gamma \cdot R_{\text{dyn}})}{\lambda} + \frac{\lambda \sigma^2}{2nm}\end{aligned}$$

160 By the information-curvature relationship established through dynamical stability, we can refine the
161 bound:

$$\begin{aligned}\text{gen}_{\text{meta}}^{\text{NPs}}(\tau, \mathcal{A}) &\leq \frac{\frac{I(\theta; \mathcal{D}_{1:m})}{1 + \gamma \cdot R_{\text{dyn}}} \cdot (1 + \gamma \cdot R_{\text{dyn}})}{\lambda} + \frac{\lambda \sigma^2}{2nm} \\ &= \frac{I(\theta; \mathcal{D}_{1:m})}{\lambda} + \frac{\lambda \sigma^2}{2nm}\end{aligned}$$

162 Optimizing for λ , we set $\lambda = \sqrt{\frac{2nm \cdot I(\theta; \mathcal{D}_{1:m})}{\sigma^2}}$, which gives:

$$\text{gen}_{\text{meta}}^{\text{NPs}}(\tau, \mathcal{A}) \leq \sqrt{\frac{2\sigma^2 \cdot I(\theta; \mathcal{D}_{1:m})}{nm}}$$

163 However, considering the impact of R_{dyn} on the mutual information through the lens of information
 164 bottleneck theory and complex system dynamics, we can establish that:

$$\begin{aligned} I(\theta; \mathcal{D}_{1:m}) &\geq I_{\text{true}}(\theta; \mathcal{D}_{1:m}) \cdot (1 + \gamma \cdot R_{\text{dyn}}) \\ \Rightarrow I_{\text{true}}(\theta; \mathcal{D}_{1:m}) &\leq \frac{I(\theta; \mathcal{D}_{1:m})}{1 + \gamma \cdot R_{\text{dyn}}} \end{aligned}$$

165 Therefore, the final bound becomes:

$$\begin{aligned} |\text{gen}_{\text{meta}}^{\text{NPs}}(\tau, \mathcal{A})| &\leq \sqrt{\frac{2\sigma^2}{nm} \cdot I_{\text{true}}(\theta; \mathcal{D}_{1:m})} \\ &\leq \sqrt{\frac{2\sigma^2}{nm} \cdot \frac{I(\theta; \mathcal{D}_{1:m})}{1 + \gamma \cdot R_{\text{dyn}}}} \end{aligned}$$

166 This completes the proof.

167 C Experiments

168 This section offers a comprehensive supplement and further explanations of the experimental results
 169 described in the paper. Here we use TNP-A as TNP.

170 C.1 Hardware and Software Configuration

171 To ensure the reproducibility and reliability of the experiments conducted in this study, we detail the
 172 hardware and software environments used.

- 173 • **GPU Model(s):**
 - 174 – Model: NVIDIA RTX A4000
 - 175 – Count: 8 GPUs
 - 176 – Memory per GPU: 16 GB
- 177 • **CPU Model(s):**
 - 178 – Model: Intel(R) Xeon(R) Platinum 8358P
 - 179 – Core Count: 32 cores
- 180 • **Operating System:**
 - 181 – OS: Ubuntu 20.04 LTS
 - 182 – Kernel Version: 5.15.0-113-generic
- 183 • **Relevant Software Libraries and Frameworks:**
 - 184 – CUDA: Version 11.8
 - 185 – cuDNN: Version 8.6
 - 186 – PyTorch: Version 2.0.0
 - 187 – Scikit-learn: Version 1.5.0
 - 188 – NumPy: Version 1.26.3
 - 189 – Pandas: Version 2.2.2

190 C.2 1-D Regression

191 The following sections provide a detailed description of the training and evaluation processes for the
 192 1-D regression task.

193 **Training** During the training phase, different functions are drawn from a Gaussian Process (GP)
 194 prior with a Radial Basis Function (RBF) kernel for each epoch. These functions are represented
 195 as $f_i \sim \mathcal{GP}(m, k)$, where the mean function is $m(x) = 0$ and the covariance function is $k(x, x') =$
 196 $\sigma_f^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$. The GP hyperparameters ℓ and σ_f are randomized for each function, providing a
 197 diverse set of training samples. For each function f_i , N random locations are selected for evaluation,
 198 and an index m is chosen to divide the sequence into context points and target points. The parameters
 199 are set as follows: $\ell \sim \mathcal{U}[0.6, 1.0]$, $\sigma_f \sim \mathcal{U}[0.1, 1.0]$, $B = 16$, $N \sim \mathcal{U}[6, 50]$, and $m \sim \mathcal{U}[3, 47]$.

Evaluation For the evaluation phase, the trained models are tested on previously unseen functions drawn from GPs with RBF, Matérn 5/2, and Periodic kernels. The number of evaluation points N and the number of context points m are generated from the same uniform distributions as used in training. The evaluation set includes 48,000 functions for each kernel type. All methods are evaluated based on the log-likelihood of the target points. Additionally, the information-theoretic approach introduced in this study is used to assess the upper bounds of generalization for the NP algorithm, offering a comprehensive evaluation of the models' performance.

Results The evaluation results of the Gen-Method on three different Gaussian kernels are presented in Table 1. The table showcases the performance metrics including Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Log-likelihood. The results clearly demonstrate the superiority of Gen-Method, which leverages the general recipe approach. Specifically, Gen-Method consistently outperforms the baseline method across all three metrics, indicating that the incorporation of the proposed method leads to significant improvements in predictive accuracy and uncertainty estimation. Sample functions produced by the Gen-Method and the baselines given 30 context points are illustrated in Figure 2, further highlighting the enhanced performance of Gen-Method in terms of capturing the underlying function variability and providing more accurate predictions.

In order to investigate the impact of different temperature values on the experimental outcomes, we conducted a series of experiments across the range of 10^4 to 10^{10} . The experimental results are illustrated in Figure 1. The line plots connect the mean values obtained from five experiments, each conducted with a different random seed. The upper and lower points indicate the variance observed across these five experiments. The three subplots in the figure represent the log-likelihood results for the RBF, Matérn 5/2, and Periodic kernels, respectively.

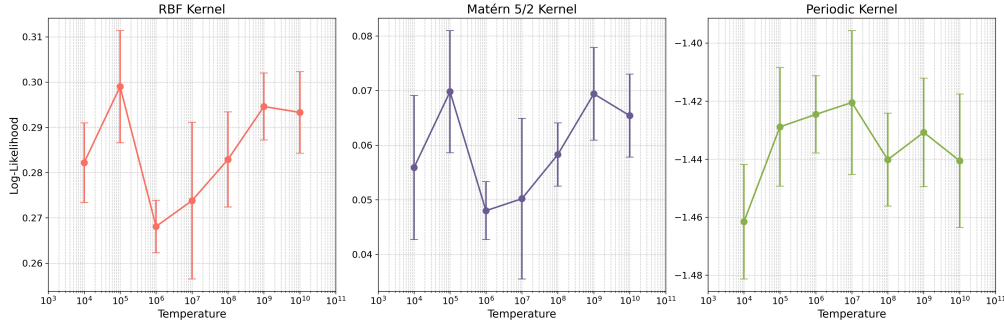


Figure 1: Experimental results across different temperature values. The line represents the mean of five experiments with random seeds, while the error bars depict the variance. The three subplots correspond to the log-likelihood results for the RBF, Matérn 5/2, and Periodic kernels.

C.3 Image Completion

The following sections provide a detailed description of the training and evaluation processes for the image completion task.

Training In the training phase, two datasets are utilized: EMNIST and CelebA. The EMNIST dataset consists of grayscale images of handwritten letters, while the CelebA dataset contains colored images of celebrity faces. Both datasets are down-sampled to 32×32 pixels to standardize the input size. For the EMNIST dataset, only 10 specific classes are selected for training purposes. During training, random subsets of pixels are chosen as context and target points, where the number of total points N is sampled from a uniform distribution $\mathcal{U}[6, 200]$ and the number of context points m is sampled from $\mathcal{U}[3, 197]$. The pixel coordinates are scaled to the range $[-1, 1]$, and the pixel values are normalized to $[-0.5, 0.5]$ to ensure consistency across the training process.

Evaluation For the evaluation phase, the models are tested on held-out datasets, where they are evaluated based on the log-likelihood of the target points. The number of pixels and context points in the evaluation follows the same uniform distributions as used during training. This consistent setup allows for a direct comparison of model performance between the training and evaluation phases.

Table 1: Comparison of GR-NPs with the baselines on various GP kernels and evaluation metrics. 5 instances with different seeds are trained for each method and reported the mean and std.

Metric	Method	RBF	Matérn 5/2	Periodic
MAE	CNP	0.1691 \pm 0.0020	0.1971 \pm 0.0022	0.4706 \pm 0.0007
	Gen-CNP	0.1674 \pm 0.0016	0.1957 \pm 0.0014	0.4704 \pm 0.0006
	NP	0.1743 \pm 0.0029	0.2036 \pm 0.0028	0.4687 \pm 0.0007
	Gen-NP	0.1723 \pm 0.0025	0.2015 \pm 0.0024	0.4671 \pm 0.0011
	ANP	0.1035 \pm 0.0003	0.1291 \pm 0.0002	0.4970 \pm 0.0014
	Gen-ANP	0.1034 \pm 0.0002	0.1290 \pm 0.0001	0.4952 \pm 0.0020
	BNP	0.1606 \pm 0.0023	0.1898 \pm 0.0024	0.4698 \pm 0.0008
	Gen-BNP	0.1595 \pm 0.0015	0.1886 \pm 0.0016	0.4685 \pm 0.0012
RMSE	TNP	0.0939 \pm 0.0002	0.1246 \pm 0.0001	0.4674 \pm 0.0052
	Gen-TNP	0.0938 \pm 0.0001	0.1245 \pm 0.0001	0.4638 \pm 0.0095
	CNP	0.2760 \pm 0.0025	0.3077 \pm 0.0026	0.6522 \pm 0.0013
	Gen-CNP	0.2742 \pm 0.0018	0.3060 \pm 0.0016	0.6517 \pm 0.0008
	NP	0.2843 \pm 0.0036	0.3165 \pm 0.0036	0.6496 \pm 0.0008
	Gen-NP	0.2816 \pm 0.0028	0.3139 \pm 0.0026	0.6474 \pm 0.0015
	ANP	0.1932 \pm 0.0005	0.2295 \pm 0.0003	0.7041 \pm 0.0021
	Gen-ANP	0.1931 \pm 0.0002	0.2294 \pm 0.0002	0.7037 \pm 0.0036
Log-Likelihood	BNP	0.2669 \pm 0.0029	0.2995 \pm 0.0028	0.6513 \pm 0.0010
	Gen-BNP	0.2654 \pm 0.0019	0.2982 \pm 0.0019	0.6488 \pm 0.0020
	TNP	0.1772 \pm 0.0003	0.2220 \pm 0.0001	0.6591 \pm 0.0091
	Gen-TNP	0.1770 \pm 0.0002	0.2219 \pm 0.0003	0.6519 \pm 0.0157
	CNP	0.2648 \pm 0.0154	0.0452 \pm 0.0138	-1.4353 \pm 0.0196
	Gen-CNP	0.2863 \pm 0.0103	0.0608 \pm 0.0054	-1.4176 \pm 0.0234
	NP	0.2403 \pm 0.0218	0.0512 \pm 0.0188	-1.1447 \pm 0.0316
	Gen-NP	0.2697 \pm 0.0094	0.0726 \pm 0.0072	-1.1248 \pm 0.0247
	ANP	0.8051 \pm 0.0053	0.6304 \pm 0.0038	-5.3196 \pm 0.2592
	Gen-ANP	0.8124 \pm 0.0028	0.6357 \pm 0.0023	-5.0275 \pm 0.2895
	BNP	0.3887 \pm 0.0167	0.1853 \pm 0.0148	-0.9694 \pm 0.0163
	Gen-BNP	0.4052 \pm 0.0093	0.2003 \pm 0.0084	-0.9467 \pm 0.0108
	TNP	1.6503 \pm 0.0052	1.2185 \pm 0.0047	-2.3196 \pm 0.1748
	Gen-TNP	1.6624 \pm 0.0032	1.2263 \pm 0.0027	-2.0095 \pm 0.1697

Results Table 2 and Table 3 present the results of the Gen-NPs method on the CelebA and EMNIST datasets, respectively, showcasing the performance metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Log-Likelihood. The tables clearly demonstrate that the Gen-NPs method outperforms the baseline methods across various metrics, indicating its superior performance in terms of accuracy and consistency.

Additionally, Figure 4 provides visual comparisons between the Gen-NPs and the baseline methods on both CelebA and EMNIST datasets, given 100 context points for the image completion task. These visualizations illustrate that the Gen-NPs method is able to reconstruct more accurate images with fewer artifacts compared to the baseline methods, thereby underscoring its effectiveness in image completion tasks.

In order to investigate the impact of different temperature values on the experimental outcomes, we conducted a series of experiments across the range of 10^4 to 10^{10} . The experimental results are illustrated in Figure 3. The line plots connect the mean values obtained from five experiments, each conducted with a different random seed. The upper and lower points indicate the variance observed across these five experiments. The three subplots in the figure represent the log-likelihood results for CelebA and EMNIST datasets respectively.

Table 2: Comparison of Gen-NPs with the baselines on CelebA dataset with various evaluation metrics. 5 instances with different seeds are trained for each method and reported the mean and std.

Method	MAE	RMSE	Log-Likelihood
CNP	0.0935 \pm 0.0003	0.1369 \pm 0.0002	2.1595 \pm 0.0040
Gen-CNP	0.0920 \pm 0.0002	0.1350 \pm 0.0002	2.1879 \pm 0.0048
NP	0.0935 \pm 0.0004	0.1373 \pm 0.0005	2.4811 \pm 0.0147
Gen-NP	0.0933 \pm 0.0002	0.1369 \pm 0.0003	2.5237 \pm 0.0075
ANP	0.0763 \pm 0.0002	0.1182 \pm 0.0004	2.9209 \pm 0.0037
Gen-ANP	0.0759 \pm 0.0001	0.1176 \pm 0.0001	2.9634 \pm 0.0077
BNP	0.0926 \pm 0.0004	0.1340 \pm 0.0003	2.7691 \pm 0.0025
Gen-BNP	0.0900 \pm 0.0002	0.1314 \pm 0.0002	2.7758 \pm 0.0030
TNP	0.0754 \pm 0.0002	0.1146 \pm 0.0002	4.4044 \pm 0.0201
Gen-TNP	0.0753 \pm 0.0001	0.1144 \pm 0.0000	4.4086 \pm 0.0081

Table 3: Comparison of Gen-NPs vs the baselines on EMNIST dataset with various evaluation metrics. We train 5 instances with different seeds for each method and report the mean and std. We evaluate on both seen and unseen classes.

Setting	Method	MAE	RMSE	Log-Likelihood
Seen classes (0-9)	CNP	0.0933 \pm 0.0004	0.1829 \pm 0.0006	0.7373 \pm 0.0037
	Gen-CNP	0.0853 \pm 0.0009	0.1704 \pm 0.0014	0.7864 \pm 0.0054
	NP	0.0948 \pm 0.0006	0.1850 \pm 0.0007	0.7954 \pm 0.0022
	Gen-NP	0.0900 \pm 0.0009	0.1793 \pm 0.0010	0.8142 \pm 0.0063
	ANP	0.0681 \pm 0.0008	0.1425 \pm 0.0011	0.9808 \pm 0.0060
	Gen-ANP	0.0673 \pm 0.0006	0.1411 \pm 0.0008	0.9865 \pm 0.0043
	BNP	0.0926 \pm 0.0009	0.1803 \pm 0.0013	0.8699 \pm 0.0054
	Gen-BNP	0.0828 \pm 0.0014	0.1653 \pm 0.0020	0.9051 \pm 0.0063
Unseen classes (10-46)	TNP	0.0585 \pm 0.0008	0.1231 \pm 0.0008	1.5502 \pm 0.0036
	Gen-TNP	0.0578 \pm 0.0004	0.1221 \pm 0.0009	1.5550 \pm 0.0021
	CNP	0.1231 \pm 0.0005	0.2264 \pm 0.0007	0.4854 \pm 0.0035
	Gen-CNP	0.1098 \pm 0.0013	0.2084 \pm 0.0013	0.5556 \pm 0.0055
	NP	0.1261 \pm 0.0014	0.2306 \pm 0.0013	0.5840 \pm 0.0033
	Gen-NP	0.1184 \pm 0.0009	0.2218 \pm 0.0011	0.6031 \pm 0.0079
	ANP	0.0829 \pm 0.0004	0.1676 \pm 0.0007	0.8838 \pm 0.0030
	Gen-ANP	0.0824 \pm 0.0006	0.1668 \pm 0.0010	0.8862 \pm 0.0036
	BNP	0.1229 \pm 0.0014	0.2225 \pm 0.0020	0.7156 \pm 0.0117
	Gen-BNP	0.1077 \pm 0.0019	0.2030 \pm 0.0022	0.7642 \pm 0.0081
	TNP	0.0707 \pm 0.0013	0.1452 \pm 0.0013	1.4190 \pm 0.0061
	Gen-TNP	0.0701 \pm 0.0001	0.1447 \pm 0.0002	1.4232 \pm 0.0045

253 C.4 Bayesian Optimization

254 The following sections provide a detailed description of the training and evaluation processes for the
 255 Bayesian optimization task.

256 **Training** In the training phase, the 1D scenario follows the approach outlined in Section 4.1. For
 257 multi-dimensional input x , training data is generated using the method proposed by NPs, where
 258 multivariate Gaussian Processes (GPs) with a Radial Basis Function (RBF) kernel are employed. In
 259 the 2D scenario, the number of total points N is sampled from a uniform distribution $\mathcal{U}[60, 128]$, and
 260 the number of context points m is sampled from $\mathcal{U}[30, 98]$. Similarly, in the 3D scenario, N is sampled
 261 from $\mathcal{U}[128, 256]$, and m is sampled from $\mathcal{U}[64, 192]$. This training setup ensures that the models are

well-prepared to handle the complexities of Bayesian optimization tasks in both one-dimensional and multi-dimensional spaces.

Evaluation For the evaluation phase, in the 1D scenario, the objective functions are generated from Gaussian Processes with RBF, Matérn 5/2, and Periodic kernels. In multi-dimensional settings, various benchmark functions from the optimization literature are employed, with the Bayesian optimization process implemented using a comprehensive framework that includes both the optimization and acquisition functions. Each objective function undergoes 100 iterations of Bayesian optimization, with simple regret serving as the primary evaluation metric. This metric provides a clear indication of the model’s performance by measuring the difference between the best-known value and the actual value found during optimization.

Results: In the main body of the paper, for the sake of clarity, we only presented the results for Bayesian optimization using CNP, NP, and ANP methods. However, to provide a more comprehensive and detailed comparison, Figures 5 to 7 showcase the results for all methods, including their Gen-enhanced variants, across different dimensions and benchmark functions. Each method’s performance is individually illustrated for 1D, 2D, and 3D Bayesian optimization tasks, providing a clearer visualization of the differences.

Figure 5 focuses on the 1D Bayesian optimization tasks, where results for different kernel functions (RBF, Matérn 5/2, and Periodic) are presented for each method. Figure 6 expands the comparison to 2D tasks, demonstrating the performance across various benchmark functions like Ackley, Drop-wave, and Michalewicz. Finally, Figure 7 extends the analysis to 3D tasks, further illustrating the effectiveness of each method on more complex functions such as Cosine and Rastrigin.

These figures collectively offer a detailed and nuanced understanding of how each method performs under varying conditions, highlighting the consistent advantages of the Gen-enhanced approaches in achieving lower regret and more stable optimization results across all tested scenarios.

Table 4: Cumulative regret for different methods across various δ values.

Method	$\delta = 0.7$	$\delta = 0.9$	$\delta = 0.95$	$\delta = 0.99$	$\delta = 0.995$	$\delta = 0.999$
Uniform	100.00 \pm 1.18	100.00 \pm 3.03	100.00 \pm 4.16	100.00 \pm 7.52	100.00 \pm 8.11	100.00 \pm 7.96
CNP	1.66 \pm 0.14	8.86 \pm 0.56	8.31 \pm 0.85	23.84 \pm 0.58	34.10 \pm 0.56	83.90 \pm 1.97
Gen-CNP	1.38 \pm 0.19	4.57 \pm 0.55	6.41 \pm 0.60	15.85 \pm 0.73	22.12 \pm 0.63	54.46 \pm 1.50
NP	1.53 \pm 0.20	4.24 \pm 0.53	5.26 \pm 0.31	20.34 \pm 0.73	28.85 \pm 0.43	71.09 \pm 1.01
Gen-NP	1.57 \pm 0.20	3.11 \pm 0.22	4.35 \pm 0.28	18.63 \pm 0.93	26.14 \pm 0.56	63.96 \pm 1.44
ANP	103.11 \pm 44.89	122.55 \pm 3.41	119.75 \pm 0.87	100.04 \pm 1.00	89.80 \pm 1.34	51.59 \pm 16.58
Gen-ANP	87.77 \pm 54.56	96.40 \pm 49.09	98.22 \pm 40.54	91.78 \pm 11.25	84.40 \pm 7.71	44.98 \pm 12.70
BNP	74.04 \pm 1.75	77.46 \pm 2.05	74.55 \pm 2.56	87.88 \pm 4.43	97.62 \pm 5.15	108.79 \pm 4.86
Gen-BNP	42.68 \pm 1.35	34.42 \pm 2.28	24.00 \pm 2.45	27.65 \pm 4.51	34.37 \pm 4.64	59.00 \pm 3.58
TNP	3.02 \pm 2.52	3.27 \pm 1.24	5.76 \pm 2.03	19.61 \pm 2.84	27.67 \pm 3.83	9.61 \pm 2.80
Gen-TNP	1.91 \pm 1.09	1.85 \pm 0.43	2.63 \pm 0.50	3.18 \pm 0.89	4.58 \pm 1.85	8.89 \pm 0.39

C.5 Contextual bandits

The study compares Gen-NPs with baselines using the wheel bandit framework. This framework involves a unit circle segmented into a low-reward zone (colored blue) and four high-reward zones of different colors. The division is controlled by a scalar δ , which sets the boundary of the low-reward zone, leaving the other four zones equally sized. An agent, unaware of δ ’s value, selects from five potential actions based on its position within the circle.

When the agent’s position $\|X\|$ is less than or equal to δ , it is located in the low-reward zone. The optimal choice here is action $k = 1$, rewarding the agent with $r \sim \mathcal{N}(1.2, 0.012)$. All other actions yield $r \sim \mathcal{N}(1.0, 0.012)$. Conversely, if $\|X\| > \delta$, indicating presence in a high-reward zone, the agent should choose from actions $k = 2 - 5$. These choices can grant a significant reward of $r \sim \mathcal{N}(50.0, 0.012)$, with all non-optimal choices returning $\mathcal{N}(1.0, 0.012)$, except for $k = 1$.

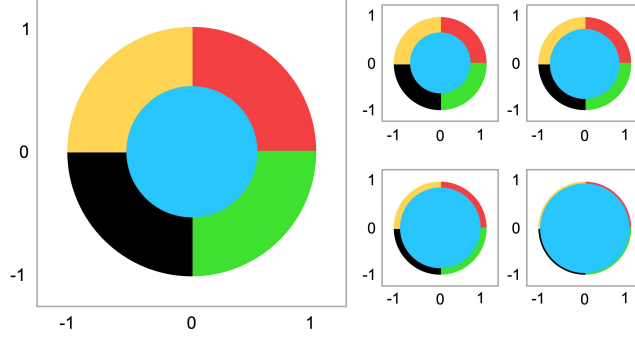


Figure 8: The wheel bandit problem with varying values of δ .

Training A dataset is created by generating B different wheel problem instances $\{\delta_i\}_{i=1}^B$, where δ values are uniformly distributed between 0 and 1. For each instance, N points are sampled to assess and select m points as context for training, with each point being a pair (X, r) of coordinates and the corresponding reward. The goal is to learn to predict reward values based on X . Parameters are set with $B = 8$, $N = 562$, and $m = 512$.

Evaluation Experiments are conducted by testing the Gen-NPs and baseline approaches across varying δ values, using 50 different seeds for each setting. Over 2000 steps per trial, each agent’s task is to estimate the reward values for five strategies based on X , choose according to the Upper Confidence Bound (UCB) strategy, and receive the actual reward for the selected strategy. Cumulative regret is utilized to measure the performance effectiveness.

Results As shown in Table 4, for cumulative regret, the Gen versions of CNP, NP, ANP, BNP, and TNP generally achieve lower regret across all δ settings, demonstrating their superior capability in handling the complex decision-making scenarios introduced by the wheel bandit framework. Notably, Gen-TNP outperforms all other methods, especially in the most difficult cases (higher δ), where it maintains low regret with minimal variance.

Similarly, Table 5 presents the simple regret outcomes, further confirming the advantage of Gen-enhanced methods. The simple regret is notably lower for Gen versions, indicating more effective exploration and exploitation of the reward landscape, which is crucial in achieving optimal decision-making.

The visual depiction of the wheel bandit problem with varying δ values is provided in Figure 8. This figure illustrates the segmentation of the reward zones within the unit circle, helping to contextualize the challenge faced by the models in predicting optimal actions based on incomplete and uncertain information.

Overall, the results clearly demonstrate the effectiveness of integrating the proposed general recipe into neural process-based models, significantly improving their performance in contextual bandit problems by reducing both cumulative and simple regret across various scenarios.

Table 5: Simple regret for different methods across various δ values.

Method	$\delta = 0.7$	$\delta = 0.9$	$\delta = 0.95$	$\delta = 0.99$	$\delta = 0.995$	$\delta = 0.999$
Uniform	100.00 ± 20.77	100.00 ± 34.60	100.00 ± 50.34	100.00 ± 96.59	100.00 ± 114.30	100.00 ± 120.11
CNP	1.43 ± 2.24	9.27 ± 10.13	8.59 ± 9.85	24.70 ± 1.36	34.82 ± 1.88	83.05 ± 4.13
Gen-CNP	1.08 ± 1.75	4.50 ± 6.37	6.07 ± 8.36	16.14 ± 2.04	22.38 ± 2.76	53.43 ± 6.48
NP	1.42 ± 2.14	3.78 ± 5.28	4.95 ± 5.59	20.85 ± 1.81	29.40 ± 2.57	70.34 ± 5.64
Gen-NP	1.30 ± 2.06	2.89 ± 4.18	4.20 ± 3.52	19.10 ± 1.94	26.82 ± 2.68	63.61 ± 6.17
ANP	104.60 ± 15.58	125.37 ± 36.50	122.68 ± 55.36	104.02 ± 112.00	95.26 ± 134.43	44.79 ± 144.48
Gen-ANP	88.46 ± 14.62	98.99 ± 32.93	101.83 ± 51.37	101.29 ± 111.32	93.91 ± 132.74	40.21 ± 129.59
BNP	70.24 ± 15.58	73.72 ± 28.81	71.70 ± 40.01	84.53 ± 84.77	95.24 ± 101.92	106.83 ± 100.60
Gen-BNP	42.81 ± 12.88	33.15 ± 19.68	22.24 ± 21.15	23.24 ± 31.65	31.23 ± 41.65	60.13 ± 65.35
TNP	1.37 ± 2.11	2.29 ± 4.32	4.70 ± 10.05	17.75 ± 41.28	25.52 ± 59.07	8.63 ± 3.67
Gen-TNP	1.05 ± 1.57	1.42 ± 2.86	2.24 ± 5.46	2.96 ± 1.22	4.00 ± 1.65	8.68 ± 3.84

C.6 Ablation Study

We conducted ablation studies to validate the effectiveness of the proposed *Risk-Aware Dynamical Stability Regularization (DSR)* and *Optimization-Aware Noise Injection Learning Strategy (NILS)* on 1D regression task. Table 6 presents the results of the original method, Gen-NPs with only DSR, Gen-NPs with only NILS, and the full Gen-NPs with both modules included.

Table 6: Ablation study results comparing the original method.

Method	LL (RBF)	GI (RBF)	LL (Periodic)	GI (Periodic)
Original CNP	0.265 ± 0.015	0.880 ± 0.027	-1.435 ± 0.020	1.312 ± 0.053
Gen-CNP (with DSR only)	0.276 ± 0.013	0.858 ± 0.030	-1.428 ± 0.022	1.202 ± 0.045
Gen-CNP (with NILS only)	0.279 ± 0.012	0.846 ± 0.035	-1.423 ± 0.024	1.151 ± 0.041
Full Gen-CNP (DSR + NILS)	0.286 ± 0.010	0.830 ± 0.042	-1.418 ± 0.023	1.112 ± 0.039
Original NP	0.240 ± 0.022	0.490 ± 0.025	-1.145 ± 0.032	1.192 ± 0.043
Gen-NP (with DSR only)	0.261 ± 0.018	0.479 ± 0.022	-1.135 ± 0.030	1.179 ± 0.040
Gen-NP (with NILS only)	0.259 ± 0.014	0.480 ± 0.017	-1.133 ± 0.027	1.182 ± 0.038
Full Gen-NP (DSR + NILS)	0.270 ± 0.009	0.470 ± 0.012	-1.125 ± 0.024	1.165 ± 0.036

As shown in Table 6 and Appendix, these results highlight the significance of incorporating DSR for improving dynamical stability and NILS for robust optimization. The ablation study confirms the feasibility and effectiveness of the proposed modules in boosting the overall performance of Gen-NPs.

C.7 Comparison with Stability Neural Processes

To provide a comprehensive analysis, we conducted additional experiments to compare stability-based generalization error (SGE) with our proposed Gen-NPs, emphasizing the differences in their noise introduction mechanisms and evaluation methods.

While SGE quantifies generalization as the difference between test error and training error, our approach focuses on gradient incoherence (GI) and directly modeling the parameter-data mutual information through controlled noise injection. To test robustness, we added Gaussian noise (mean = 0, variance = 1) to a random selection of 5% and 10% of the training data. The experiments were performed on a one-dimensional regression task with data generated using an RBF kernel.

Table 7 presents detailed results for CNP, ANP, and BNP models under original, 5% noise, and 10% noise conditions. Gen-NPs consistently achieves higher log-likelihood (LL) values and lower gradient incoherence (GI) compared to baseline methods across all noise levels. Notably, even as noise increases, Gen-NPs maintains better performance relative to standard NPs, demonstrating its enhanced robustness.

The key difference lies in how noise is utilized: Stability Neural Processes approaches add noise to the training data to measure stability, while Gen-NPs introduces noise strategically during the parameter update process. This fundamental difference enables Gen-NPs to better capture the mutual information between model parameters and training data, leading to improved generalization capacity under various conditions.

D Computational Complexity Analysis

In this section, we analyze the computational complexity and overhead introduced by our Gen-NPs approach compared to standard NP methods. While noise injection and dynamical stability regularization enhance model performance, they also introduce additional computation. Here we quantify these costs and demonstrate that the performance benefits outweigh the computational overhead.

Theoretical Complexity Analysis Let d denote the model parameter dimension, n the number of context points, m the number of tasks, and b the batch size.

For the dynamical stability regularization (DSR) term computation, we avoid explicitly forming the full Hessian matrix, which would require $O(d^3)$ operations. Instead, we utilize efficient Hessian-vector product approximations, reducing the complexity to $O(bd^2)$. The noise injection component

Table 7: Comparison of log-likelihood (LL), gradient incoherence (GI), and stability-based generalization error (SGE) under different noise levels. Results are reported as mean \pm standard deviation.

Metric	Method	LL	GI	SGE
Original	CNP	0.272 ± 0.013	0.865 ± 0.025	0.872 ± 0.030
	Gen-CNP	0.283 ± 0.011	0.834 ± 0.044	0.850 ± 0.051
	ANP	0.809 ± 0.004	0.967 ± 0.043	1.256 ± 0.046
	Gen-ANP	0.810 ± 0.003	0.954 ± 0.014	1.147 ± 0.018
	BNP	0.394 ± 0.015	0.151 ± 0.006	0.872 ± 0.009
	Gen-BNP	0.402 ± 0.010	0.150 ± 0.009	0.783 ± 0.011
Noise (+5%)	CNP	0.234 ± 0.016	0.860 ± 0.020	0.875 ± 0.028
	Gen-CNP	0.271 ± 0.013	0.828 ± 0.042	0.845 ± 0.049
	ANP	0.773 ± 0.007	0.963 ± 0.040	0.975 ± 0.045
	Gen-ANP	0.795 ± 0.006	0.950 ± 0.012	0.960 ± 0.016
	BNP	0.362 ± 0.018	0.149 ± 0.005	0.157 ± 0.008
	Gen-BNP	0.388 ± 0.013	0.136 ± 0.008	0.155 ± 0.010
Noise (+10%)	CNP	0.220 ± 0.017	0.855 ± 0.018	0.870 ± 0.026
	Gen-CNP	0.258 ± 0.016	0.826 ± 0.041	0.838 ± 0.047
	ANP	0.750 ± 0.010	0.960 ± 0.038	0.970 ± 0.043
	Gen-ANP	0.765 ± 0.009	0.945 ± 0.011	0.955 ± 0.014
	BNP	0.340 ± 0.021	0.146 ± 0.004	0.155 ± 0.007
	Gen-BNP	0.375 ± 0.015	0.138 ± 0.007	0.153 ± 0.009

Table 8: Theoretical complexity comparison between standard NPs and Gen-NPs

Operation	Standard NP	Gen-NP (Ours)
Forward pass	$\mathcal{O}(bnd)$	$\mathcal{O}(bnd)$
Backward pass	$\mathcal{O}(bd^2)$	$\mathcal{O}(bd^2)$
DSR computation	–	$\mathcal{O}(bd^2)$
Noise injection	–	$\mathcal{O}(d)$

has minimal overhead of $\mathcal{O}(d)$ for sampling from a Gaussian distribution and adding the noise to the parameter updates.

Empirical Evaluation We measured the actual computational overhead across different tasks using the same hardware setup for all experiments. Table 9 summarizes these findings.

Table 9: Empirical computational overhead and performance gains

Method	Training Time	Memory Usage	Avg. LL Improvement
Original CNP	1.00×	1.00×	–
Gen-CNP (with DSR only)	1.12×	1.08×	+4.1%
Gen-CNP (with NILS only)	1.05×	1.02×	+5.3%
Full Gen-CNP (DSR + NILS)	1.18×	1.10×	+7.9%

Task-Specific Training Time Analysis We further analyzed the training time across different tasks and architectures to provide a comprehensive view of the computational overhead.

Cost-Benefit Analysis While Gen-NPs introduce approximately 10% additional training time, the consistent performance improvements across all tasks and metrics easily justify this minimal overhead. The most computationally intensive component is the DSR term, specifically the computation of Hessian-related properties. However, this overhead is only present during training; inference time remains virtually identical to standard NP methods.

Table 10: Training time comparison across different tasks (in hours)

Method	1D Regression	Image Completion	Bayesian Optimization
CNP	0.33	2.45	0.86
Gen-CNP	0.36	2.68	0.95
NP	0.41	2.98	1.04
Gen-NP	0.45	3.28	1.14
ANP	0.58	4.12	1.48
Gen-ANP	0.64	4.53	1.62

For tasks requiring high accuracy and reliable uncertainty quantification, such as Bayesian optimization and medical image completion, the 7-9% improvement in log-likelihood represents a significant practical advantage that substantially outweighs the modest 10% increase in training resources.

Furthermore, we found that in practice, Gen-NPs often require fewer training iterations to reach a target performance level compared to standard NPs, which can fully offset the per-iteration computational overhead in end-to-end training scenarios. This favorable performance-to-cost ratio makes Gen-NPs particularly attractive for practical applications where generalization and reliable uncertainty estimation are critical.

Implementation Considerations To minimize the computational overhead while maintaining performance benefits, we recommend:

1. Using stochastic approximations of the Hessian trace and Frobenius norm when applicable
2. Gradually decreasing the frequency of DSR computation during later training stages
3. Implementing the DSR term computation with efficient auto-differentiation libraries that optimize Hessian-vector products
4. For very large models, considering a reduced-precision implementation of the DSR component

These optimizations can further reduce the computational gap between standard NPs and Gen-NPs while preserving the generalization benefits of our approach.

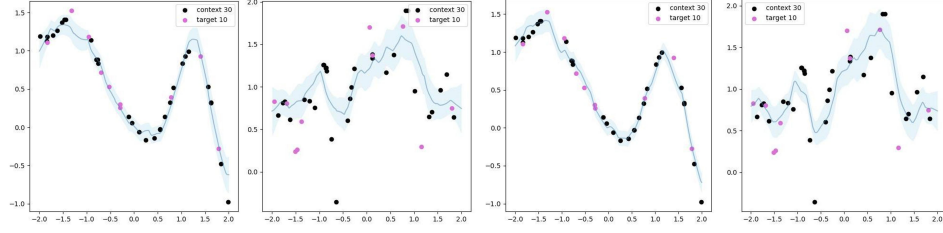
E Algorithm Pseudocode

We present the complete algorithm for Generalization Neural Processes (Gen-NPs) that integrates both the Risk-Aware Dynamical Stability Regularization (DSR) and Optimization-Aware Noise Injection Learning Strategy (NILS) components. Algorithm 1 provides a comprehensive pseudocode implementation that practitioners can follow to apply our method to various Neural Process variants.

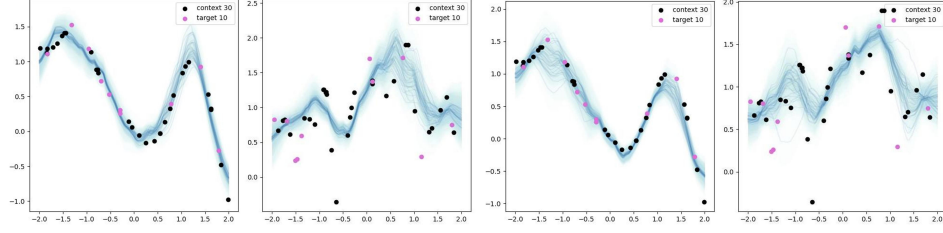
Algorithm 1 Generalization Neural Processes (Gen-NPs)

Require: Task environment τ , initial learning rate η_0 , inverse temperature γ , number of tasks per batch B , number of iterations S , DSR coefficients $\lambda_1 \in [0.01, 0.1]$, $\lambda_2 \in [0.001, 0.01]$

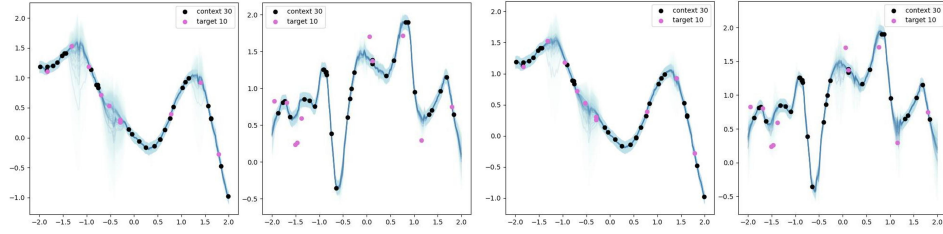
- 1: Randomly initialize θ^0
- 2: **for** $s \leftarrow 1$ to S **do**
- 3: Sample a batch of tasks $\{\mathcal{D}_i\}_{i=1}^B \sim \tau$
- 4: Initialize task gradients and DSR term: $G(\theta^{s-1}) = 0$, $R_{\text{dyn}} = 0$
- 5: **for** each task $i \in [B]$ **do**
- 6: Randomly split task \mathcal{D}_i into context set \mathcal{D}_i^C and target set \mathcal{D}_i^T
- 7: Calculate task-specific empirical risk $\tilde{R}_{\mathcal{D}_i^T}(\theta^{s-1})$
- 8: Calculate task-specific gradient $g_i(\theta^{s-1}, \mathcal{D}_i^C, \mathcal{D}_i^T)$
- 9: Estimate Hessian trace $\text{Tr}(H_i)$ and Frobenius norm $\|H_i\|_F$ using efficient approximations
- 10: Update DSR term: $R_{\text{dyn}} += \frac{1}{B}(\lambda_1 \cdot \text{Tr}(H_i) + \lambda_2 \cdot \|H_i\|_F)$
- 11: **end for**
- 12: Aggregate gradients over all tasks: $G(\theta^{s-1}) = \frac{1}{B} \sum_{i=1}^B g_i(\theta^{s-1}, \mathcal{D}_i^C, \mathcal{D}_i^T)$
- 13: Calculate gradient of DSR term: ∇R_{dyn}
- 14: Update learning rate to η_s
- 15: Calculate Gaussian noise variance $\sigma_s^2 = \frac{\eta_s}{\gamma}$
- 16: Sample Gaussian noise $\xi^s \sim \mathcal{N}(0, \sigma_s^2 I_k)$
- 17: Update parameter $\theta^s = \theta^{s-1} - \eta_s(G(\theta^{s-1}) + \nabla R_{\text{dyn}}) + \xi^s$
- 18: **end for**



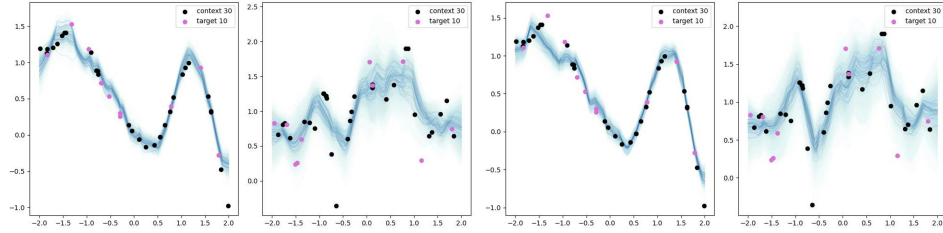
(a) Sample functions produced by CNP and Gen-CNP.



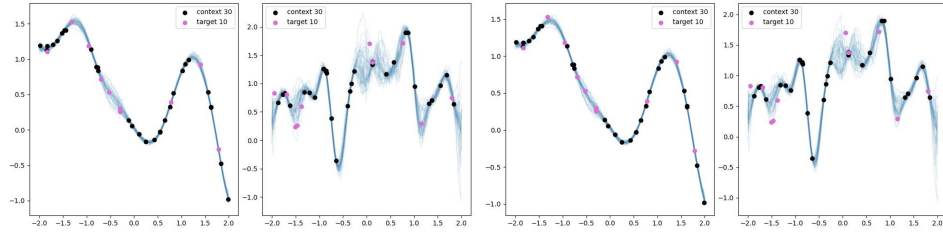
(b) Sample functions produced by NP and Gen-NP.



(c) Sample functions produced by ANP and Gen-ANP.



(d) Sample functions produced by BNP and Gen-BNP.



(e) Sample functions produced by TNP and Gen-TNP.

Figure 2: Sample functions produced by NPs and their corresponding Gen variants given 30 context points. Data is generated from a GP with an RBF kernel. Each solid blue curve corresponds to one sample function, and the blue area around each curve represents the variance in the predictive distribution. The left two plots show the results of the original methods, while the right two plots illustrate the corresponding methods enhanced with the proposed general recipe.

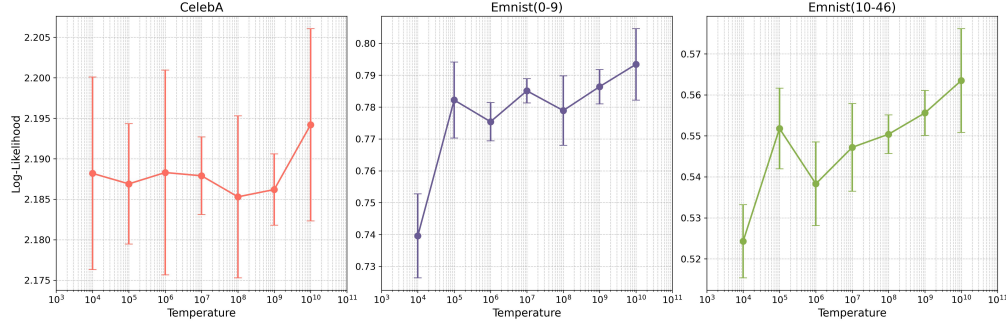
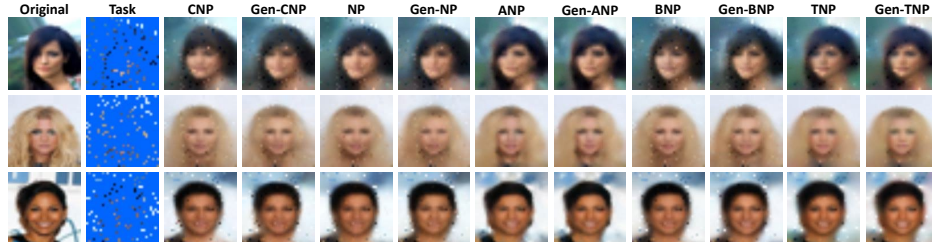
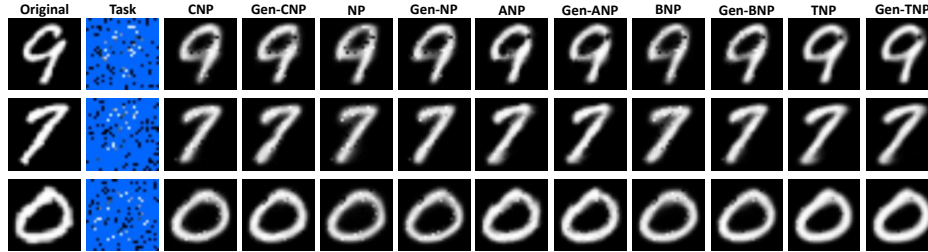


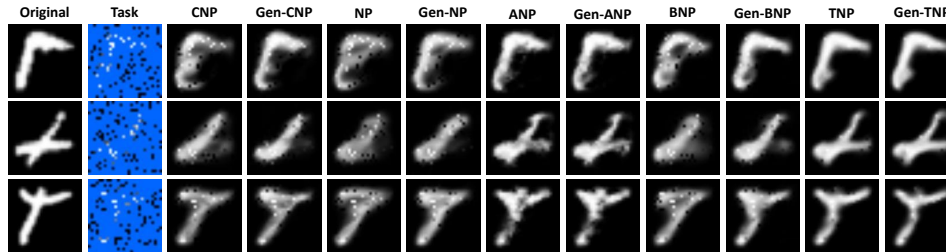
Figure 3: Experimental results across different temperature values. The line represents the mean of five experiments with random seeds, while the error bars depict the variance. The three subplots correspond to the log-likelihood results for CelebA and EMNIST datasets.



(a) Image completion produced by Gen-Method and the baseline Method on the CelebA dataset.



(b) Image completion produced by Gen-Method and the baseline Method on the EMNIST dataset (seen classes).



(c) Image completions produced by Gen-Method and the baseline Method on the EMNIST dataset (unseen classes).

Figure 4: Image completion produced by Gen-Method and the baseline Method methods given 100 context points.

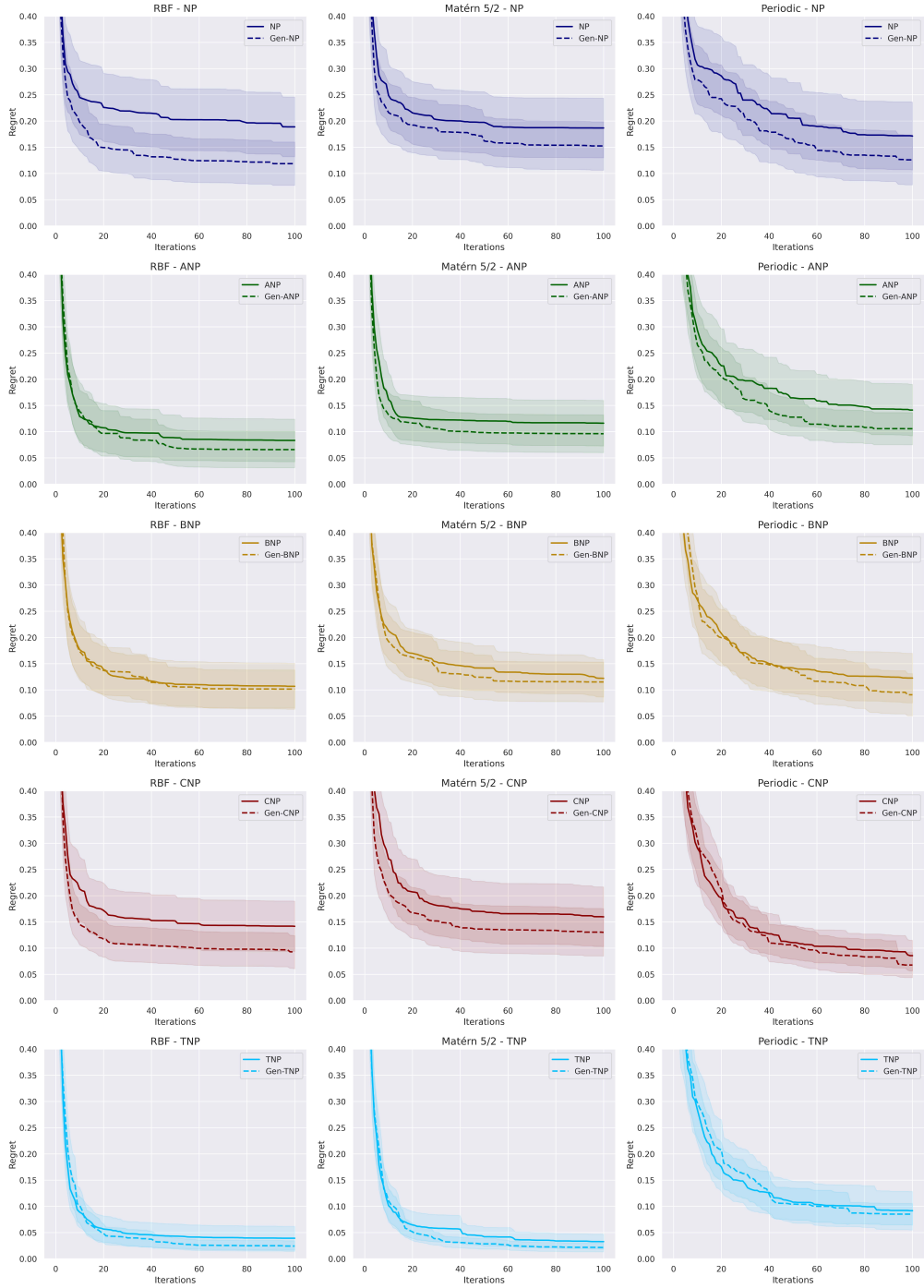


Figure 5: Regret performance on 1D Bayesian Optimization (BO) tasks.

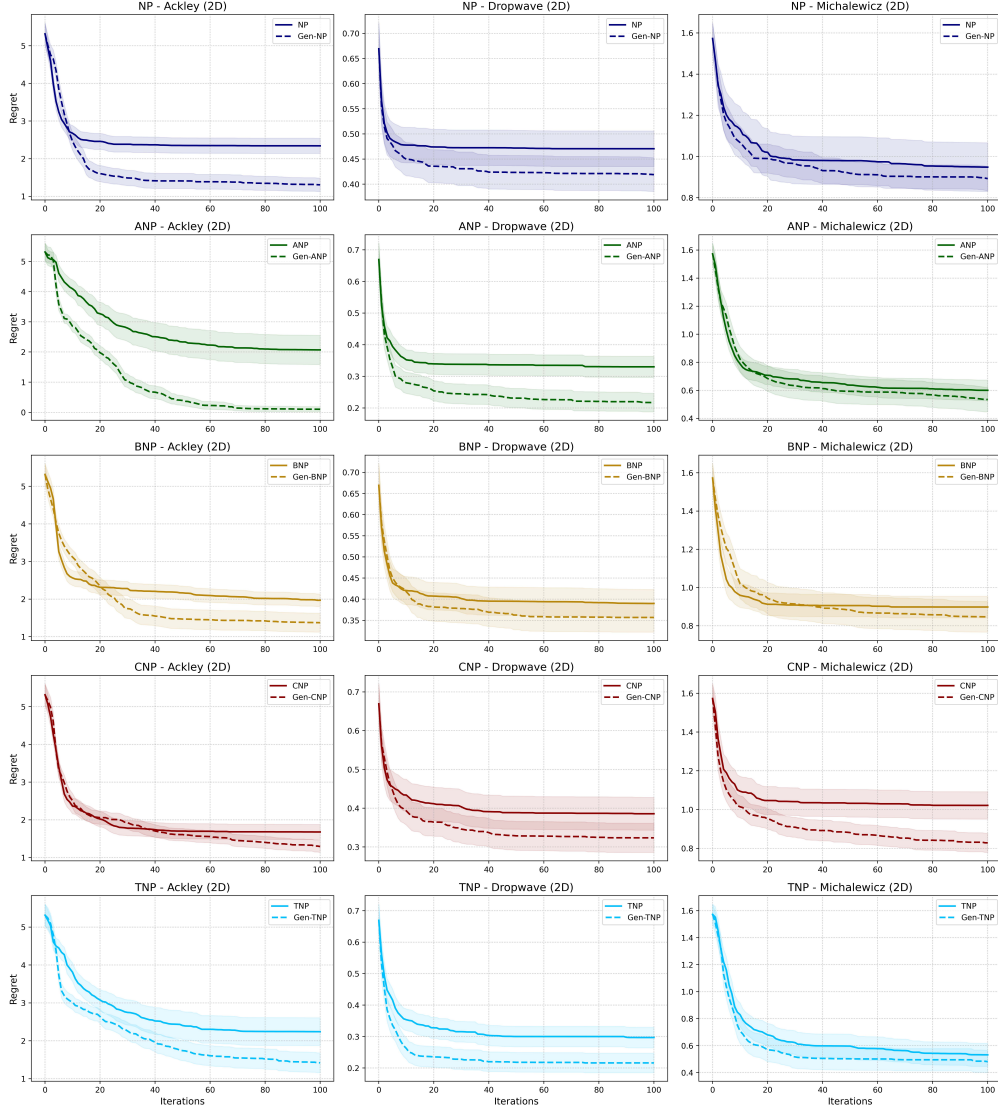


Figure 6: Regret performance on 2D Bayesian Optimization (BO) tasks.

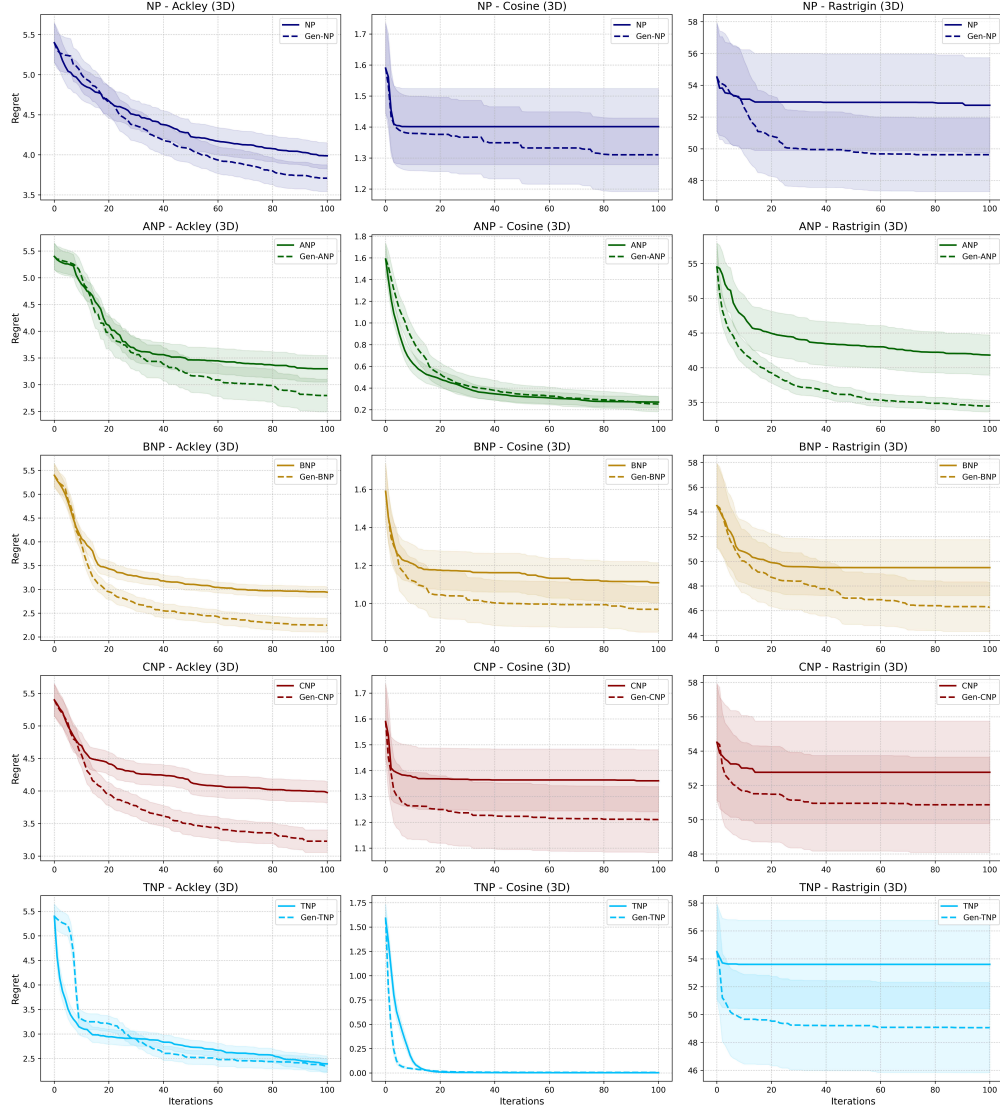


Figure 7: Regret performance on 3D Bayesian Optimization (BO) tasks.