Streaming Generated Gaussian Process Experts for Online Learning and Control: Extended Version

Zewen Yang[1], Dongfa Zhang[1], Xiaobing Dai[2], Fengyi Yu[1], Chi Zhang[3], Bingkun Huang[1], Hamid Sadeghian[1], Sami Haddadin[4]

[1]Chair of Robotics and Systems Intelligence, Munich Institute of Robotics and Machine Intelligence, Technical University of Munich

[2] Chair of Information-oriented Control, School of Computation, Information and Technology, Technical University of Munich

[3]Department of Informatics, School of Computation, Information and Technology, Technical University of Munich

[4]Mohamed bin Zayed University of Artificial Intelligence

zewen.yang@tum.de

arXiv:2508.03679v3 [cs.LG] 14 Nov 2025

Abstract

Gaussian Processes (GPs), as a nonparametric learning method, offer flexible modeling capabilities and calibrated uncertainty quantification for function approximations. Additionally, GPs support online learning by efficiently incorporating new data with polynomial-time computation, making them well-suited for safety-critical dynamical systems that require rapid adaptation. However, the inference and online updates of exact GPs, when processing streaming data, incur cubic computation time and quadratic storage memory complexity, limiting their scalability to large datasets in real-time settings. In this paper, we propose a streaming kernel-induced progressively generated expert framework of Gaussian processes (SkyGP) that addresses both computational and memory constraints by maintaining a bounded set of experts, while inheriting the learning performance guarantees from exact Gaussian processes. Furthermore, two SkyGP variants are introduced, each tailored to a specific objective, either maximizing prediction accuracy (SkyGP-Dense) or improving computational efficiency (SkyGP-Fast). The effectiveness of SkyGP is validated through extensive benchmarks and real-time control experiments demonstrating its superior performance compared to state-of-the-art approaches.

Code — https://github.com/Zewen-Yang/SkyGP

1 Introduction

Real-time learning has become essential for modeling the dynamics of physical systems, where machine learning models must be continuously updated to adapt to changing environments while satisfying the requirements of responsiveness and safety. This capability is especially critical for autonomous systems, such as underwater vehicles, aerial drones, and healthcare robots, that operate in complex and safety-critical environments (Yang et al. 2025). In such scenarios, the ability to learn accurate models online and seamlessly integrate them into control loops is key to ensuring robust and efficient operation (Dai et al. 2023).

Gaussian Processes (GPs) provide a powerful nonparametric framework for modeling dynamical systems, particularly in safety-critical applications, due to their ability to quantify uncertainty (Wachi et al. 2018). However, despite their modeling flexibility, standard GP inference suffers from severe scalability limitations. It scales cubically with the number of data points, requiring $\mathcal{O}(N^3)$ time and $\mathcal{O}(N^2)$ memory for N training data. This scalability bottleneck makes conventional GPs impractical for online applications involving continuously streaming data or long-term deployments.

To mitigate these limitations, a variety of scalable approximation techniques have been developed. Among global approximations, sparse Gaussian Processes introduce a set of $M \ll N$ inducing points to efficiently summarize the training data, reducing the complexity to $\mathcal{O}(NM^2)$ for training and $\mathcal{O}(M^2)$ for prediction. Notable methods include fully independent training conditional approximation (Snelson and Ghahramani 2005), variational free energy (Naish-guzman and Holden 2007), and latent projection techniques (Reeb et al. 2018). While streaming variants such as incremental sparse spectrum GP (ISSGP) (Gijsberts and Metta 2013) and streaming sparse GP (SSGP) (Bui, Nguyen, and Turner 2017) allow for incremental online updates, these frequently require computationally expensive optimization at each update, which can undermine their practical use in latency-sensitive settings. Moreover, these methods typically forgo guarantees on prediction error bounds, sacrificing reliability in critical applications.

Distributed Gaussian Processes (DGPs) provide complementary scalability by partitioning data across multiple processors or agents for parallel processing. Within this paradigm, mixture-of-experts (MoE) approaches (Tresp 2000b; Yuan and Neubauer 2008; Trapp et al. 2020) aggregate predictions from independently trained GP experts using fixed or learned weights. Extensions such as product-of-experts (PoE) (Cohen et al. 2020), generalized PoE (gPoE) (Cao and Fleet 2015), and correlated PoE with sparse GPs (Schürch et al. 2023) have been proposed to better utilize uncertainty in aggregation. The Bayesian committee machine (BCM) (Tresp 2000a) and its robust variant (rBCM) (Deisenroth and Ng 2015; Liu et al. 2018) explicitly integrate the GP prior to mitigate overconfidence. To address real-time learning problems within distributed frameworks, Lederer et al. (2021) proposed LoG-GP, which incrementally constructs a tree-structured ensemble of GP experts offering a scalable solution for DGPs. However, the reliance

on partitioning along a single dimension limits effectiveness in high-dimensional spaces. The structure of DGPs enables parallel and distributed deployment, with each GP model hosted on a separate computational node or agent. This is widely employed in multi-agent systems, where agents can independently operate local GPs and collaborate through a communication network for cooperative learning (Yang et al. 2021, 2024a,b; Lederer et al. 2023). Nevertheless, these methods overlook online learning requirements, particularly the need to efficiently incorporate new observations and adapt to nonstationary environments (Yuan and Zhu 2024; Dai et al. 2024, 2025). While Yang, Dai, and Hirche (2025) tackle asynchronous communication, their computation-aware random splitting strategy fails to exploit spatial or temporal correlations in the streaming data.

These limitations motivate our streaming kernel-induced progressively generated expert framework for Gaussian processes (SkyGP), which handles non-stationary, streaming data by dynamically allocating GP experts based on kernel similarity and temporal recency. Our main contributions:

- We propose a progressive expert generation strategy that leverages kernel-induced centers to determine whether incoming data should be incorporated into an existing expert model or used to initialize a new one. SkyGP enables dynamic partitioning of streaming data and addresses the limitations of the state-of-the-art (SOTA) approaches.
- We develop a time-aware and configurable expert aggregation framework that incorporates temporal weighting to effectively manage the generated GP experts. The framework adapts to system constraints, such as memory and computational budgets, while ensuring bounded complexity during both training and inference.
- We provide a learning-based policy for dynamical system control tasks with a rigorous theoretical analysis. The main theorem reveals the relationship between model uncertainty and control performance.
- We validate the proposed approach on real-world benchmark datasets and real-time control tasks. Extensive experiments demonstrate that the Sky-GP outperforms SOTA methods in terms of prediction accuracy, computational efficiency, and closed-loop control performance.

2 Problem Statement and Preliminaries

This paper investigates the problem of online function approximation in a streaming data environment for learning and control tasks. The objective is to infer an unknown target function $f(\cdot) : \mathbb{R}^m \to \mathbb{R}$ using a estimated function $\hat{f}(\cdot)$ based on given an sequential stream of input-output data pairs $(\boldsymbol{x}^s, y^s)$ indexed by $s \to \infty$, where $m, s \in \mathbb{N}_{>0}$. Each input $\boldsymbol{x}^s \in \mathbb{R}^m$ is paired with an output $y^s = f(\boldsymbol{x}^s)+\varepsilon \in \mathbb{R}$, where the noise ε follows a normal distribution with zero mean and variance σ_n^2, and $\sigma_n \in \mathbb{R}_{>0}$.

Gaussian Processes represent a Bayesian non-parametric approach commonly employed for function regression and approximation tasks. It assumes that the target function $f(\cdot)$ is sampled from a GP prior, denoted as $f(\cdot) \sim \mathcal{GP}(m(\cdot), \kappa(\cdot,\cdot))$, where $m(\cdot)$ is the mean function (often set to zero when no prior knowledge is assumed) (Rasmussen and Williams 2006) and $\kappa(\cdot,\cdot)$ is the covariance kernel (e.g., the squared exponential kernel). With the streaming observations accumulated up to time step $t_k \in \mathbb{R}_{\geq 0}$ forming the dataset $\mathbb{D}(t_k) = \{(\boldsymbol{x}^s, y^s)\}_{s=1,2,\ldots,N(t_k)}$, where $N(t_k) = |\mathbb{D}(t_k)|$, the posterior mean and variance functions of the GP can be derived as follows

$$\mu(\boldsymbol{x}) = \boldsymbol{\kappa}(\boldsymbol{x}, \boldsymbol{X})\boldsymbol{\alpha}, \quad \sigma^2(\boldsymbol{x}) = \kappa(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{v}^T\boldsymbol{v}, \quad (1)$$

where $\boldsymbol{\kappa}(\boldsymbol{x}, \boldsymbol{X}) = [\kappa(\boldsymbol{x}, \boldsymbol{x}^s)]_{s=1,\ldots,N(t_k)}$ is the covariance vector between the test input $\boldsymbol{x}$ and training inputs $\boldsymbol{X} = [\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{N(t_k)}]$, and $\boldsymbol{\alpha}$, $\boldsymbol{v}$ are computed using the Cholesky decomposition $\boldsymbol{L} = \text{cholesky}(\boldsymbol{K} + \sigma_n^2\boldsymbol{I})$ as

$$\boldsymbol{\alpha} = \boldsymbol{L}^\top \backslash (\boldsymbol{L}^\top \backslash \boldsymbol{y}), \qquad \boldsymbol{v} = \boldsymbol{L} \backslash \boldsymbol{\kappa}(\boldsymbol{x}, \boldsymbol{X}) \quad (2)$$

with the training output $\boldsymbol{y} = [y^1, \ldots, y^{N(t_k)}]^\top$. The kernel matrix $\boldsymbol{K}$ is defined by its elements $K_{ij} = \kappa(\boldsymbol{x}^i, \boldsymbol{x}^j)$ (Rasmussen and Williams 2006).

In online learning scenarios, the predictive mean and variance computations in (1) demonstrate $\mathcal{O}(N(t_k))$ and $\mathcal{O}(N^2(t_k))$ complexities respectively, indicating these operations computationally intractable as data grows without bound. To address these scalability challenges, a widely adopted strategy involves partitioning the data into multiple subsets and training separate GP experts with their means $\mu_i(\cdot)$ and variances $\sigma_i^2(\cdot)$, where $i \in \mathbb{N}$ denotes the index of the GP expert. Various aggregation techniques have been proposed for DGPs, and the structure of such aggregation methods can be formulated in a general framework as

$$\tilde{\mu}(\boldsymbol{x}) = \sum\nolimits_{i\in\mathcal{N}} \omega_i(\boldsymbol{x})\mu_i(\boldsymbol{x}), \quad (3a)$$

$$\tilde{\sigma}^2(\boldsymbol{x}) = \sum\nolimits_{i\in\mathcal{N}} \varpi_i(\boldsymbol{x})\sigma_i^2(\boldsymbol{x}), \quad (3b)$$

where $\mathcal{N} \subset \mathbb{N}$ is the set of all GP experts, the functions $\omega_i(\cdot) : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ and $\varpi_i(\cdot) : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ represent the aggregation weights for the mean and variance, respectively.

3 Generated Experts of Gaussian Processes

In this section, we introduce an efficient online distributed learning framework designed to adaptively process streaming data in real time. At the core of SkyGP is a fully online and adaptive architecture that maintains a bounded, scalable set of GP experts. To enable dynamic expert allocation and effective aggregation, we first present the kernel-based center mechanism in Section 3.1. Building on this, we describe the progressive generation strategy for updating and predicting with the GP experts in Section 3.2, followed by the aggregation strategy in Section 3.3. Finally, we analyze the bounded computational complexity of SkyGP in Section 3.4.

3.1 Kernel-Induced Distance

To enable partitioning and localization for newly generated GP experts, we adopt a center-based representation of the training inputs, where the center of the i-th expert $\mathcal{GP}_i$ is defined as $\boldsymbol{c}_i = \frac{1}{N_i}\sum_{k=1}^{N_i} \boldsymbol{x}_i^k$, following (Nguyen-tuong, Peters, and Seeger 2008). Each expert maintains its own representative center to support fast and scalable online allocation. To facilitate online computation, the center of each

Algorithm 1: Expert Localization with an Adaptive Window

Require: $\boldsymbol{x}^k$, ν_{prev}, $\{\vartheta_i^{k-1}\}_{i\in\mathcal{N}(t_{k-1})}$

Step I: Compute adaptive window size

if $k > 1$ **then**

  Compute distance: $d_{\text{temp}} \leftarrow 1/\kappa(\boldsymbol{x}^{k-1}, \boldsymbol{x}^k)$

  Window size: $W \leftarrow \min(\bar{W},\ \lfloor \exp(d_{\text{temp}}/\varrho) \rfloor)$

else $W \leftarrow 0$ **end if**

Step II: Locate nearest expert

if $W \neq 0$ **then**

  $\mathcal{I} \leftarrow \{\nu_{\text{prev}} - W, \ldots, \nu_{\text{prev}} + W\}$ from $\mathcal{T}(t_{k-1})$

  $\mathcal{I} \leftarrow \{i \in \mathcal{I} \cap \mathcal{N}(t_{k-1}) \mid \vartheta_i^{k-1} > \bar{\vartheta}\}$

else $\mathcal{I} \leftarrow \nu_{\text{prev}}$ **end if**

$\nu_{\text{nr}} \leftarrow \arg\max_{i\in\mathcal{I}} \kappa(\boldsymbol{x}^k, \boldsymbol{c}_i)$; $\nu_{\text{prev}} \leftarrow \nu_{\text{nr}}$

$\mathcal{GP}_{\text{nr}}(t_k) \leftarrow$ Choose GP expert corresponding to ν_{nr}

return $\mathcal{I}$ and ν_{prev}

expert is updated incrementally as new data points are assigned. Specifically, when a new point $\boldsymbol{x}^k$ arriving at time step t_k is allocated to the selected expert, the expert center is updated as

$$\boldsymbol{c}_i^k = (k-1)\boldsymbol{c}_i^{k-1}/k + \boldsymbol{x}^k/k. \tag{4}$$

To evaluate the distance between the expert center $\boldsymbol{c}_i^k$ and the newly arrived input $\boldsymbol{x}^k$ in the feature space induced by the kernel, we define the kernel-based distance as follows

$$d_i^k(\boldsymbol{c}_i^k, \boldsymbol{x}^k) = 1/\kappa(\boldsymbol{c}_i^k, \boldsymbol{x}^k). \tag{5}$$

3.2 Progressive Expert Generation Strategy

Expert Localization and Generation In conventional online GP approximation methods, new data points are typically assigned to the first available expert that has not yet reached its capacity (Nguyen-tuong, Peters, and Seeger 2008; Lederer et al. 2021; Yang, Dai, and Hirche 2025). Once the expert becomes full, it is either no longer updated or directly split into sub-experts, often without explicitly accounting for the underlying distributional properties and space features of the data. In contrast, our SkyGP framework dynamically allocates each incoming data point to the most appropriate expert based on kernel proximity and time-aware factor. Specifically, from the prediction phase, we have already calculated the kernel distances between the new arriving point and the experts within an adaptive window $W \in \mathbb{I}_{[1,\bar{W}]}$ and found the nearest experts with in the expert set $\mathcal{N}(t_k)$, where $\mathcal{N}(t_0) = 1$ (see Algorithm 1). After locating the previous index ν_{prev} of the nearest expert denoted by ν_{nr}, we are able to process the streaming data based on the index list $\mathcal{I}$ filtered by the time-aware factor $\vartheta \in (0, 1]$ initialized by 1, which encodes the usage history of the experts. Moreover, we define the upper bound of ϑ as $\bar{\vartheta}$ to filter out the outdated experts. If the expert has not reached its capacity, the data point is added to it (see Figure 1). Otherwise, the two SkyGP variants handle this situation in two different ways outlined in Algorithm 2.

In particular, a data replacement strategy is employed in SkyGP-Dense. When the new data pair $(\boldsymbol{x}^k, y^k)$ is confirmed to add to $\mathbb{D}_{\text{nr}}(t_k)$, and the data point within $\mathcal{GP}_{\text{nr}}$ that

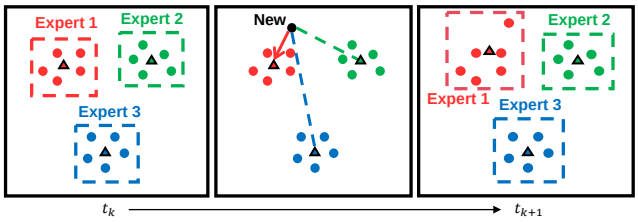


Figure 1: Data allocation process

is furthest from its center is cast off and moved to a separate dropped dataset $\mathbb{D}_{\text{nr}}^{\text{off}}(t_k)$, whose center ${}^{\text{off}}\boldsymbol{c}_{\text{nr}}^k$ is then updated in the same method in (4). Since such a replacement requires recomputing the expert's kernel matrix and Cholesky factorization, we introduce an event-triggered mechanism to limit updates only to critical cases. Let $\boldsymbol{c}_{\text{nr}}$ and ${}^{\text{off}}\boldsymbol{c}_a$ be the center and the dropped center of expert $\mathcal{GP}_{\text{nr}}$ with $N_{\text{nr}}(t_k)$ data points $\{(\boldsymbol{x}_{\text{nr}}^s, y_{\text{nr}}^s\}_{s=1}^{N_{\text{nr}}(t_k)}$. The event-triggered data forwarding strategy is designed as

$$\mathbb{D}_{\text{nr}}(t_k) = \begin{cases} \mathbb{D}_{\text{nr}}(t_{k-1}) \cup (\boldsymbol{x}^k, y^k) \setminus (\boldsymbol{x}^{k_{\text{off}}}, y^{k_{\text{off}}}), & \text{if } \Delta < 0 \\ \mathbb{D}_{\text{nr}}(t_{k-1}), & \text{otherwise} \end{cases}, \tag{6}$$

where $\Delta \in \mathbb{R}$ is defined as $\Delta = \max_{s=1,\cdots,N_{\text{nr}}(t_k)} \Delta(s)$ and

$$\Delta(s) = \kappa(\boldsymbol{x}^s, \boldsymbol{c}_{\text{nr}}) - \kappa(\boldsymbol{x}^s, {}^{\text{off}}\boldsymbol{c}_{\text{nr}}) - \kappa(\boldsymbol{x}^k, \boldsymbol{c}_{\text{nr}}) + \kappa(\boldsymbol{x}^k, {}^{\text{off}}\boldsymbol{c}_{\text{nr}}).$$

This kernel-based selection strategy encourages experts to retain points that align with the current local distribution while avoiding repeated inclusion of previously discarded regions. Although computationally expensive, this step is infrequent and remains tractable under the predefined data threshold $\bar{N}$. If no expert accepts $\boldsymbol{x}^k$, a new expert will be created and the center will be initialized with $\boldsymbol{x}^k$. In SkyGP-Fast, no replacement is performed. This design ensures all update GP expert is a rank-one update to the Cholesky factor, which incurs a computational complexity of $\mathcal{O}(N_{\text{nr}}^2)$. SkyGP-Fast bypasses recomputations by appending new experts instead of replacing data, favoring low-latency updates and scalability. In contrast, SkyGP-Dense limits memory usage by reusing expert slots, making it more suitable for resource-constrained settings. Together, these two modes offer a flexible trade-off between computational efficiency and memory control across diverse online learning scenarios.

Dynamic Expert List Creation The expert search is implemented over a center-indexed list that maintains a globally ordered list of expert centers based on their insertion positions. The index of this list is defined as $\nu_i \in \mathbb{N}$ with corresponding $\mathcal{GP}_i$ expert. The newly instantiated expert with center $\boldsymbol{x}^k$ will be inserted adjacent to the nearest existing expert identified during search. To update the table when new GP expert generated, we compare the kernel-based distance between $\boldsymbol{x}^k$ and the left/right neighbors of the selected experts with their corresponding center $\boldsymbol{c}_{\text{nr}}^-$ and $\boldsymbol{c}_{\text{nr}}^+$:

$$d_{\text{left}} = 1/\kappa(\boldsymbol{x}^k, \boldsymbol{c}_{\text{nr}}^-), \qquad d_{\text{right}} = 1/\kappa(\boldsymbol{x}^k - \boldsymbol{c}_{\text{nr}}^+) \tag{7}$$

Algorithm 2: Expert Update Strategy

Require: $\boldsymbol{x}^k$, ν_{prev}, $\mathcal{I}$, $\{\vartheta_i^{k-1}\}_{i\in\mathcal{N}(t_{k-1})}$

Step I: Aggregated Prediction

1: $\mathcal{I}_{\text{temp}} \leftarrow \{\nu \in \mathcal{I} \mid \vartheta_\nu^{k-1} > \bar{\vartheta}\}$
2: $\mathcal{I}_{\text{agg}} \leftarrow$ indices of the largest $\bar{\mathcal{N}}$ values in $\mathcal{I}_{\text{temp}}$
3: $\tilde{\mu}(\boldsymbol{x}^k), \tilde{\sigma}(\boldsymbol{x}^k) \leftarrow$ (3) with MOE/POE/BCM (9) to (11)

Step II: GP Expert Model Update

4: $\{\vartheta_i^k\}_{i\in\mathcal{N}} \leftarrow \{\vartheta_i^{k-1}\}_{i\in\mathcal{N}}$
5: **for all** $\nu \in \mathcal{I}_{\text{agg}}$ **do**
6: $\quad\vartheta_\nu^k \leftarrow 1, \mathbb{D}_{\text{temp}} \leftarrow \{\boldsymbol{x}\in\mathbb{D}_\nu(t_{k-1}) | \kappa(\boldsymbol{x},\boldsymbol{c}_\nu) > \kappa(\boldsymbol{x}^k,\boldsymbol{c}_\nu)\}$
7: $\quad$**if** $|\mathbb{D}_\nu(t_{k-1})| < \bar{N}$ **then**
8: $\quad\quad\mathbb{D}_\nu(t_k) \leftarrow \mathbb{D}_\nu(t_{k-1}) \cup (\boldsymbol{x}^k, y^k)$
9: $\quad\quad$Update $\mathcal{GP}_\nu(t_k) \leftarrow$ Cholesky rank-one update
10: $\quad\quad$**return** $\{\vartheta_i^k\}_{i\in\mathcal{N}(t_k)}$
11: $\quad$**else if** `SkyGP-Dense` and $\mathbb{D}_{\text{temp}} = \emptyset$ **then**
12: $\quad\quad$Update $\mathbb{D}_\nu(t_k) \leftarrow$ (6), $\mathcal{GP}_\nu(t_k) \leftarrow$ (2)
13: $\quad\quad$Update ${}^{\text{off}}\boldsymbol{c}_\nu \leftarrow$ (4) with dropped data
14: $\quad\quad$**return** $\{\vartheta_i^k\}_{i\in\mathcal{N}(t_k)}$
15: $\quad$**end if**
16: **end for**
17: Create $\mathcal{GP}_{|\mathcal{N}(t_{k-1})|+1}$ with dataset $\mathbb{D}(t_k) = (x_t, y_t)$
18: Update expert list $\mathcal{T}(t_k) \leftarrow \mathcal{T}(t_{k-1})$ and (8)
19: Update $\mathcal{N}(t_k) \leftarrow \{\mathcal{N}(t_{k-1}), |\mathcal{N}(t_{k-1})| + 1\}$

Step III: Update Time-Aware Factor

20: **for** $i = \mathcal{N}\backslash\mathcal{I}_{\text{agg}}$ **do** $\vartheta_i^k \leftarrow \rho\vartheta_i^{k-1}$ **end for**
21: **return** $\{\vartheta_i^k\}_{i\in\mathcal{N}(t_k)}$

According to the previous ν_{nr} in Algorithm 1, the new index of $\mathcal{GP}_{new}$ is

$$\nu_{\text{new}} = \begin{cases} \min(\nu_{\text{nr}}\,|\mathcal{N}|), & \text{if } d_{\text{right}} < d_{\text{left}} \\ \max(\nu_{\text{nr}} - 1,\ 0) & \text{otherwise} \end{cases}. \tag{8}$$

3.3 Aggregated Prediction

Leveraging DGP framework in (3), our proposed SkyGP can be seamlessly integrated with existing aggregation strategies to enhance predictive performance and uncertainty estimation across decentralized data streams. These weights are typically chosen to satisfy certain properties, such as non-negativity and normalization, ensuring that the aggregated predictions are coherent and interpretable. For example, the aggregated weight functions of MoE are

$$\omega_i(\boldsymbol{x}) = w_i, \tag{9a}$$

$$\varpi_i(\boldsymbol{x}) = w_i\big(\sigma_i^2(\boldsymbol{x}) + \mu_i^2(\boldsymbol{x})\big) - \tilde{\mu}^2(\boldsymbol{x})/w_i, \tag{9b}$$

satisfying $\sum_{i\in\mathcal{I}_{\text{agg}}} w_i = 1$, which is also utilized in the mixture of explicitly localized experts. Considering the posterior variance in GP experts, the PoE family methods employ the following aggregation structure

$$\omega_i(\boldsymbol{x}) = w_i\sigma_i^2(\boldsymbol{x})/\varpi_i(\boldsymbol{x}), \tag{10a}$$

$$\varpi_i(\boldsymbol{x}) = 1/\sum\nolimits_{s\in\mathcal{I}_{\text{agg}}} w_s\sigma_s^{-2}(\boldsymbol{x}). \tag{10b}$$

Furthermore, these aggregation schemes can be extended to incorporate the prior variance of the unknown function, denoted as σ_*, to further refine the combined predictive uncertainty in the BCM family approaches. In this context, the weight function for the mean remains consistent with that of the PoE in (10a), while the corresponding weight function for the aggregated posterior variance is given by

$$\varpi_i(\boldsymbol{x}) = 1/\Big(\sum\nolimits_{s\in\mathcal{I}_{\text{agg}}} w_j\sigma_j^{-2}(\boldsymbol{x}) + \big(1 - \sum\nolimits_{j\in\mathcal{I}_{\text{agg}}} w_j\big)\sigma_*^{-2}\Big). \tag{11}$$

This extension ensures that the aggregated predictions remain well-calibrated, especially in distributed or federated learning scenarios where each expert may have access to different subsets of the data.

According to Algorithm 1, given a new input $\boldsymbol{x}^k$, SkyGP performs prediction by first identifying the definable $\bar{\mathcal{N}}$ number of experts within the search window. Based on the returned aggregation expert list $\mathcal{I}_{\text{agg}}$, each expert predicts the posterior mean $\mu_i(\boldsymbol{x}^k)$ and variance $\sigma_i^2(\boldsymbol{x}^k)$. These predictions then are fused using a principled weighting strategy, e.g., MoE, PoE or BCM, yielding the final prediction in (3).

3.4 Bounded Complexity Analysis

In this section, we formalize the computational efficiency of the proposed SkyGP framework, which is designed to maintain bounded per-step complexity during both model update and prediction. The primary sources of computational cost include nearest expert localization, aggregated prediction, and expert model update. Each component is explicitly designed to ensure low complexity and scalability in streaming scenarios, while maintaining high prediction accuracy.

Model Update For expert localization, as described in Algorithm 1, the computational cost from evaluating kernel similarities between the current input and the expert centers to obtaining index set $\mathcal{I}$ (line 1-8) is $\mathcal{O}(1)$. The subsequent selection of the nearest expert from the candidate set $\mathcal{I}$ in line 9, even when incorporating the time-aware factor, requires only a linear search over $\mathcal{I}$ with the same order of complexity, i.e., $\mathcal{O}(W)$. Consequently, the overall computational complexity of Algorithm 1 is $\mathcal{O}(W)$.

The overall computational cost of the Algorithm 2 is dominated by three components. First, the selection of the aggregated expert set in line 2 requires filtering the candidate index set $\mathcal{I}$ and selecting the top $\bar{\mathcal{N}}$ elements, which can be performed in $\mathcal{O}(|\mathcal{I}|\log\bar{\mathcal{N}})$ time. Second, the update phase iterates over the $\bar{\mathcal{N}}$ aggregated experts. For each, the construction of the temporary data subset incurs $\mathcal{O}(|\mathbb{D}_\nu(t_{k-1})|)$ cost, which is bounded by $\mathcal{O}(\bar{N})$ considering $|\mathbb{D}_\nu(t_k)| \le \bar{N}$. Moreover, when a GP model is updated via a rank-one Cholesky modification for SkyGP-Fast, an additional $\mathcal{O}(\bar{N}^2)$ cost per expert is incurred. On the other hand, SkyGP-Dense requires full Cholesky factor recomputation when performing replacement operations, incurring a higher cost of $\mathcal{O}(\bar{N}^3)$. Third, the time-aware weighting update over the remaining experts requires $\mathcal{O}(\bar{\mathcal{N}})$ operations. Hence, the total worst-case complexity can be expressed as $\mathcal{O}(W\log\bar{\mathcal{N}} + \bar{\mathcal{N}}\bar{N}^2)$ for SkyGP-Fast, and similarly $\mathcal{O}(W\log\bar{\mathcal{N}} + \bar{\mathcal{N}}\bar{N}^3)$ for SkyGP-Dense.

Prediction Aggregated mean and variance in SkyGP are computed using the bounded top $\bar{\mathcal{N}}$ nearest experts. As each local GP performs inference with $\mathcal{O}(N^2(t_k))$ complexity, the overall prediction complexity per step is $\mathcal{O}(\bar{\mathcal{N}}N^2(t_k))$.

Remark 1. Other auxiliary operations in SkyGP, such as list insertion, temporal decay updates, incur at most linear complexity with respect to the number of queried experts $\bar{\mathcal{N}}$, i.e., $\mathcal{O}(\bar{\mathcal{N}})$. As these operations are not computationally dominant, we omit their detailed analysis for clarity.

4 Safe Learning-based Control Policy

In this section, we present a learning-based control policy that integrates the proposed SkyGP framework into nonlinear systems with unknown dynamics. The key idea is to leverage real-time GP predictions to estimate model uncertainties and design a feedback controller that ensures safe and stable trajectory tracking. Following the introduction of the prediction error bound of SkyGP in Section 4.1, we analyze the control performance for general dynamical systems and design a tracking control policy specifically for Euler–Lagrange (EL) systems in Section 4.2.

4.1 Learning Performance of SkyGP

We consider a function f belonging to a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_\kappa$ associated with a positive definite kernel $\kappa(\cdot,\cdot)$, such that $\|f\|_\kappa \leq \Gamma$. Let $\mu_i(\boldsymbol{x})$ and $\sigma_i(\boldsymbol{x})$ denote the posterior mean and standard deviation from the i-th GP expert trained on dataset $\mathbb{D}_i$ for $i = 1, \cdots, \mathcal{N}$. Then the following probabilistic prediction error bound holds.

Lemma 1. *Consider the regression task in a compact domain $\mathbb{X}$, and suppose the kernel function $\kappa(\cdot,\cdot)$ is Lipschitz with Lipschitz constant $L_\kappa \in \mathbb{R}_{0,+}$. Choose $\delta \in (0, 1/\bar{\mathcal{N}})$ and $\tau \in \mathbb{R}_+$, then the prediction error satisfies*

$$|f(x) - \tilde{\mu}(\boldsymbol{x} \mid \mathbb{D})| \leq \beta\sigma(\boldsymbol{x}) + \gamma(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \mathbb{X} \tag{12}$$

with a probability of at least $1 - \bar{\mathcal{N}}\delta$, where $\gamma(\boldsymbol{x}) = \sum_{i\in\mathcal{N}} \omega_i(\boldsymbol{x})\gamma_i$, $\sigma(\boldsymbol{x}) = \sum_{i\in\mathcal{N}} \omega_i(\boldsymbol{x})\sigma_i(\boldsymbol{x})$, and

$$\beta = 2\Big(2\sum\nolimits_{j=1}^{n} \log\Big\lceil \frac{\sqrt{n}}{2\tau}(\bar{x}_j - \underline{x}_j)\Big\rceil - 2\log\frac{\delta}{n}\Big)^{1/2}, \tag{13}$$

$$\gamma_i = \big(\sqrt{\beta_\delta}L_{\sigma,i} + \Gamma\sqrt{2L_\kappa} + L_{\mu,i}\big)\tau, \tag{14}$$

with $\bar{x}_j = \max_{x\in\mathbb{X}} x_j$, $\underline{x}_j = \min_{x\in\mathbb{X}} x_j$ and x_j as the j-th dimension of x. The positive constants $L_{\mu,i} \in \mathbb{R}_{0,+}$ and $L_{\sigma,i} \in \mathbb{R}_{0,+}$ are the Lipschitz constant for posterior mean $\mu_i(\cdot)$ and variance $\sigma_i(\cdot)$, respectively for $i \in \mathcal{N}$.

Lemma 1 shows the probabilistic theoretical error bound of the aggregated prediction $\tilde{\mu}(\cdot)$, which also holds for the proposed Sky-GP[1]. Note that the boundness of the prediction error does not rely on the choice of aggregating strategy reflected by the weighting function $\omega_i(\cdot)$. For notational simplicity, denote the prediction error bound as $\eta(\boldsymbol{x}) = \beta\sigma(\boldsymbol{x}) + \gamma(\boldsymbol{x})$, which is used in the following control analysis.

Remark 2. Compared with the other probabilistic (Srinivas et al. 2012; Scharnhorst et al. 2022) or deterministic bounds (Maddalena, Scharnhorst, and Jones 2021; Hashimoto et al. 2022), the error bound chosen in this paper follows (Lederer, Umlauft, and Hirche 2019), which results in an amplitude constant β of posterior variance $\sigma(\cdot)$. Moreover, it is shown that there always exists a sufficiently small grid factor $\tau \in \mathbb{R}_+$ such that $\gamma(\boldsymbol{x}) \ll \beta\sigma(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathbb{X}$, i.e., the posterior variance dominates the error bound $\eta(\boldsymbol{x})$. Therefore, the prediction error bound could also be written as $\eta(\boldsymbol{x}) \leq 2\beta\sigma(\boldsymbol{x})$ (Lederer et al. 2024).

[1]The proofs of all lemmas and theorems are provided in Appendix A.

4.2 Safe Critical Control Application

A General Control System The objective is to design a GP-based control law that drives an unknown dynamical system to exhibit a desired behavior. Specifically, we consider a class of control applications focused on output stabilization, where the goal is to ensure that the system output converges to the equilibrium point asymptotically. The continuous-time nonlinear system is described as follows

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}), \qquad \boldsymbol{z} = \boldsymbol{h}(t, \boldsymbol{x}) \tag{15}$$

where $\boldsymbol{x} \in \mathbb{X} \subset \mathbb{R}^m$, $\boldsymbol{u} \in \mathbb{U} \subset \mathbb{R}^{m_u}$ and $\boldsymbol{z} \in \mathbb{R}^{m_z}$ denote the system state, control input and system output, respectively. The transition function $\boldsymbol{f}(\cdot,\cdot) = [f_1(\cdot,\cdot), \cdots, f_m(\cdot,\cdot)]^\top : \mathbb{X} \times \mathbb{U} \to \mathbb{R}^m$ is unknown, where each scalar $f_j(\cdot,\cdot)$ belongs to a RKHS $\mathcal{H}_{\kappa,j}$ corresponding to a kernel function $\kappa_j(\cdot,\cdot) : \mathbb{X}_e \times \mathbb{X}_e \to \mathbb{R}_{0,+}$ with $\mathbb{X}_e = \mathbb{X} \times \mathbb{U} \subset \mathbb{R}^{m+m_u}$ for all $j = 1, \cdots, m$. The measurement function $\boldsymbol{h}(\cdot,\cdot) : \mathbb{R}_{\geq 0} \times \mathbb{X} \to \mathbb{R}^{m_z}$ is known. To achieve system output stabilization, i.e., $\lim_{t\to\infty} \boldsymbol{z}(t) = \boldsymbol{0}_{m_z\times 1}$, a control policy $\boldsymbol{\pi}(\cdot,\cdot,\cdot) : \mathbb{R}_{\geq 0} \times \mathbb{X} \times \mathcal{H}_\kappa \to \mathbb{R}^{m_u}$ is designed satisfying the following assumption.

Assumption 1. Suppose there exists a differentiable function $V(\cdot,\cdot) : \mathbb{R}_{\geq 0} \times \mathbb{R}^{m_y} \to \mathbb{R}_{\geq 0}$ satisfying

$$\underline{\alpha}(\|\boldsymbol{z}\|) \leq V(t, \boldsymbol{z}) \leq \bar{\alpha}(\|\boldsymbol{z}\|) \tag{16}$$

with class-$\mathcal{K}$ functions $\underline{\alpha}(\cdot), \bar{\alpha}(\cdot) : \mathbb{R}_{\geq 0} \to \mathbb{R}_{0,+}$. The control law $\boldsymbol{\pi}(\cdot,\cdot,\cdot)$ is designed such that

$$\begin{aligned} &\nabla_{\boldsymbol{x}} V(t,\boldsymbol{x})(\boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{\pi}(t,\boldsymbol{x},\boldsymbol{\mu})) + \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\pi}(t,\boldsymbol{x},\boldsymbol{\mu}))) \qquad (17)\\ &+ \nabla_t V(t,\boldsymbol{x}) \leq -\alpha(V(t, \boldsymbol{h}(t,\boldsymbol{x}))) - \|\nabla_{\boldsymbol{x}} V(t,\boldsymbol{x})\|^2/(4\varepsilon), \end{aligned}$$

with a positive constant $\varepsilon \in \mathbb{R}_{>0}$, where the function $\alpha(\cdot) : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ belongs to class-$\mathcal{K}$ and

$$\begin{aligned} \nabla_t V(t,\boldsymbol{x}) &= \frac{\partial V(t, \boldsymbol{h}(t,\boldsymbol{x}))}{\partial t} + \frac{\partial V(t, \boldsymbol{h}(t,\boldsymbol{x}))}{\partial \boldsymbol{h}(t,\boldsymbol{x})}\frac{\partial \boldsymbol{h}(t,\boldsymbol{x})}{\partial t},\\ \nabla_{\boldsymbol{x}} V(t,\boldsymbol{x}) &= \frac{\partial V(t, \boldsymbol{h}(t,\boldsymbol{x}))}{\partial \boldsymbol{h}(t,\boldsymbol{x})}\frac{\partial \boldsymbol{h}(t,\boldsymbol{x})}{\partial \boldsymbol{x}}, \end{aligned} \tag{18}$$

for any $\boldsymbol{\mu}(\cdot) = [\mu_1(\cdot), \cdots, \mu_m(\cdot)]^T$ satisfying $\mu_i(\cdot) \in \mathcal{H}_{\kappa,j}$ for all $j = 1, \cdots, m$.

This assumption indicates the asymptotic output stability of the equivalent system $\dot{\boldsymbol{x}} = \boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u})$ using the control law $\boldsymbol{\pi}(t, \boldsymbol{x}, \boldsymbol{\mu})$. Compared to the condition in conventional asymptotic stability (Khalil 2002), the additional term $\|\nabla_{\boldsymbol{x}} V(t,\boldsymbol{x})\|^2/(4\varepsilon)$ introduced in (17) is used following input-to-state control Lyapunov functions to partially compensate for the potential effect from unknown disturbances. Later, an example is provided to demonstrate how to design a model-based controller that satisfies this assumption. Notably, the true dynamical system is unknown $\boldsymbol{f}(\cdot)$ instead of $\boldsymbol{\mu}(\cdot)$, whose performance is shown as follows.

Theorem 1. *If there exists a control law $\boldsymbol{\pi}(\cdot,\cdot,\cdot)$ satisfying Assumption 1. Choose $\delta \in (0, 1/m)$, then the output $\boldsymbol{z}(t)$ of the true system* (15) *is ultimately bounded by*

$$\lim_{t\to\infty} \|\boldsymbol{z}(t)\| \leq \underline{\alpha}^{-1}(\alpha^{-1}(\varepsilon\bar{\eta}^2)) \tag{19}$$

with a probability of at least $1 - m\delta$. The positive constant $\bar{\eta} = \max_{t\in\mathbb{R}_{0,+},\boldsymbol{x}\in\mathbb{X}} \|\boldsymbol{\eta}(\boldsymbol{\xi}(t,\boldsymbol{x},\boldsymbol{\mu}))\|$ denotes an upper bound on the model error between $\boldsymbol{f}(\cdot)$ and its GP approximation $\boldsymbol{\mu}$, where $\boldsymbol{\xi}(t,\boldsymbol{x},\boldsymbol{\mu}) = [\boldsymbol{x}^T, \boldsymbol{\pi}^T(t,\boldsymbol{x},\boldsymbol{\mu})]^T$ and $\boldsymbol{\eta}(\cdot) = [\eta_1(\cdot),\cdots,\eta_{d_x}(\cdot)]^T$ and $\eta_i(\cdot)$ obtained using prediction error bound.

This theorem guarantees that the system output ultimately remains within a bounded neighborhood of the desired equilibrium, despite the presence of model uncertainties. To instantiate this theorem and derive a practically applicable controller, we next consider a representative Euler–Lagrange (EL) system and demonstrate the design and performance of a corresponding learning-based policy.

Euler-Lagrange System Control Consider an EL system

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \boldsymbol{u} + \boldsymbol{d}(\boldsymbol{q},\dot{\boldsymbol{q}}), \tag{20}$$

where $\boldsymbol{q}\in\mathbb{Q}\subset\mathbb{R}^{m_q}$ and $\boldsymbol{u}\in\mathbb{U}\subset\mathbb{R}^{m_u}$ denote the generalized coordinate and control input with $m_q = m/2$. Define $\boldsymbol{x} = [\boldsymbol{q}^\top, \dot{\boldsymbol{q}}^\top]^\top \in \mathbb{X}$, then the dynamics of (20) is rewritten as

$$\dot{\boldsymbol{x}} = \underbrace{\begin{bmatrix}\dot{\boldsymbol{q}}\\ \boldsymbol{M}^{-1}(\boldsymbol{q})(\boldsymbol{u} - \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{g}(\boldsymbol{x}))\end{bmatrix}}_{\boldsymbol{g}(\boldsymbol{x},\boldsymbol{u})} + \underbrace{\begin{bmatrix}\boldsymbol{0}_{m_q\times 1}\\ \boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{d}(\boldsymbol{x})\end{bmatrix}}_{\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})}$$

The control task is to track a pre-defined trajectory $\boldsymbol{x}_d(\cdot) = [\boldsymbol{q}_d^\top(\cdot), \dot{\boldsymbol{q}}_d^\top(\cdot)]^\top : \mathbb{R}_{\geq 0} \to \mathbb{X}$, such that the output is defined as tracking error as $\boldsymbol{z}(t) = \boldsymbol{x}(t) - \boldsymbol{x}_d(t)$. To achieve this control objective, a learning-based control law inspired by computed torque control is proposed as

$$\begin{aligned}\boldsymbol{\pi}(t,\boldsymbol{x},\boldsymbol{\mu}) &= \boldsymbol{C}(\boldsymbol{x})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{x}) - \boldsymbol{\mu}(\boldsymbol{x}) + \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}_d(t) \\ &+\boldsymbol{M}(\boldsymbol{q})(\boldsymbol{K}_p(\boldsymbol{q}-\boldsymbol{q}_d(t))+\boldsymbol{K}_d(\dot{\boldsymbol{q}}-\dot{\boldsymbol{q}}_d(t)))-\boldsymbol{B}^T(\boldsymbol{x})\boldsymbol{P}\boldsymbol{z}/(2\varepsilon),\end{aligned} \tag{21}$$

where $\boldsymbol{B} = [\boldsymbol{0}_{d_q\times d_q}, \boldsymbol{M}^{-\top}(\boldsymbol{q})]^\top$. The control gains $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are chosen such that $\boldsymbol{A} = \begin{bmatrix}\boldsymbol{0}_{m_q\times m_q} & \boldsymbol{I}_{m_q}\\ \boldsymbol{K}_p & \boldsymbol{K}_d\end{bmatrix}$ is Hurwitz. The matrix $\boldsymbol{P}$ is the solution to the Lyapunov equation associated with the quadratic Lyapunov function $V(t,\boldsymbol{z}) = \boldsymbol{z}^\top\boldsymbol{P}\boldsymbol{z}$, where $\boldsymbol{P}\succ 0$ satisfies $\boldsymbol{A}^\top\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} = -\boldsymbol{Q}$ with $\boldsymbol{Q}\succ 0$ being a positive definite matrix. The existence and uniqueness of the solution $\boldsymbol{P}$ is guaranteed due to Hurwitz $\boldsymbol{A}$ (Khalil 2002). Then, the control performance in terms of tracking error bound is shown as follows.

Theorem 2. *Consider the EL system* (20) *driven by the control law* (21) *using the proposed Sky-GP satisfying all assumptions in Lemma 1. Choose $\delta \in (0, 1/m_q) \subset \mathbb{R}$, then the tracking error $\boldsymbol{z}$ is ultimately bounded by*

$$\lim_{t\to\infty} \|\boldsymbol{z}(t)\| \leq \varepsilon\bar{\lambda}(\boldsymbol{P})(\underline{\lambda}(\boldsymbol{Q})\underline{\lambda}(\boldsymbol{P}))^{-1}\bar{\eta}^2 \tag{22}$$

with a probability of at least $1 - m_q\delta$.

Theorem 2 shows the tracking error bound obtained by using the controller (21) with the proposed Sky-GP, which is relevant to the worst-case learning performance $\bar{\eta}$ and the desired convergence strength reflected by $\boldsymbol{Q}$. Specifically, smaller prediction error $\bar{\eta}$ and larger eigenvalues of $\boldsymbol{Q}$ induce smaller tracking error. Note that increasing the control gains with larger eigenvalues of $\boldsymbol{Q}$ also leads to high sensitivity of the measurement noise, which may deteriorate the control performance. Therefore, an accurate prediction for a smaller $\bar{\eta}$ is essential to achieve high tracking precision.

5 Experimental Evaluation

We conduct comprehensive experiments to evaluate the proposed SkyGP framework from both regression and control perspectives. In Section 5.1, we assess the regression performance on multiple real-world benchmark datasets. Then, a closed-loop control performance comparison is evaluated against SOTA approaches on a nonlinear dynamical system in Section 5.2. Finally, we present an ablation study analyzing different algorithmic design choices[2].

5.1 Regression Performance Evaluation

We evaluate the proposed SkyGP framework on four real-world benchmark datasets. The SARCOS dataset contains 44,484 samples with 21 input features. The PUMA-DYN32NM (PUMA) dataset consists of 7,168 samples with 32 input features. The KIN40K dataset includes 10,000 samples with 8-dimensional inputs. The ELECTRIC dataset records household electricity consumption, comprising over 2 million samples with 11-dimensional inputs. For simplicity, we use the first 20,000 samples of ELECTRIC dataset. To assess the performance of SkyGP, we compare our two variants against several strong baselines: LoG-GP (Lederer et al. 2021) and Local GPs (Nguyen-tuong, Peters, and Seeger 2008) with each expert holding at most 50 data points, SSGP with 50 inducing points (Bui, Nguyen, and Turner 2017), and ISSGP with 200 random features (Gijsberts and Metta 2013). All methods use an ARD kernel with hyperparameters pre-trained with the first 1000 samples in each dataset. Each model is evaluated in a sequential setting, where predictions and updates are performed on-the-fly as each new data point arrives. We evaluated the average prediction and update times, as well as the standardized mean squared error (SMSE) and the mean standardized log loss (MSLL) in a sequential interpretation (Lederer et al. 2021). The maximal number of experts $\bar{\mathcal{N}} \in \{1, 2, 4\}$ are used in all benchmark. Each expert with SkyGP holds at most $\bar{N} = 50$ data points and the maximal search window size $\bar{W} = 40$ and uses the rBCM aggregation method. Set the time-aware decay weighting factor $\rho = 0.995$ and $\bar{\vartheta} = 10^{-3}$. SkyGP-Fast with a maximum of one expert is denoted by SkyGP-F-1, and SkyGP-Dense with one expert by SkyGP-D-1. Other variants follow the same naming convention.

Table 1 shows the average SMSE and MSLL across three benchmark datasets, demonstrating that the proposed SkyGP variants consistently outperform baseline methods such as

[2]For additional results and ablation study analysis, refer to Appendix B.

Table 1: Average SMSE and MSLL on 3 datasets.

Model	SARCOS		PUMA		ELECTRIC	
	ϵ_{smse}	ϵ_{msll}	ϵ_{smse}	ϵ_{msll}	ϵ_{smse}	ϵ_{msll}
LoG-GP	0.044	-1.83	0.20	-1.00	0.11	-3.08
SkyGP-F-1	0.037	-1.83	0.26	-0.92	0.08	-3.29
SkyGP-F-4	0.024	-1.91	0.09	-1.39	0.07	-3.36
SkyGP-D-1	0.031	-1.89	0.23	-1.10	0.08	-3.31
SkyGP-D-4	**0.017**	**-2.03**	**0.08**	-1.40	0.07	**-3.38**
LocalGPs-1	0.031	-1.90	0.18	-1.11	0.08	-3.26
LocalGPs-4	0.071	-1.38	0.18	-0.79	0.14	-2.20
ISSGP	0.023	-1.91	**0.08**	**-1.46**	**0.06**	-2.19
SSGP	0.068	-1.11	0.09	-1.05	–	–

Table 2: Average prediction and update time on 3 dadtasets.

Model	SARCOS		PUMA		ELECTRIC	
	t_{pred}	t_{up}	t_{pred}	t_{up}	t_{pred}	t_{up}
LoG-GP	0.30	0.22	**0.22**	0.20	0.18	0.24
SkyGP-F-1	**0.16**	**0.04**	0.26	**0.04**	0.15	**0.04**
SkyGP-F-4	0.23	**0.04**	0.35	**0.04**	0.21	**0.04**
SkyGP-D-1	0.17	0.09	0.25	0.08	**0.14**	0.06
SkyGP-D-4	0.24	0.16	0.28	0.16	0.22	0.08
LocalGPs-1	1.17	0.15	1.07	0.08	2.78	0.06
LocalGPs-4	1.25	0.23	1.14	0.08	2.88	0.06
ISSGP	18	7	3	6	4	8
SSGP	5	4	2	6	–	–

LoG-GP and LocalGPs in both predictive accuracy and uncertainty calibration. While ISSGP achieves the best SMSE and MSLL on the PUMA dataset, its prediction and update time is over 30 times slower, as detailed in Table 2, making it less suitable for real-time applications. Notably, SkyGP-D-4 achieves the best overall performance, with the lowest SMSE (0.017) and MSLL (-2.03) on the SARCOS dataset, and competitive results on PUMA and ELECTRIC. Table 2 summarizes the average prediction and update times across three datasets. SkyGP-F-1 achieves the lowest overall latency, with update times consistently at 0.04s and the fastest prediction on SARCOS (0.16s), making it well-suited for real-time applications. SkyGP-D-1 has the best prediction time on ELECTRIC (0.14s) while maintaining competitive update performance. In contrast, traditional baselines such as ISSGP and SSGP exhibit significantly higher computational costs, with ISSGP requiring up to 18s for prediction and 7s for update on SARCOS, rendering them impractical for online deployment. Notably, SSGP fails to produce results on the ELECTRIC dataset within 20s, so we omit it.

5.2 Control Performance Evaluation

We evaluate the control performance of the system described as $m\ddot{q} + 9.8 = u + f(\boldsymbol{x})$ with $m = 1$, where

$$f(\boldsymbol{x}) = 1 + x_1 x_2/10 + \cos(x_2)/2 - 10\sin(5x_1) + (1 + \exp(-x_2/10))^{-1}/2. \quad (23)$$

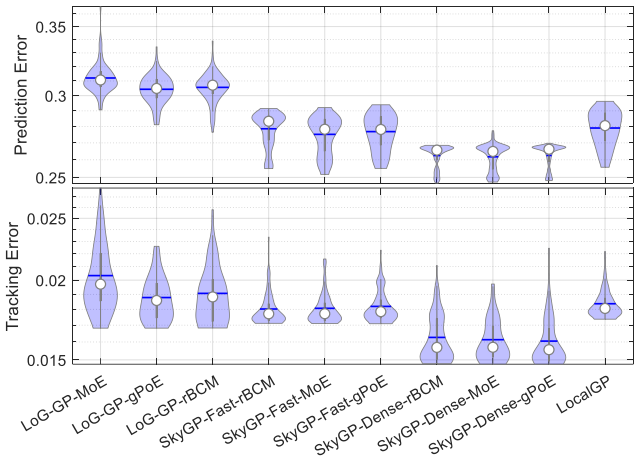


Figure 2: Learning and tracking performance comparison of the control task from the 100 times Monte Carlo tests.

Choose $x_1 = q, x_2 = \dot{q}$, then the dynamics is reformulated as $\dot{x}_1 = x_2,\ \dot{x}_2 = g(u) + f(\boldsymbol{x})$ with $g(u) = u - 9.8 = (u - c\dot{q} - g)/m$ similar as (21) with $c = 0$ and $g = 9.8$.

$$\pi(t, \boldsymbol{x}, \tilde{\mu}) = 9.8 - \tilde{\mu}(\boldsymbol{x}) - a_r w_r^2 \sin(w_r t) + k_p(q - q_d(t)) + k_d(\dot{q} - \dot{q}_d(t)) - [0, 1]\boldsymbol{P}(\boldsymbol{x} - \boldsymbol{x}_d)/(2\varepsilon) \quad (24)$$

with control gains $k_p = 5$, $k_d = 10$ and $\varepsilon = 1$. The matrix $\boldsymbol{P}$ is obtained by solving the Lyapunov equation with $\boldsymbol{Q} = \boldsymbol{I}_2$. The two variants of SkyGP are with $\bar{N} = 50$, $W = 10$, $\Gamma = 1$ and the maximal data size set to 100. Other hyperparameters are set the same in Section 5.1. Furthermore, the desired reference is chosen with the form $q_r(t) = a_r \sin(w_r t)$ with coefficients $a_r = 1, w_r = 0.1$.

The maximum norm values of tracking errors $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{x}_d$ and prediction errors $f(\boldsymbol{x}) - \tilde{\mu}(\boldsymbol{x})$ are shown in Figure 2. The box plot summarizes the distribution of the performance on prediction error and tracking error over 100 trials with a random initial state $\boldsymbol{x}(0)$ uniformly distributed in $[0, 1]^2$. The proposed SkyGP variants demonstrate both lower average and median in prediction and tracking errors, indicating more accurate and consistent learning and control behavior compared to all baselines. Particularly, the prediction and tracking errors of SkyGP-Dense are lower than those of SkyGP-Fast with sufficiently fast computation time, which is consistent with the regression benchmarks in Section 5.1. These results reinforce the advantages of the SkyGP framework in closed-loop dynamical systems, where both precision and safety are critical.

6 Conclusion and Discussion

In this paper, we introduced SkyGP for scalable and adaptive learning. Designed to address the computational and memory bottlenecks of exact GPs in real-time settings, SkyGP maintains a bounded set of experts while preserving strong predictive performance with uncertainty quantification. SkyGP-Fast, targeted for computational efficiency, and SkyGP-Dense, designed for high prediction accuracy. Our extensive empirical evaluation across multiple real-world re-

gression datasets and real-time control tasks demonstrates the superiority of SkyGP over SOTA baselines.

A limitation of the center update rule in Equation (4) is that it may fail to accurately capture special data distributions, such as annular or multimodal patterns. We adopt the method from (Nguyen-tuong, Peters, and Seeger 2008) for its simplicity and computational efficiency. Another limitation lies in the use of the dynamic table list, which may struggle to handle data with sudden distributional shifts, potentially affecting the timely allocation or reuse of experts.

Acknowledgements

The authors acknowledge the financial support by the Federal Ministry of Education and Research of Germany in the programme of "Souverän. Digital. Vernetzt." under joint project 6G-life with project identification number: 16KISK002.

References

Bui, T. D.; Nguyen, C.; and Turner, R. E. 2017. Streaming Sparse Gaussian Process Approximations. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Cao, Y.; and Fleet, D. J. 2015. Generalized Product of Experts for Automatic and Principled Fusion of Gaussian Process Predictions. arXiv:1410.7827.

Cohen, S.; Mbuvha, R.; Marwala, T.; and Deisenroth, M. 2020. Healing Products of Gaussian Process Experts. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 2068–2077. PMLR.

Dai, X.; Lederer, A.; Yang, Z.; and Hirche, S. 2023. Can Learning Deteriorate Control? Analyzing Computational Delays in Gaussian Process-Based Event-Triggered Online Learning. In Matni, N.; Morari, M.; and Pappas, G. J., eds., *Proceedings of The 5th Annual Learning for Dynamics and Control Conference*, volume 211 of *Proceedings of Machine Learning Research*, 445–457. PMLR.

Dai, X.; Yang, Z.; Xu, M.; Zhang, S.; Liu, F.; Hattab, G.; and Hirche, S. 2024. Decentralized event-triggered online learning for safe consensus control of multi-agent systems with Gaussian process regression. *European Journal of Control*, 80: 101058. 2024 European Control Conference Special Issue.

Dai, X.; Yang, Z.; Zhang, S.; Zhai, D.-H.; Xia, Y.; and Hirche, S. 2025. Cooperative Online Learning for Multiagent System Control via Gaussian Processes With Event-Triggered Mechanism. *IEEE Transactions on Neural Networks and Learning Systems*, 36(7): 13304–13318.

Deisenroth, M.; and Ng, J. W. 2015. Distributed Gaussian Processes. In Bach, F.; and Blei, D., eds., *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, 1481–1490. Lille, France: PMLR.

Gijsberts, A.; and Metta, G. 2013. Real-time model learning using Incremental Sparse Spectrum Gaussian Process Regression. *Neural Networks*, 41: 59–69. Special Issue on Autonomous Learning.

Hashimoto, K.; Saoud, A.; Kishida, M.; Ushio, T.; and Dimarogonas, D. V. 2022. Learning-based Symbolic Abstractions for Nonlinear Control Systems. *Automatica*, 146: 110646.

Khalil, H. K. 2002. *Nonlinear Systems*. Prentice-Hall.

Lederer, A.; Begzadić, A.; Hirche, S.; Cortés, J.; and Herbert, S. 2024. Safe barrier-constrained control of uncertain systems via event-triggered learning. *arXiv preprint arXiv:2408.16144*.

Lederer, A.; Conejo, A. J. O.; Maier, K. A.; Xiao, W.; Umlauft, J.; and Hirche, S. 2021. Gaussian Process-Based Real-Time Learning for Safety Critical Applications. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, 6055–6064. PMLR.

Lederer, A.; Umlauft, J.; and Hirche, S. 2019. Uniform error bounds for Gaussian process regression with application to safe control. *Advances in Neural Information Processing Systems*, 32.

Lederer, A.; Yang, Z.; Jiao, J.; and Hirche, S. 2023. Cooperative Control of Uncertain Multiagent Systems via Distributed Gaussian Processes. *IEEE Transactions on Automatic Control*, 68(5): 3091–3098.

Liu, H.; Cai, J.; Wang, Y.; and Ong, Y. S. 2018. Generalized Robust Bayesian Committee Machine for Large-scale Gaussian Process Regression. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 3131–3140. PMLR.

Maddalena, E. T.; Scharnhorst, P.; and Jones, C. N. 2021. Deterministic Error Bounds for Kernel-based Learning Techniques under Bounded Noise. *Automatica*, 134: 109896.

Naish-guzman, A.; and Holden, S. 2007. The Generalized FITC Approximation. In Platt, J.; Koller, D.; Singer, Y.; and Roweis, S., eds., *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc.

Nguyen-tuong, D.; Peters, J.; and Seeger, M. 2008. Local Gaussian Process Regression for Real Time Online Model Learning and Control. In Koller, D.; Schuurmans, D.; Bengio, Y.; and Bottou, L., eds., *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc.

Rasmussen, C. E.; and Williams, C. K. I. 2006. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press. ISBN 978-0-262-18253-9.

Reeb, D.; Doerr, A.; Gerwinn, S.; and Rakitsch, B. 2018. Learning Gaussian Processes by Minimizing PAC-Bayesian Generalization Bounds. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Scharnhorst, P.; Maddalena, E. T.; Jiang, Y.; and Jones, C. N. 2022. Robust Uncertainty Bounds in Reproducing Kernel Hilbert Spaces: A Convex Optimization Approach. *IEEE Transactions on Automatic Control*, 68(5): 2848–2861.

Schürch, M.; Azzimonti, D.; Benavoli, A.; and Zaffalon, M. 2023. Correlated product of experts for sparse Gaussian process regression. *Machine Learning*, 112(5): 1411–1432.

Snelson, E.; and Ghahramani, Z. 2005. Sparse Gaussian Processes using Pseudo-inputs. In Weiss, Y.; Schölkopf, B.; and Platt, J., eds., *Advances in Neural Information Processing Systems*, volume 18. MIT Press.

Srinivas, N.; Krause, A.; Kakade, S. M.; and Seeger, M. W. 2012. Information-theoretic Regret Bounds for Gaussian Process Optimization in the Bandit Setting. *IEEE Transactions on Information Theory*, 58(5): 3250–3265.

Trapp, M.; Peharz, R.; Pernkopf, F.; and Rasmussen, C. E. 2020. Deep Structured Mixtures of Gaussian Processes. In Chiappa, S.; and Calandra, R., eds., *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, 2251–2261. PMLR.

Tresp, V. 2000a. A Bayesian committee machine. *Neural computation*, 12(11): 2719–2741.

Tresp, V. 2000b. Mixtures of Gaussian Processes. In Leen, T.; Dietterich, T.; and Tresp, V., eds., *Advances in Neural Information Processing Systems*, volume 13. MIT Press.

Wachi, A.; Sui, Y.; Yue, Y.; and Ono, M. 2018. Safe Exploration and Optimization of Constrained MDPs Using Gaussian Processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Yang, Z.; Dai, X.; Dubey, A.; Hirche, S.; and Hattab, G. 2024a. Whom to Trust? Elective Learning for Distributed Gaussian Process Regression. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '24, 2020–2028. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9798400704864.

Yang, Z.; Dai, X.; Fang, L.; Zhou, J.; and Yan, Z. 2025. Safe event-triggered control of unmanned surface vehicles with Gaussian processes: Resilience in denial of service attacks and uncertain dynamics. *Engineering Applications of Artificial Intelligence*, 153: 110776.

Yang, Z.; Dai, X.; and Hirche, S. 2025. Asynchronous Distributed Gaussian Process Regression. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(21): 22065–22073.

Yang, Z.; Dong, S.; Lederer, A.; Dai, X.; Chen, S.; Sosnowski, S.; Hattab, G.; and Hirche, S. 2024b. Cooperative Learning with Gaussian Processes for Euler-Lagrange Systems Tracking Control Under Switching Topologies. In *2024 American Control Conference (ACC)*, 560–567.

Yang, Z.; Sosnowski, S.; Liu, Q.; Jiao, J.; Lederer, A.; and Hirche, S. 2021. Distributed Learning Consensus Control for Unknown Nonlinear Multi-Agent Systems based on Gaussian Processes. In *2021 60th IEEE Conference on Decision and Control (CDC)*, 4406–4411.

Yuan, C.; and Neubauer, C. 2008. Variational Mixture of Gaussian Process Experts. In Koller, D.; Schuurmans, D.; Bengio, Y.; and Bottou, L., eds., *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc.

Yuan, Z.; and Zhu, M. 2024. Lightweight Distributed Gaussian Process Regression for Online Machine Learning. *IEEE Transactions on Automatic Control*, 69(6): 3928–3943.

A Proofs

A.1 Proof of Lemma 1

Proof. Following the prediction error bound for single GP expert in (Lederer, Umlauft, and Hirche 2019), the error between $f(\cdot)$ and $\mu_i(\cdot)$ is probabilistic bounded by

$$|f(\boldsymbol{x}) - \mu_i(\boldsymbol{x})| \leq \beta\sigma_i(\boldsymbol{x}) + \gamma_i, \qquad \forall \boldsymbol{x} \in \mathbb{X} \tag{25}$$

with a probability of at least $1-\delta$. Considering that the weights $\omega_i(\cdot)$ satisfy $\sum_{i\in\mathcal{N}} \omega_i(\boldsymbol{x}) = 1$, the aggregated prediction error is written as

$$\begin{aligned} |f(\boldsymbol{x}) - \tilde{\mu}_i(\boldsymbol{x})| =& |f(\boldsymbol{x}) - \sum\nolimits_{i\in\mathcal{N}} \omega_i(\boldsymbol{x})\mu_i(\boldsymbol{x})| && (26)\\ \leq& \sum\nolimits_{i\in\mathcal{N}} \omega_i(\boldsymbol{x})|f(\boldsymbol{x}) - \mu_i(\boldsymbol{x})|. && (27)\end{aligned}$$

Apply the prediction error bound in (25)

$$\begin{aligned} |f(\boldsymbol{x}) - \tilde{\mu}_i(\boldsymbol{x})| \leq& \sum\nolimits_{i\in\mathcal{N}} \omega_i(\boldsymbol{x})(\beta\sigma_i(\boldsymbol{x}) + \gamma_i) && (28)\\ =& \beta\sum\nolimits_{i\in\mathcal{N}} \omega_i(\boldsymbol{x})\sigma_i(\boldsymbol{x}) + \sum\nolimits_{i\in\mathcal{N}} \omega_i(\boldsymbol{x})\gamma_i \end{aligned}$$

with a probability of at least $1-|\mathcal{N}|\delta$ using the union bound. Considering $|\mathcal{N}| \leq \bar{\mathcal{N}}$, the result in Lemma 1 is derived, concluding the proof. □

A.2 Proof of Theorem 1

Proof. Note that the difference between $\boldsymbol{f}(\cdot,\cdot)$ and $\boldsymbol{\mu}(\cdot,\cdot)$ leads to the bounded time derivative of $V(\cdot,\cdot)$ as

$$\begin{aligned} \dot{V}(t,\boldsymbol{z}) =& \nabla_t V(t,\boldsymbol{x}) && (29)\\ &+ \nabla_{\boldsymbol{x}} V(t,\boldsymbol{x})(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\pi}(t,\boldsymbol{x},\boldsymbol{\mu})) + \boldsymbol{g}(\boldsymbol{x},\boldsymbol{\pi}(t,\boldsymbol{x},\boldsymbol{\mu}))) \\ \leq& \nabla_t V(t,\boldsymbol{x}) && (30)\\ &+ \nabla_{\boldsymbol{x}} V(t,\boldsymbol{x})(\boldsymbol{\mu}(\boldsymbol{x},\boldsymbol{\pi}(t,\boldsymbol{x},\boldsymbol{\mu})) + \boldsymbol{g}(\boldsymbol{x},\boldsymbol{\pi}(t,\boldsymbol{x},\boldsymbol{\mu}))) \\ &+ \|\nabla_{\boldsymbol{x}} V(t,\boldsymbol{x})\|\|\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\pi}(t,\boldsymbol{x},\boldsymbol{\mu})) - \boldsymbol{\mu}(\boldsymbol{x},\boldsymbol{\pi}(t,\boldsymbol{x},\boldsymbol{\mu}))\| \\ \leq& -\alpha(V(t,\boldsymbol{z})) - \|\nabla_{\boldsymbol{x}} V(t,\boldsymbol{x})\|^2/(4\varepsilon) && (31)\\ &+ \|\nabla_{\boldsymbol{x}} V(t,\boldsymbol{x})\|\|\boldsymbol{\eta}(\boldsymbol{\xi}(t,\boldsymbol{x},\boldsymbol{\mu}))\| \end{aligned}$$

where $\boldsymbol{\xi}(t,\boldsymbol{x},\boldsymbol{\mu}) = [\boldsymbol{x}^T, \boldsymbol{\pi}^T(t,\boldsymbol{x},\boldsymbol{\mu})]^T$ and $\boldsymbol{\eta}(\cdot) = [\eta_1(\cdot),\cdots,\eta_{d_x}(\cdot)]^T$ with $\eta_i(\cdot)$ obtained using prediction error bound with kernel $\kappa_i(\cdot,\cdot)$. Moreover, applying Young's inequality with $\varepsilon \in \mathbb{R}_+$ on the third term inducing

$$\begin{aligned} \|\nabla_{\boldsymbol{x}} V(t,\boldsymbol{x})\|\|\boldsymbol{\eta}(\boldsymbol{\xi}(t,\boldsymbol{x},\boldsymbol{\mu}))\| \leq& \|\nabla_{\boldsymbol{x}} V(t,\boldsymbol{x})\|^2/(4\varepsilon) && (32)\\ &+ \varepsilon\|\boldsymbol{\eta}(\boldsymbol{\xi}(t,\boldsymbol{x},\boldsymbol{\mu}))\|^2, \end{aligned}$$

then the bound of $\dot{V}(t,\boldsymbol{z})$ is further written as

$$\begin{aligned} \dot{V}(t,\boldsymbol{z}) \leq& -\alpha(V(t,\boldsymbol{z})) + \varepsilon\|\boldsymbol{\eta}(\boldsymbol{\xi}(t,\boldsymbol{x},\boldsymbol{\mu}))\|^2 && (33)\\ \leq& -\alpha(V(t,\boldsymbol{z})) + \varepsilon\bar{\eta}^2 && (34)\end{aligned}$$

with $\bar{\eta} = \max_{t\in\mathbb{R}_{0,+},\boldsymbol{x}\in\mathbb{X}} \|\boldsymbol{\eta}(\boldsymbol{\xi}(t,\boldsymbol{x},\boldsymbol{\mu}))\|$. Therefore, it is easy to derive the ultimate bound of $V(t,\boldsymbol{z}(t))$ as

$$\lim_{t\to\infty} V(t,\boldsymbol{z}(t)) \leq \alpha^{-1}(\varepsilon\bar{\eta}^2), \tag{35}$$

which leads to the bound of $\boldsymbol{z}(t)$ as

$$\begin{aligned} \lim_{t\to\infty} \|\boldsymbol{z}(t)\| \leq& \lim_{t\to\infty} \underline{\alpha}^{-1}(V(t,\boldsymbol{z}(t))) && (36)\\ \leq& \underline{\alpha}^{-1}(\alpha^{-1}(\varepsilon\bar{\eta}^2)), && (37)\end{aligned}$$

concluding the proof. □

A.3 Proof of Theorem 2

Proof. It is easy to see the existence of class-$\mathcal{K}$ functions $\underline{\alpha}(\cdot)$ and $\bar{\alpha}(\cdot)$ as

$$\underline{\alpha}(\|\boldsymbol{z}\|) = \underline{\lambda}(\boldsymbol{P})\|\boldsymbol{z}\|^2, \quad \bar{\alpha}(\|\boldsymbol{z}\|) = \bar{\lambda}(\boldsymbol{P})\|\boldsymbol{z}\|^2 \tag{38}$$

using the control law (21). Moreover, apply (21) to (20), the controlled dynamics is written as

$$\begin{aligned} \ddot{\boldsymbol{q}} =& \boldsymbol{M}^{-1}(\boldsymbol{q})(\boldsymbol{u} - \boldsymbol{C}(\boldsymbol{x})\dot{\boldsymbol{q}} - \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{d}(\boldsymbol{x})) \\ =& \boldsymbol{M}^{-1}(\boldsymbol{q})(\boldsymbol{d}(\boldsymbol{x}) - \boldsymbol{\mu}(\boldsymbol{x}) - \boldsymbol{B}^T(\boldsymbol{x})\boldsymbol{P}\boldsymbol{z}/(2\varepsilon)) \\ &+ [\boldsymbol{K}_p, \boldsymbol{K}_d]\boldsymbol{z} + \ddot{\boldsymbol{q}}_d(t), && (39)\end{aligned}$$

such that the dynamics w.r.t $\boldsymbol{z}$ becomes

$$\dot{\boldsymbol{z}} = \boldsymbol{A}\boldsymbol{z} - \boldsymbol{B}(\boldsymbol{x})\boldsymbol{B}^T(\boldsymbol{x})\boldsymbol{P}\boldsymbol{z}/(2\varepsilon) + \boldsymbol{B}(\boldsymbol{x})(\boldsymbol{d}(\boldsymbol{x}) - \boldsymbol{\mu}(\boldsymbol{x})).$$

Investigate the time derivative of $V(t,\boldsymbol{z})$ as

$$\begin{aligned} \dot{V}(t,\boldsymbol{z}) =& \boldsymbol{z}^T(\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A})\boldsymbol{z} + 2\boldsymbol{z}^T\boldsymbol{P}\boldsymbol{B}(\boldsymbol{x})(\boldsymbol{d}(\boldsymbol{x}) - \boldsymbol{\mu}(\boldsymbol{x})) \\ &- 2\boldsymbol{z}^T\boldsymbol{P}\boldsymbol{B}(\boldsymbol{x})\boldsymbol{B}^T(\boldsymbol{x})\boldsymbol{P}\boldsymbol{z}/(2\varepsilon) && (40)\\ \leq& -\boldsymbol{z}^T\boldsymbol{Q}\boldsymbol{z} + \|\boldsymbol{B}^T(\boldsymbol{x})\boldsymbol{P}\boldsymbol{z}\|^2/\varepsilon + \varepsilon\|\boldsymbol{d}(\boldsymbol{x}) - \boldsymbol{\mu}(\boldsymbol{x})\|^2 \\ &- \|\boldsymbol{B}^T(\boldsymbol{x})\boldsymbol{P}\boldsymbol{z}\|^2/\varepsilon && (41)\end{aligned}$$

using Young's inequality. Moreover, considering the relaxation with eigenvalues and prediction error bound for GP, the upper bound of $\dot{V}(t,\boldsymbol{z})$ becomes

$$\begin{aligned} \dot{V}(t,\boldsymbol{z}) \leq& -\underline{\lambda}(\boldsymbol{Q})\|\boldsymbol{z}\|^2 + \varepsilon\|\boldsymbol{\eta}(\boldsymbol{x})\|^2 && (42)\\ \leq& -\underline{\lambda}(\boldsymbol{Q})\bar{\lambda}^{-1}(\boldsymbol{P})V(t,\boldsymbol{z}) + \varepsilon\bar{\eta}^2. && (43)\end{aligned}$$

with a probability of at least $1-d_q\delta$. According to the previous result, it is directly derived that

$$\lim_{t\to\infty} \|\boldsymbol{z}(t)\| \leq \frac{\varepsilon\bar{\lambda}(\boldsymbol{P})}{\underline{\lambda}(\boldsymbol{P})\underline{\lambda}(\boldsymbol{Q})}\bar{\eta}^2, \tag{44}$$

which concludes the proof. □

B Additional Results

B.1 Simulation Setup

The codes are executed in Python 3.10.12 on the laptop with Intel i5-12500H with 16.0GB RAM.

B.2 Regression Benchmark

We evaluated the performance of the LoG-GP, LocalGPs, ISSGP, SSGP, using the following parameters:

- LoG-GP: Each expert has $K = 2$ children with $\bar{\mathcal{N}} = 50$.
- LocalGP: Each expert is with $\bar{\mathcal{N}} = 50$. We use the divided threshold as 0.9 in (Nguyen-tuong, Peters, and Seeger 2008).
- ISSGP: We use $D = 200$ random Fourier features with as the same in (Gijsberts and Metta 2013).
- SSGP: We use 100 inducing points for initial optimization. For online optimization of hyperparameters and inducing points, we use the position of the inducing points every 500 samples using 50 optimization steps per batch, in order to balance both predictive accuracy and computational efficiency.

We use the average prediction and update times as a performance metric, as well as the standardized mean squared error (SMSE) and the mean standardized log loss (MSLL) following the sequential evaluation protocol described in (Rasmussen and Williams 2006).

$$\text{SMSE}_k = \frac{1}{k s_y^2} \sum_{i=1}^{k} \left(\hat{\mu}_{k-1}(\boldsymbol{x}^i) - y^i \right)^2, \tag{45}$$

where $\hat{\mu}_{k-1}(\boldsymbol{x}^i)$ denotes the predictive mean after observing the first $k-1$ data points, and s_y^2 is the empirical variance of the ground truth labels y^i.

In Table 3, it is shown that the proposed SkyGP-Fast variant, consistently achieves the lowest model update times of 0.04 ms on both the SARCOS and ELECTRIC datasets and the lowest prediction times of 0.16 ms and 0.14 ms, respectively. Although LoGGP attains the fastest prediction time on the KIN40K and PUMA datasets, it falls short in model update time compared to SkyGP-Fast. Notably, all SkyGP variants maintain better performance across all datasets, even as the number of active experts increases, compared to the baselines, i.e., SSGP, ISSGP, and LocalGPs, which exhibit significantly higher computational overhead. The SkyGP-Dense variant provides improvements in predictive accuracy due to its data replacement strategy, while incurring only a marginal increase in computational cost—making it a compelling option when accuracy is prioritized. Overall, the results highlight SkyGP's superior scalability and practical utility for streaming online learning.

To visualize the performance comparison across all methods, we plot the SMSE and MSLL trends over simulation iterations in Figure 3 to 10. On the SARCOS dataset, both SkyGP-Fast and SkyGP-Dense with $\bar{\mathcal{N}} = 2, 4$ consistently achieve lower SMSE than the state-of-the-art LoGGP method. While SkyGP-Fast with $\bar{\mathcal{N}} = 1$ initially exhibits higher error before iteration 1500, it surpasses LoGGP in accuracy in later stages. Similarly, SkyGP-Dense outperforms LoGGP after around 500 iterations. Compared to SSGP, only the SkyGP-Dense variants with $\bar{\mathcal{N}} = 2, 4$ achieve better performance, however, SSGP suffers from high computational costs that hinder its practical use in real-time scenarios. ISSGP performs the worst among all evaluated methods on the SARCOS dataset. While LocalGPs with $\bar{\mathcal{N}} = 1$ show comparable or slightly better performance than SkyGP-Fast with the same number of experts, both SkyGP-Fast and SkyGP-Dense using gPoE and rBCM aggregation strategies outperform LocalGPs when $\bar{\mathcal{N}} = 2, 4$.

Similar to the observations on the SARCOS dataset, both SkyGP-Fast and SkyGP-Dense with $\bar{\mathcal{N}} = 2, 4$ consistently achieve lower SMSE on the PUMA dataset compared to the state-of-the-art LoGGP and LocalGPs methods. While SSGP and ISSGP outperform all SkyGP variants with $\bar{\mathcal{N}} = 1$, SkyGP-Dense with $\bar{\mathcal{N}} = 4$ surpasses ISSGP and achieves comparable performance to SSGP. As with LoGGP, LocalGPs generally perform better than SkyGP when $\bar{\mathcal{N}} = 1$. However, although LocalGPs with $\bar{\mathcal{N}} = 2$ show slightly better or comparable results to SkyGP-Fast with the same number of experts, both SkyGP-Fast and SkyGP-Dense outperform LocalGPs when $\bar{\mathcal{N}} = 4$.

In the KIN40K dataset, SkyGP with a single expert does not outperform the baseline methods. However, when $\bar{\mathcal{N}} = 4$, SkyGP-Dense with gPoE and rBCM aggregation strategies achieves superior performance compared to all other methods. SkyGP-Fast with $\bar{\mathcal{N}} = 4$ outperforms SSGP and shows comparable results to ISSGP.

On the ELECTRIC dataset, SkyGP with a single expert ($\bar{\mathcal{N}} = 1$) outperforms LoG-GP but falls short compared to ISSGP, and performs similarly to LocalGPs with the same number of experts. However, when the number of experts increases to $\bar{\mathcal{N}} = 2$ or 4, only the gPoE and rBCM aggregation strategies in SkyGP lead to improved performance over LoG-GP. A similar pattern is observed in LocalGPs, where only certain aggregation strategies with $\bar{\mathcal{N}} = 2, 4$ yield competitive or superior results compared to LoG-GP. Notably, due to the excessively high prediction and model update time of SSGP on the ELECTRIC dataset, we omit its results from the simulation.

In summary, SkyGP variants demonstrate consistently strong prediction performance across multiple datasets, particularly when using multiple experts ($\bar{\mathcal{N}} = 2, 4$) and advanced aggregation strategies such as gPoE and rBCM. On the SARCOS and PUMA datasets, both SkyGP-Fast and SkyGP-Dense outperform the SOTA LoGGP and LocalGPs methods in terms of SMSE, especially as the number of experts increases. Although SkyGP with a single expert underperforms compared to SSGP and ISSGP, SkyGP-Dense with $\bar{\mathcal{N}} = 4$ achieves comparable or superior results while maintaining computational efficiency. On the KIN40K dataset, SkyGP-Dense with four experts and appropriate aggregation outperforms all baselines, while SkyGP-Fast matches ISSGP and surpasses SSGP. In the ELECTRIC dataset, SkyGP with one expert performs better than LoGGP but lags behind ISSGP. Only gPoE and rBCM aggregation in both SkyGP and LocalGPs lead to meaningful improvements when the number of experts increases. Overall, SkyGP achieves a favorable balance between accuracy and scalability, particularly with multiple experts and efficient aggregation.

In terms of MSLL on the SARCOS dataset, SkyGP-Dense with $\bar{\mathcal{N}} = 4$ outperforms all baseline models, including SkyGP-Fast. SkyGP-Fast surpasses LoG-GP after approximately 500 iterations and outperforms ISSGP after around 2500 iterations, showing comparable performance to LocalGPs. When $\bar{\mathcal{N}} = 2$, SkyGP-Dense with the rBCM aggregation strategy achieves lower MSLL than LoG-GP, SSGP, and LocalGPs, and begins to outperform ISSGP after 2500 iterations. Notably, while both SkyGP-Dense and SkyGP-Fast outperform LoG-GP after 500 and 2500 iterations, respectively. Only SkyGP-Dense demonstrates competitive performance relative to LocalGPs. On the PUMA dataset, SkyGP variants outperform LoG-GP when using $\bar{\mathcal{N}} = 2$ or 4, and even SkyGP-Dense with a single expert ($\bar{\mathcal{N}} = 1$) achieves better performance than LoG-GP. Compared to SSGP, all SkyGP variants demonstrate similar performance, though they do not surpass ISSGP. When $\bar{\mathcal{N}} = 2$ or 4, both SkyGP-Dense and SkyGP-Fast outperform LocalGPs, whereas with only one expert, they perform worse than LocalGPs. On the KIN40K dataset, all SkyGP variants outperform the

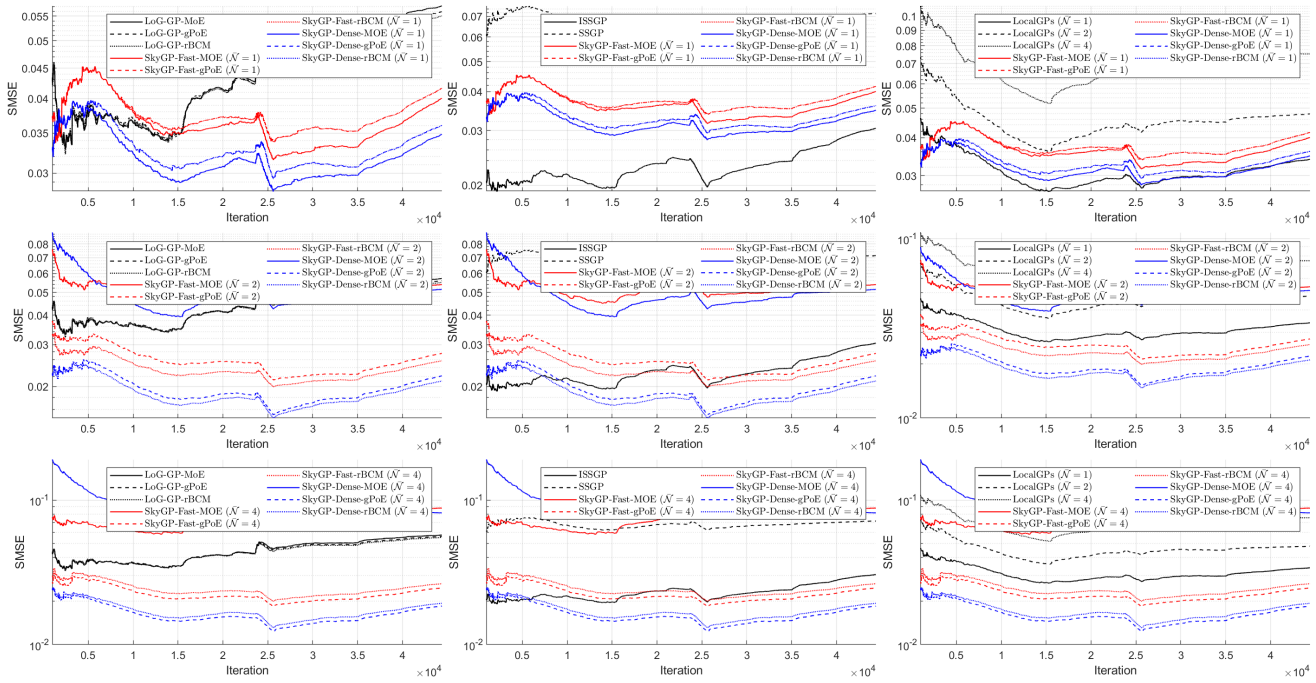


Figure 3: SMSE comparison on SARCOS dataset against all baselines.

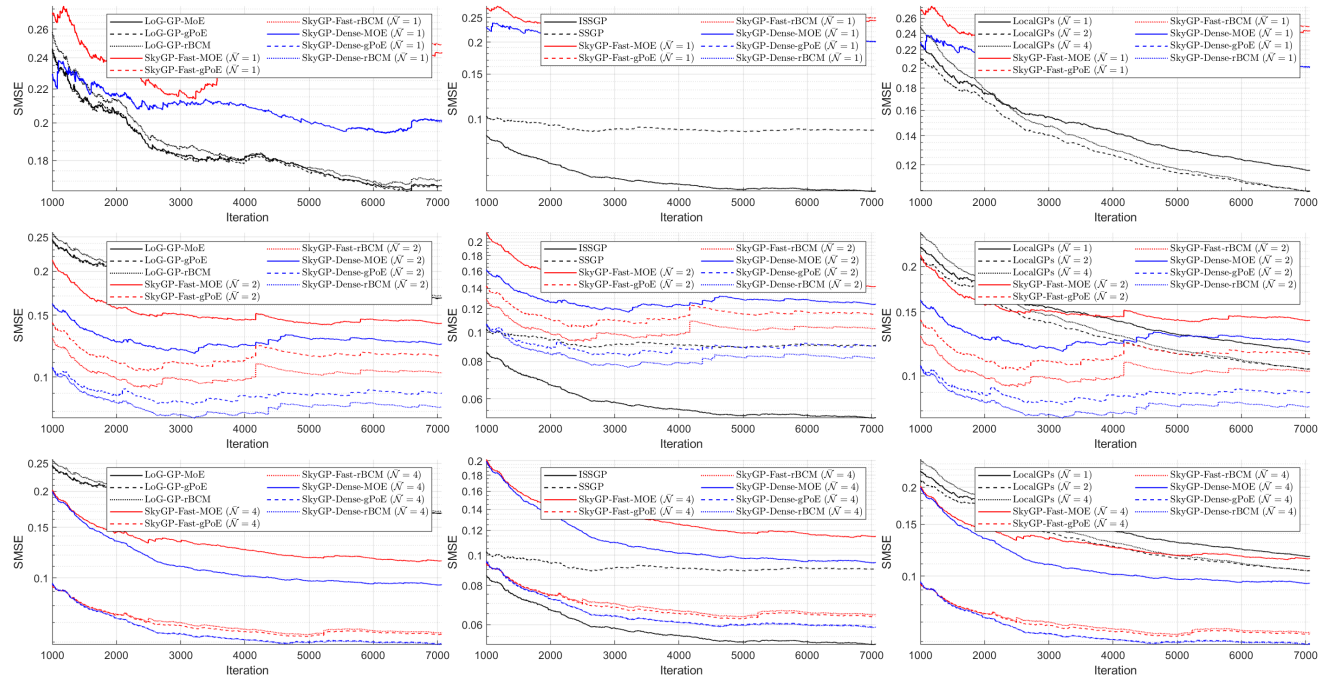


Figure 4: SMSE comparison on PUMA dataset against all baselines.

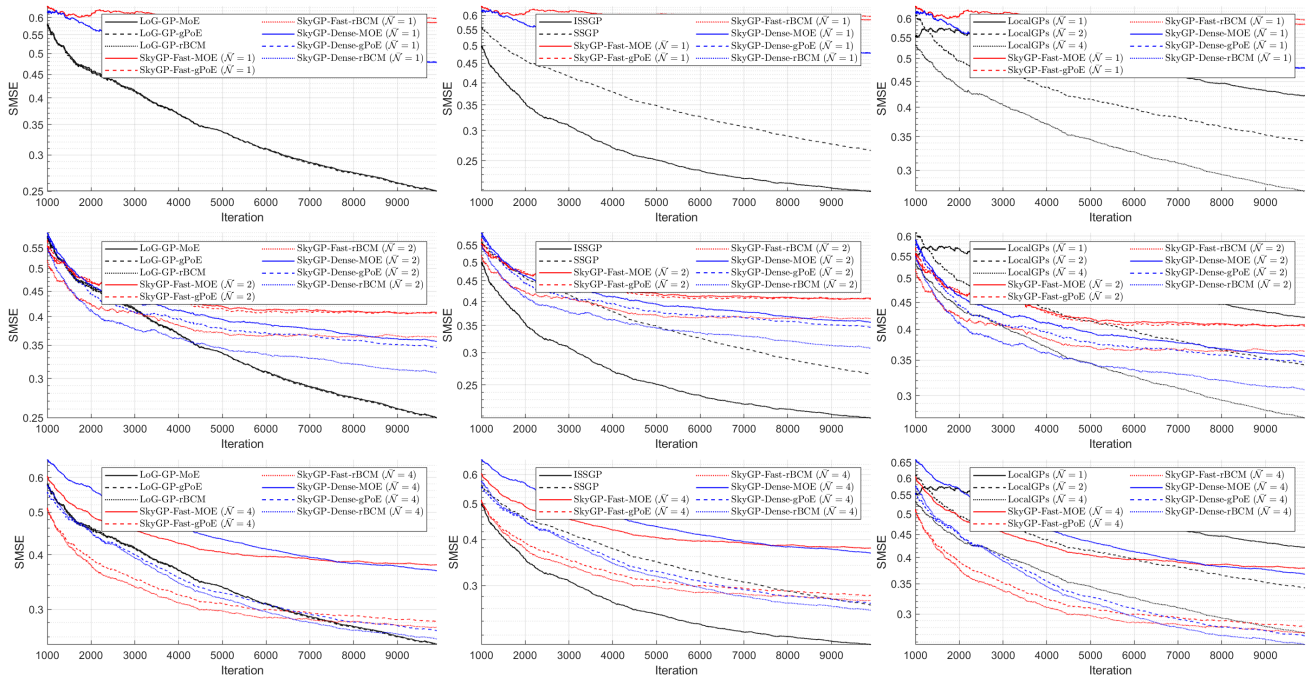


Figure 5: SMSE comparison on KIN40K dataset against all baselines.

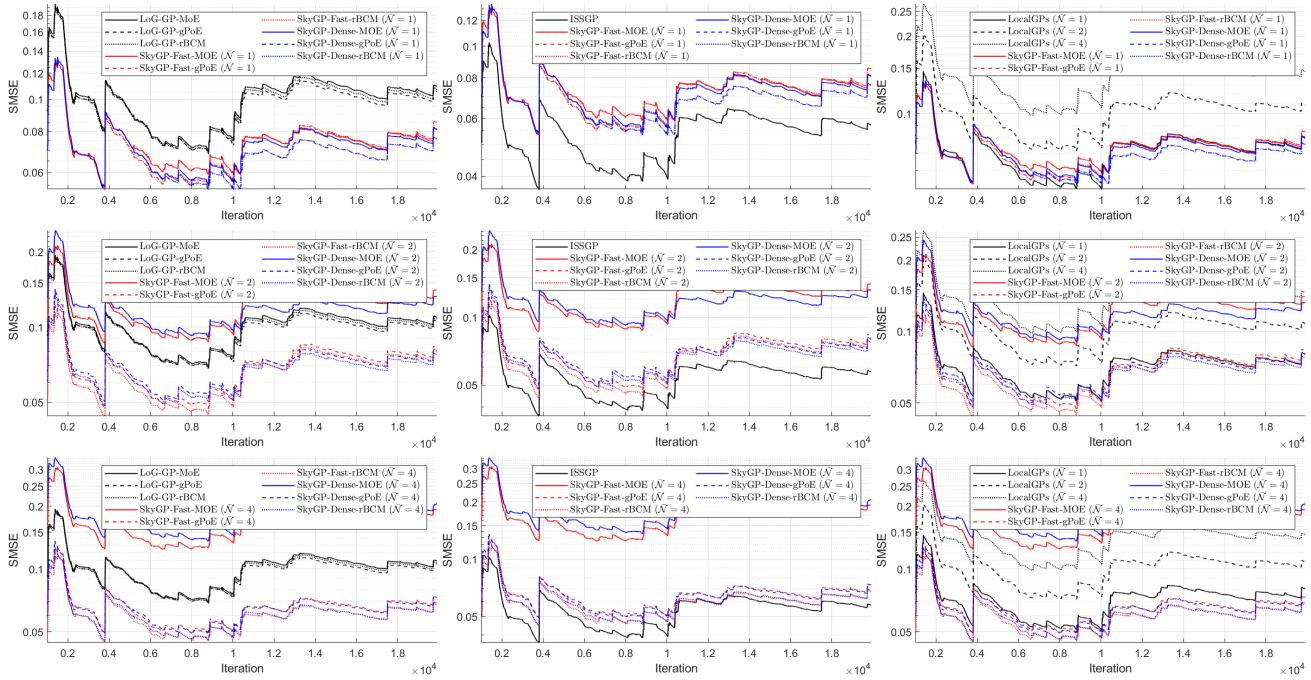


Figure 6: SMSE comparison on ELECTRIC dataset against all baselines.

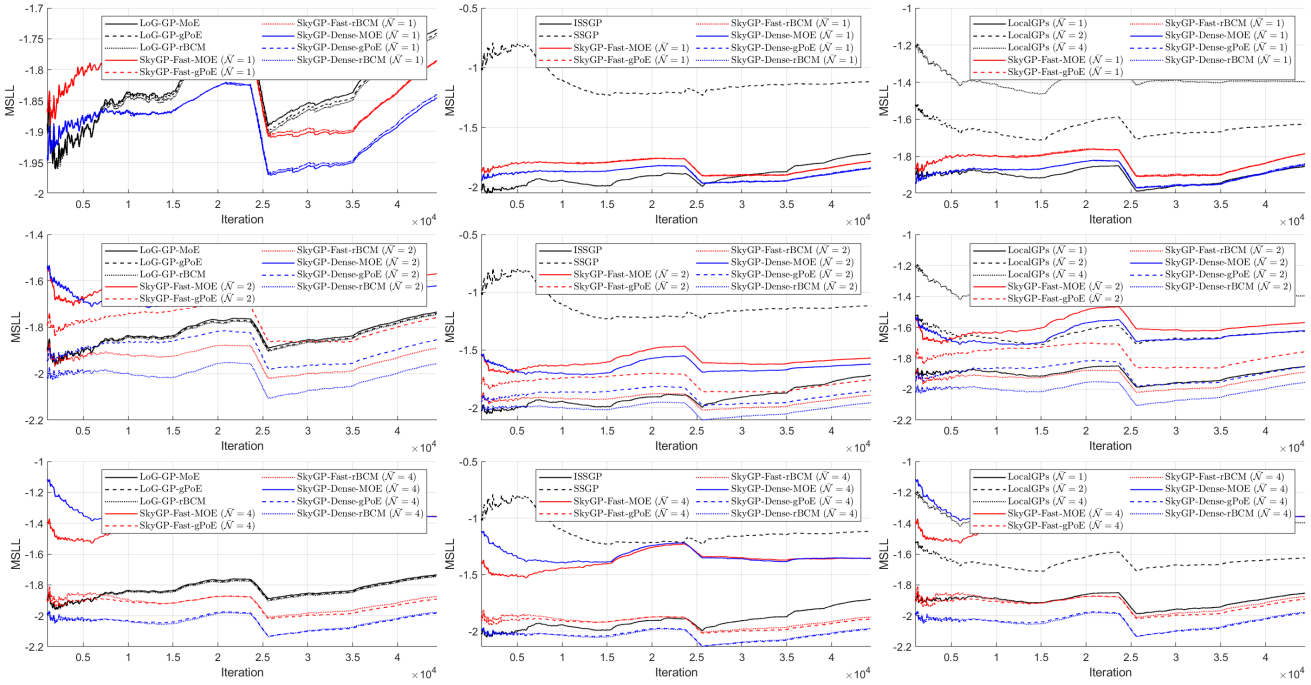


Figure 7: MSLL comparison on SARCOS dataset against all baselines.

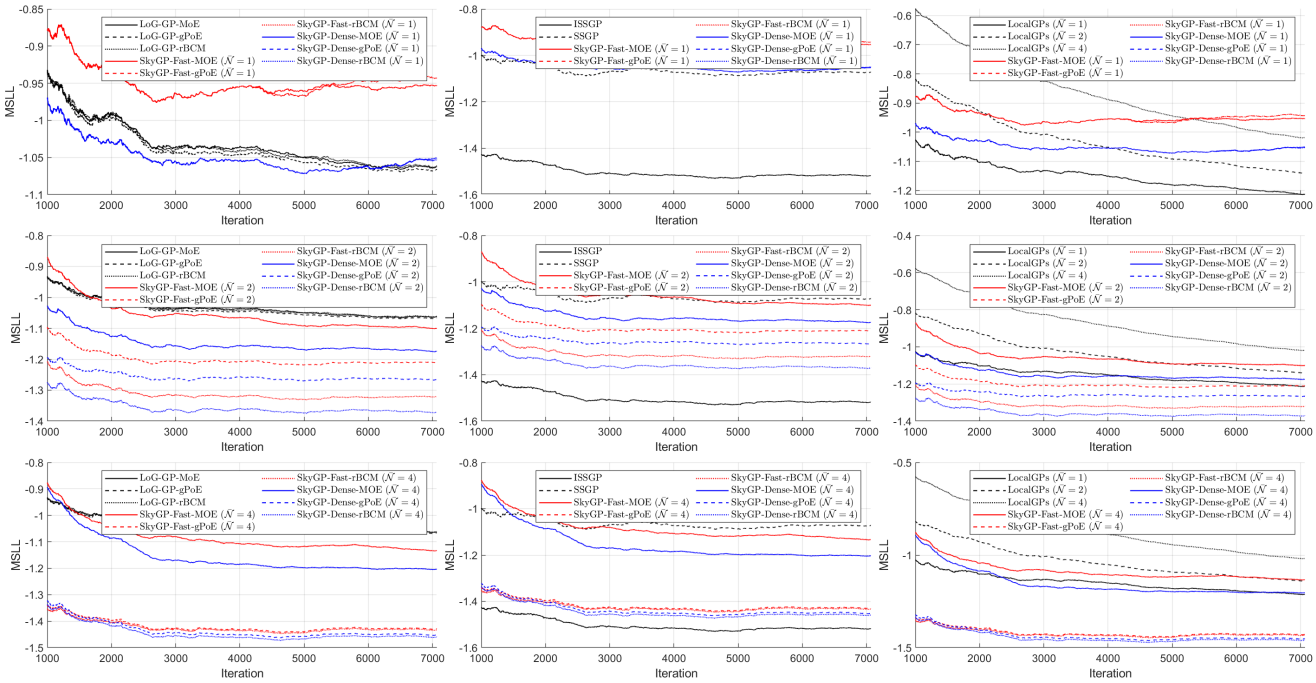


Figure 8: MSLL comparison on PUMA dataset against all baselines.

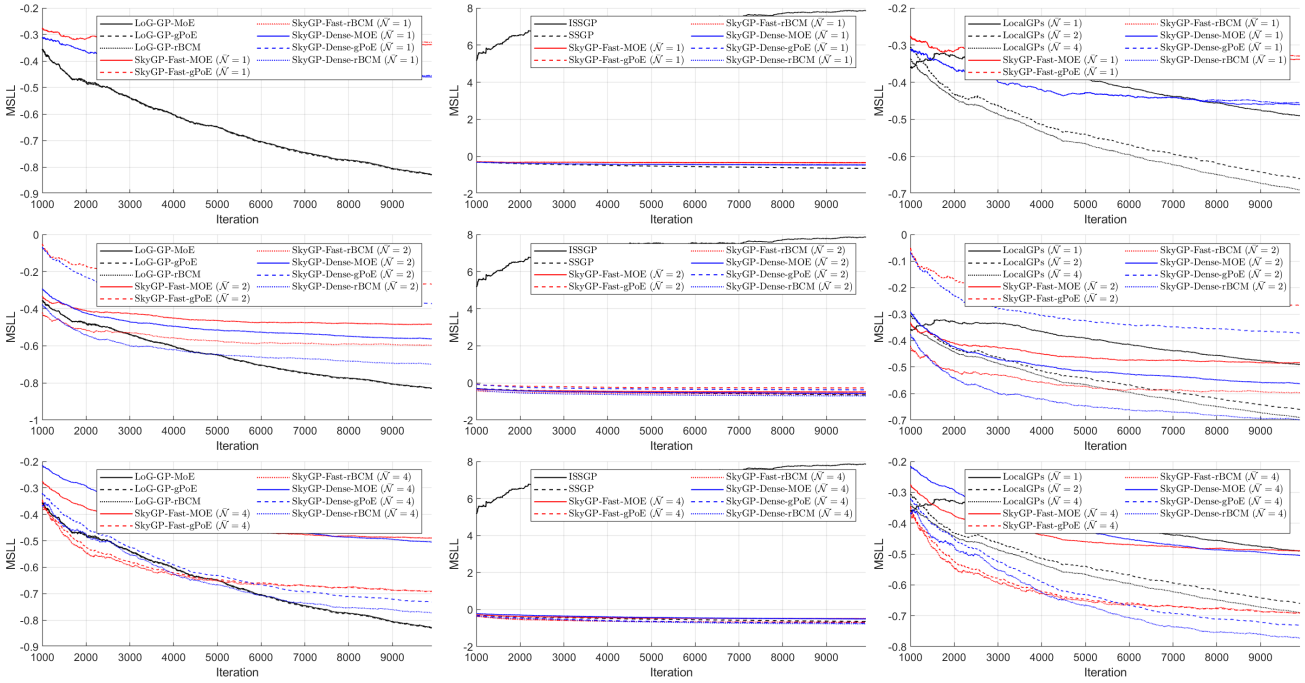


Figure 9: MSLL comparison on KIN40K dataset against all baselines.

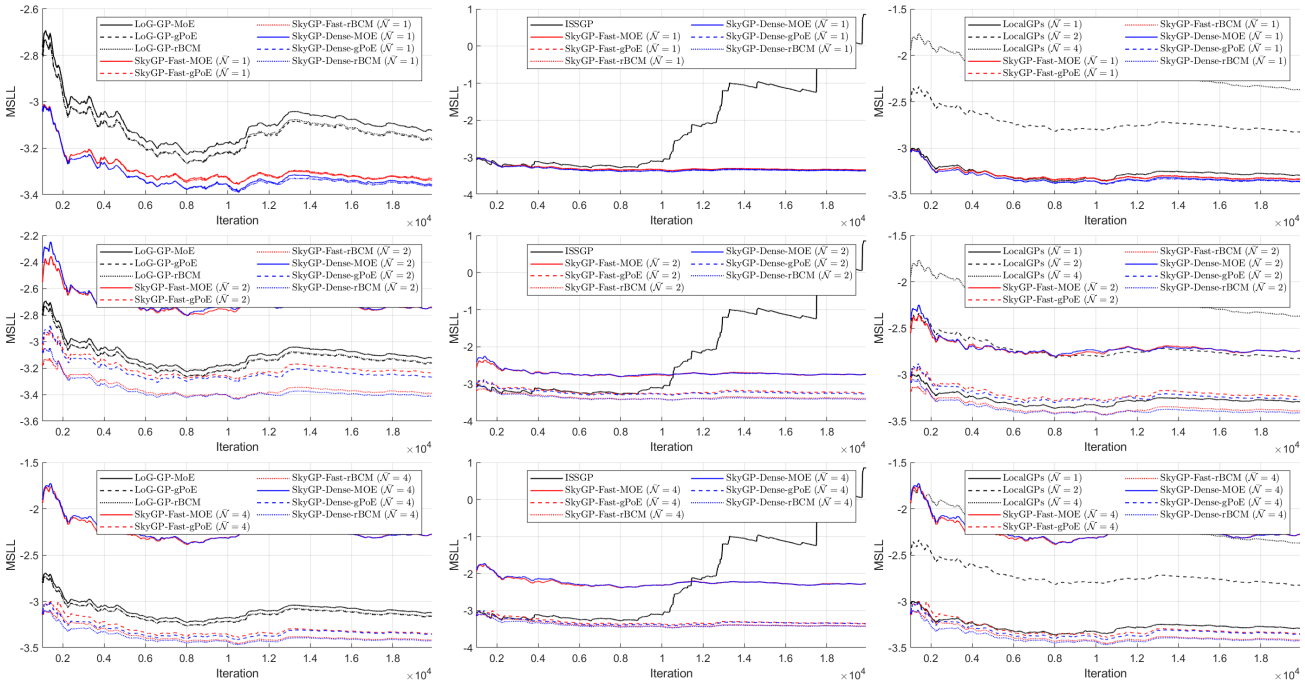


Figure 10: MSLL comparison on ELECTRIC dataset against all baselines.

Table 3: Average prediction and update time (ms) across datasets.

Model	**SARCOS**		**KIN40K**		**PUMA**		**ELECTRIC**	
	t_{pred}	t_{up}	t_{pred}	t_{up}	t_{pred}	t_{up}	t_{pred}	t_{up}
LoGGP-MOE	0.28	0.21	**0.24**	0.16	0.23	0.18	0.19	0.25
LoGGP-gPOE	0.29	0.20	**0.24**	0.16	**0.22**	0.19	0.22	0.24
LoGGP-rBCM	0.30	0.22	**0.24**	0.16	**0.22**	0.20	0.18	0.24
SkyGP-Fast-MoE ($\bar{\mathcal{N}} = 1$)	0.17	**0.04**	0.30	**0.04**	0.29	0.04	0.15	**0.04**
SkyGP-Fast-MoE ($\bar{\mathcal{N}} = 2$)	0.20	**0.04**	0.32	**0.04**	0.33	0.04	0.17	**0.04**
SkyGP-Fast-MoE ($\bar{\mathcal{N}} = 4$)	0.23	**0.04**	0.35	**0.04**	0.34	0.04	0.20	**0.04**
SkyGP-Dense-MoE ($\bar{\mathcal{N}} = 1$)	0.18	0.09	0.38	0.06	0.30	0.08	**0.14**	0.06
SkyGP-Dense-MoE ($\bar{\mathcal{N}} = 2$)	0.19	0.12	0.29	0.09	0.30	0.12	0.15	0.07
SkyGP-Dense-MoE ($\bar{\mathcal{N}} = 4$)	0.23	0.16	0.32	0.11	0.32	0.16	0.19	0.08
SkyGP-Fast-gPoE ($\bar{\mathcal{N}} = 1$)	**0.16**	**0.04**	0.30	**0.04**	0.29	**0.04**	0.15	**0.04**
SkyGP-Fast-gPoE ($\bar{\mathcal{N}} = 2$)	0.19	**0.04**	0.35	**0.04**	0.33	**0.04**	0.17	**0.04**
SkyGP-Fast-gPoE ($\bar{\mathcal{N}} = 4$)	0.23	**0.04**	0.36	**0.04**	0.35	**0.04**	0.20	**0.04**
SkyGP-Dense-gPoE ($\bar{\mathcal{N}} = 1$)	0.18	0.09	0.28	0.06	0.31	0.08	**0.14**	0.06
SkyGP-Dense-gPoE ($\bar{\mathcal{N}} = 2$)	0.21	0.12	0.29	0.09	0.29	0.12	0.17	0.07
SkyGP-Dense-gPoE ($\bar{\mathcal{N}} = 4$)	0.24	0.16	0.35	0.11	0.32	0.16	0.21	0.08
SkyGP-Fast-rBCM ($\bar{\mathcal{N}} = 1$)	**0.16**	**0.04**	0.30	**0.04**	0.26	**0.04**	0.15	**0.04**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}} = 2$)	0.18	**0.04**	0.33	**0.04**	0.28	**0.04**	0.19	**0.04**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}} = 4$)	0.23	**0.04**	0.35	**0.04**	0.31	**0.04**	0.21	**0.04**
SkyGP-Dense-rBCM ($\bar{\mathcal{N}} = 1$)	0.17	0.09	0.28	0.06	0.25	0.08	**0.14**	0.06
SkyGP-Dense-rBCM ($\bar{\mathcal{N}} = 2$)	0.18	0.12	0.30	0.09	0.25	0.12	0.16	0.07
SkyGP-Dense-rBCM ($\bar{\mathcal{N}} = 4$)	0.24	0.16	0.35	0.11	0.28	0.16	0.22	0.08
LocalGPs ($\bar{\mathcal{N}} = 1$)	1.17	0.15	2.32	0.05	1.07	0.08	2.78	0.06
LocalGPs ($\bar{\mathcal{N}} = 2$)	1.27	0.17	2.07	0.05	1.07	0.08	2.67	0.06
LocalGPs ($\bar{\mathcal{N}} = 4$)	1.25	0.23	2.03	0.05	1.14	0.08	2.88	0.06
ISSGP	18	7	3	8	3	6	4	8
SSGP	5	4	3	8	2	6	-	-

sparse GP approaches (SSGP and ISSGP). Compared to LoG-GP, SkyGP-Fast with gPoE and rBCM achieves better performance before 2500 iterations, and SkyGP-Dense outperforms it before 4500 iterations when using 2 experts. With $\bar{\mathcal{N}} = 4$, SkyGP-Fast consistently outperforms LoG-GP before 5000 iterations. When compared to LocalGPs, SkyGP with $\bar{\mathcal{N}} = 4$ demonstrates superior performance, and SkyGP-Dense with 2 experts also outperforms LocalGPs, while SkyGP-Fast with 2 experts only shows an advantage before 5000 iterations. On the ELECTRIC dataset, when $\bar{\mathcal{N}} = 1$, all variants of SkyGP outperform the baseline models, including LoG-GP, SSGP, ISSGP, and LocalGPs. When using 2 or 4 experts with gPoE or rBCM aggregation, SkyGP continues to outperform LoG-GP, ISSGP, and SSGP. However, when compared to LocalGPs, only SkyGP-Fast and SkyGP-Dense with rBCM achieve better performance when $\bar{\mathcal{N}} = 2$.

Across all datasets, SkyGP-Dense consistently achieves the best MSLL performance, particularly with $\bar{\mathcal{N}} = 4$, outperforming both SkyGP-Fast and all baseline models. SkyGP-Fast also shows better results with 2 or 4 experts, often surpassing LoG-GP, SSGP, and ISSGP. Notably, on ELECTRIC and PUMA, even single-expert SkyGP-Dense ($\bar{\mathcal{N}} = 1$) performs better than LoG-GP. And on KIN40K and ELECTRIC, all SkyGP variants outperform the sparse baselines.

B.3 Control Tasks

In the control task, we compare the baseline model with LoG-GP and LocalGP, as sparse GP methods are too slow to complete prediction and model updates within the simulation time step of 0.02 s. Moreover, both LoG-GP and SkyGP variants employ four experts for the aggregation method. As shown in Figure 11, LoG-GP shows the largest variance and tracking error throughout the simulation. SkyGP-Dense achieves the lowest tracking error after approximately 13 s and demonstrates smoother tracking behavior. Moreover, SkyGP-Fast delivers performance comparable to that of LocalGPs. In Figure 12, SkyGP-Fast shows prediction errors similar to those of LocalGPs. Similarly, SkyGP-Dense maintains a smooth error profile but displays increased variance between 2 s and 3 s. In contrast, LoG-GP performs the worst in terms of both mean and variance of the prediction error.

B.4 Ablation Experiments

In this subsection, we perform ablation studies to evaluate the contribution of various algorithmic components to the overall performance of the proposed framework. In particular, we examine the effects of employing Euclidean distance

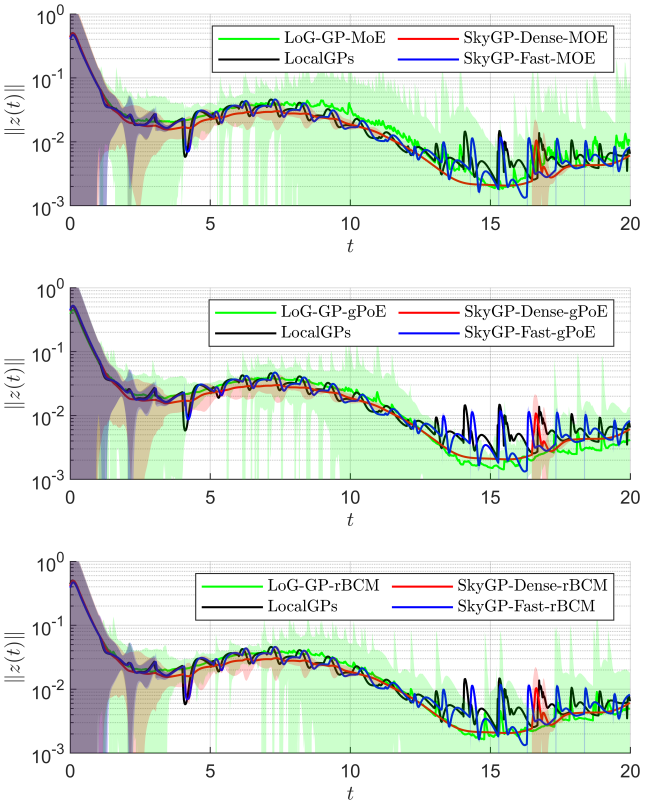


Figure 11: Tracking error comparison against LoG-GP and LocalGPs.

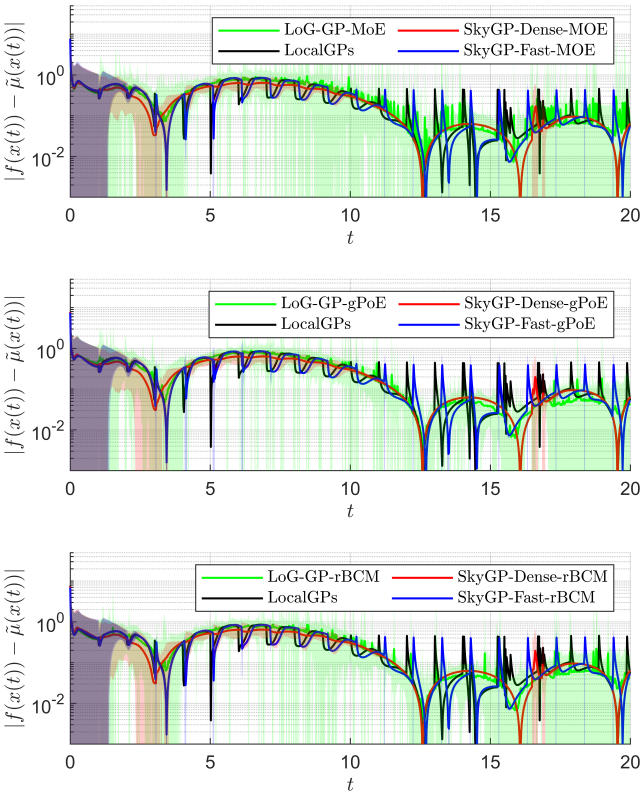


Figure 12: Prediction error comparison against LoG-GP and LocalGPs.

(ED) as the expert center denoted by

$$\boldsymbol{c}_i^k = (k-1)\boldsymbol{c}_i^{k-1}/k + \boldsymbol{x}^k \tag{46}$$

versus kernel distance (KD) in (4); the use of drop center (DC) ${}^{\text{off}}\boldsymbol{c}_{\text{nr}}$, the adaptive window mechanism (AW) in the line 2 of Algorithm 1, and the time-aware decay (TD) weighting factor ρ in Algorithm 2.

For ED versus KD, our experimental results consistently demonstrate that KD similarity achieves superior performance in both SMSE and MSLL metrics, highlighting its effectiveness in enhancing predictive accuracy and uncertainty quantification in online settings. To evaluate the effect of DC, we compare models that incorporate dedicated DCs with those that rely solely on expert center distances for data replacement decisions. On the SARCOS and PUMA datasets, using DC generally yields better performance than not using it, while on the ELECTRIC and KIN40K datasets, the difference is negligible. Overall, the results indicate that incorporating drop centers can enhance predictive accuracy without adding computational overhead. Regarding the AW strategy, we observe that disabling it yields slightly better accuracy across all datasets but increases inference latency by approximately 0.1 ms per sample. Adjusting the AW scaling factor thus enables a flexible trade-off between prediction quality and efficiency. For the TD mechanism, results on SARCOS show consistent improvement in both SMSE and MSLL, while PUMA sees gains in SMSE but only marginal changes in MSLL. For KIN40K and ELECTRIC, TD has minimal impact on either metric.

Table 4: Average SMSE on the SARCOS dataset. ✓indicates that the strategy is used; ✗indicates that it is not.

Strategy	KD		DC		AW		TD	
	✓	✗	✓	✗	✓	✗	✓	✗
SkyGP-Fast-MoE ($\bar{\mathcal{N}}=1$)	**0.039**	0.057	-	-	0.038	**0.033**	**0.038**	**0.038**
SkyGP-Fast-MoE ($\bar{\mathcal{N}}=2$)	**0.057**	0.077	-	-	**0.050**	**0.050**	**0.056**	0.057
SkyGP-Fast-MoE ($\bar{\mathcal{N}}=4$)	**0.078**	0.097	-	-	**0.078**	0.085	**0.078**	0.084
SkyGP-Dense-MoE ($\bar{\mathcal{N}}=1$)	**0.033**	0.045	**0.033**	**0.033**	**0.033**	**0.033**	**0.033**	**0.033**
SkyGP-Dense-MoE ($\bar{\mathcal{N}}=2$)	**0.052**	0.058	**0.053**	0.059	0.052	**0.050**	**0.052**	0.054
SkyGP-Dense-MoE ($\bar{\mathcal{N}}=4$)	**0.100**	0.100	0.100	**0.097**	0.100	**0.085**	0.099	**0.098**
SkyGP-Fast-gPoE($\bar{\mathcal{N}}=1$)	**0.039**	0.057	-	-	0.038	**0.033**	**0.038**	0.038
SkyGP-Fast-gPoE ($\bar{\mathcal{N}}=2$)	**0.026**	0.033	-	-	0.025	**0.020**	**0.025**	0.026
SkyGP-Fast-gPoE ($\bar{\mathcal{N}}=4$)	**0.022**	0.026	-	-	0.022	**0.017**	**0.022**	0.023
SkyGP-Dense-gPoE ($\bar{\mathcal{N}}=1$)	**0.033**	0.045	**0.030**	**0.030**	**0.033**	**0.033**	**0.033**	**0.033**
SkyGP-Dense-gPoE ($\bar{\mathcal{N}}=2$)	**0.019**	0.022	**0.019**	0.022	**0.019**	**0.019**	**0.020**	**0.020**
SkyGP-Dense-gPoE ($\bar{\mathcal{N}}=4$)	**0.017**	0.018	**0.017**	0.019	0.017	**0.016**	**0.017**	**0.017**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}}=1$)	**0.039**	0.057	-	-	0.038	**0.033**	**0.038**	**0.038**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}}=2$)	**0.023**	0.031	-	-	0.023	**0.019**	**0.023**	0.024
SkyGP-Fast-rBCM ($\bar{\mathcal{N}}=4$)	**0.024**	0.026	-	-	0.024	**0.018**	**0.024**	0.025
SkyGP-Dense-rBCM($\bar{\mathcal{N}}=1$)	**0.033**	0.045	**0.030**	**0.030**	**0.033**	**0.033**	**0.033**	**0.033**
SkyGP-Dense-rBCM ($\bar{\mathcal{N}}=2$)	**0.018**	0.021	**0.019**	0.022	**0.018**	**0.018**	**0.018**	0.019
SkyGP-Dense-rBCM ($\bar{\mathcal{N}}=4$)	**0.018**	**0.018**	**0.018**	0.019	**0.017**	**0.017**	**0.017**	0.018

Table 5: Average MSLL on the SARCOS dataset. ✓indicates that the strategy is used; ✗indicates that it is not.

Strategy	KD		DC		AW		TD	
	✓	✗	✓	✗	✓	✗	✓	✗
SkyGP-Fast-MoE ($\bar{\mathcal{N}}=1$)	**-1.82**	-1.74	-	-	-1.82	**-1.89**	**-1.82**	**-1.82**
SkyGP-Fast-MoE ($\bar{\mathcal{N}}=2$)	**-1.54**	-1.42	-	-	-1.54	**-1.66**	**-1.54**	**-1.54**
SkyGP-Fast-MoE ($\bar{\mathcal{N}}=4$)	**-1.32**	-1.22	-	-	-1.32	**-1.38**	**-1.32**	-1.31
SkyGP-Dense-MoE ($\bar{\mathcal{N}}=1$)	**-1.88**	-1.82	**-1.88**	**-1.88**	-1.88	**-1.89**	-1.88	**-1.89**
SkyGP-Dense-MoE ($\bar{\mathcal{N}}=2$)	**-1.60**	-1.57	**-1.60**	-1.56	-1.59	**-1.65**	**-1.59**	**-1.59**
SkyGP-Dense-MoE ($\bar{\mathcal{N}}=4$)	**-1.24**	-1.23	-1.24	**-1.26**	-1.24	**-1.38**	**-1.25**	**-1.25**
SkyGP-Fast-PoE($\bar{\mathcal{N}}=1$)	**-1.82**	-1.74	-	-	-1.82	**-1.89**	**-1.82**	**-1.82**
SkyGP-Fast-gPoE ($\bar{\mathcal{N}}=2$)	**-1.79**	-1.74	-	-	-1.79	**-1.88**	**-1.79**	**-1.79**
SkyGP-Fast-gPoE ($\bar{\mathcal{N}}=4$)	**-1.93**	-1.89	-	-	-1.93	**-2.04**	**-1.93**	-1.92
SkyGP-Dense-gPoE ($\bar{\mathcal{N}}=1$)	**-1.88**	-1.81	**-1.88**	**-1.88**	-1.88	**-1.89**	-1.88	**-1.89**
SkyGP-Dense-gPoE ($\bar{\mathcal{N}}=2$)	**-1.89**	-1.85	**-1.89**	-1.85	**-1.89**	-1.88	**-1.89**	**-1.89**
SkyGP-Dense-gPoE ($\bar{\mathcal{N}}=4$)	**-2.01**	-2.00	**-2.01**	-1.99	-2.01	**-2.03**	**-2.01**	**-2.01**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}}=1$)	**-1.82**	-1.74	-	-	-1.83	**-1.89**	**-1.83**	-1.82
SkyGP-Fast-rBCM ($\bar{\mathcal{N}}=2$)	**-1.93**	-1.87	-	-	-1.93	**-2.02**	**-1.93**	-1.92
SkyGP-Fast-rBCM ($\bar{\mathcal{N}}=4$)	**-1.91**	-1.89	-	-	-1.91	**-2.04**	**-1.91**	-1.90
SkyGP-Dense-rBCM($\bar{\mathcal{N}}=1$)	**-1.88**	-1.81	**-1.88**	**-1.88**	-1.88	**-1.89**	**-1.89**	**-1.89**
SkyGP-Dense-rBCM ($\bar{\mathcal{N}}=2$)	**-2.00**	-1.98	**-2.00**	-1.97	-1.89	**-2.01**	**-2.00**	-1.99
SkyGP-Dense-rBCM ($\bar{\mathcal{N}}=4$)	**-2.01**	-2.00	**-2.01**	-1.99	-2.01	**-2.03**	**-2.01**	**-2.01**

Table 6: Average SMSE on the PUMA dataset. ✓indicates that the strategy is used; ✗indicates that it is not.

Strategy	KD		DC		AW		TD	
	✓	✗	✓	✗	✓	✗	✓	✗
SkyGP-Fast-MoE ($\bar{\mathcal{N}} = 1$)	**0.26**	0.64	-	-	**0.26**	**0.26**	**0.26**	**0.26**
SkyGP-Fast-MoE ($\bar{\mathcal{N}} = 2$)	**0.18**	0.35	-	-	**0.18**	**0.18**	**0.18**	**0.18**
SkyGP-Fast-MoE ($\bar{\mathcal{N}} = 4$)	**0.16**	0.20	-	-	**0.16**	**0.16**	**0.16**	**0.16**
SkyGP-Dense-MoE ($\bar{\mathcal{N}} = 1$)	**0.23**	0.55	**0.23**	**0.23**	**0.23**	**0.23**	**0.23**	**0.23**
SkyGP-Dense-MoE ($\bar{\mathcal{N}} = 2$)	**0.15**	0.28	**0.15**	0.16	**0.15**	**0.15**	**0.15**	**0.15**
SkyGP-Dense-MoE ($\bar{\mathcal{N}} = 4$)	**0.14**	0.18	**0.14**	**0.14**	**0.14**	**0.14**	**0.14**	**0.14**
SkyGP-Fast-gPoE($\bar{\mathcal{N}} = 1$)	**0.26**	0.64	-	-	**0.26**	**0.26**	**0.26**	**0.26**
SkyGP-Fast-gPoE ($\bar{\mathcal{N}} = 2$)	**0.13**	0.22	-	-	**0.13**	0.14	**0.13**	**0.13**
SkyGP-Fast-gPoE ($\bar{\mathcal{N}} = 4$)	**0.09**	0.10	-	-	**0.09**	**0.09**	**0.09**	**0.09**
SkyGP-Dense-gPoE ($\bar{\mathcal{N}} = 1$)	**0.22**	0.55	0.23	**0.22**	**0.23**	**0.23**	**0.22**	0.023
SkyGP-Dense-gPoE ($\bar{\mathcal{N}} = 2$)	**0.11**	0.17	**0.11**	0.12	**0.11**	**0.11**	**0.11**	**0.11**
SkyGP-Dense-PoE ($\bar{\mathcal{N}} = 4$)	**0.08**	0.10	**0.08**	0.09	0.08	**0.08**	**0.08**	**0.09**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}} = 1$)	**0.26**	0.64	-	-	**0.26**	**0.26**	**0.26**	**0.26**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}} = 2$)	**0.12**	0.20	-	-	**0.12**	**0.12**	**0.12**	**0.12**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}} = 4$)	**0.09**	0.11	-	-	**0.09**	**0.09**	**0.09**	**0.09**
SkyGP-Dense-rBCM($\bar{\mathcal{N}} = 1$)	**0.22**	0.55	0.23	**0.22**	**0.23**	**0.23**	**0.23**	**0.23**
SkyGP-Dense-rBCM ($\bar{\mathcal{N}} = 2$)	**0.10**	0.16	**0.11**	**0.11**	0.11	**0.10**	**0.10**	0.11
SkyGP-Dense-rBCM ($\bar{\mathcal{N}} = 4$)	**0.08**	0.10	**0.09**	**0.09**	**0.09**	**0.09**	**0.08**	0.09

Table 7: Average MSLL on the PUMA dataset. ✓indicates that the strategy is used; ✗indicates that it is not.

Strategy	KD		DC		AW		TD	
	✓	✗	✓	✗	✓	✗	✓	✗
SkyGP-Fast-MoE ($\bar{\mathcal{N}} = 1$)	**-0.92**	-0.64	-	-	**-0.92**	**-0.92**	**-0.92**	**-0.92**
SkyGP-Fast-MoE ($\bar{\mathcal{N}} = 2$)	**-1.00**	-0.70	-	-	**-1.00**	**-1.00**	**-1.00**	**-1.00**
SkyGP-Fast-MoE ($\bar{\mathcal{N}} = 4$)	**-1.03**	-0.82	-	-	**-1.03**	**-1.03**	**-1.03**	**-1.03**
SkyGP-Dense-MoE ($\bar{\mathcal{N}} = 1$)	**-1.02**	-0.69	**-1.02**	**-1.02**	**-1.02**	**-1.02**	**-1.02**	**-1.02**
SkyGP-Dense-MoE ($\bar{\mathcal{N}} = 2$)	**-1.10**	-0.87	**-1.10**	**-1.10**	**-1.10**	**-1.10**	**-1.10**	**-1.10**
SkyGP-Dense-MoE ($\bar{\mathcal{N}} = 4$)	**-1.09**	-0.93	**-1.09**	-1.07	**-1.09**	**-1.09**	**-1.09**	**-1.09**
SkyGP-Fast-gPoE($\bar{\mathcal{N}} = 1$)	**-0.92**	-0.64	-	-	**-0.92**	**-0.92**	**-0.92**	**-0.92**
SkyGP-Fast-gPoE ($\bar{\mathcal{N}} = 2$)	**-1.15**	-0.98	-	-	**-1.15**	**-1.15**	**-1.15**	**-1.15**
SkyGP-Fast-gPoE ($\bar{\mathcal{N}} = 4$)	**-1.38**	-1.28	-	-	**-1.37**	**-1.37**	**-1.38**	**-1.38**
SkyGP-Dense-gPoE ($\bar{\mathcal{N}} = 1$)	**-1.02**	-0.69	**-1.02**	**-1.02**	**-1.02**	-1.01	**-1.02**	**-1.02**
SkyGP-Dense-gPoE ($\bar{\mathcal{N}} = 2$)	**-1.22**	-1.04	**-1.22**	**-1.22**	**-1.22**	**-1.22**	**-1.22**	**-1.22**
SkyGP-Dense-gPoE ($\bar{\mathcal{N}} = 4$)	**-1.39**	-1.31	**-1.39**	-1.38	**-1.39**	**-1.39**	**-1.39**	**-1.39**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}} = 1$)	**-0.92**	-0.64	-	-	**-0.92**	**-0.92**	**-0.92**	**-0.92**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}} = 2$)	**-1.26**	-1.09	-	-	**-1.26**	**-1.26**	**-1.26**	**-1.26**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}} = 4$)	**-1.39**	-1.26	-	-	**-1.39**	**-1.39**	**-1.39**	**-1.39**
SkyGP-Dense-rBCM($\bar{\mathcal{N}} = 1$)	**-1.01**	-0.69	**-1.02**	**-1.02**	**-1.01**	**-1.01**	**-1.01**	**-1.01**
SkyGP-Dense-rBCM ($\bar{\mathcal{N}} = 2$)	**-1.31**	-1.16	-1.31	**1.32**	**1.32**	**-1.32**	**-1.31**	**-1.31**
SkyGP-Dense-rBCM ($\bar{\mathcal{N}} = 4$)	**-1.40**	-1.30	**-1.40**	-1.39	**-1.40**	**-1.40**	**-1.40**	**-1.40**

Table 8: Average SMSE on the KIN40K dataset. ✓indicates that the strategy is used; ✗indicates that it is not.

Strategy	KD		DC		AW		TD	
	✓	✗	✓	✗	✓	✗	✓	✗
SkyGP-Fast-MoE ($\bar{\mathcal{N}}=1$)	**0.62**	0.66	-	-	0.67	**0.62**	**0.61**	0.62
SkyGP-Fast-MoE ($\bar{\mathcal{N}}=2$)	**0.46**	0.51	-	-	0.52	**0.46**	**0.46**	**0.46**
SkyGP-Fast-MoE ($\bar{\mathcal{N}}=4$)	**0.46**	0.47	-	-	0.51	**0.46**	**0.46**	**0.46**
SkyGP-Dense-MoE ($\bar{\mathcal{N}}=1$)	**0.54**	0.55	**0.54**	0.56	0.59	**0.54**	**0.54**	**0.54**
SkyGP-Dense-MoE ($\bar{\mathcal{N}}=2$)	**0.44**	**0.44**	**0.44**	**0.44**	0.49	**0.44**	**0.44**	**0.44**
SkyGP-Dense-MoE ($\bar{\mathcal{N}}=4$)	**0.48**	0.50	0.48	**0.47**	0.51	**0.48**	**0.48**	**0.48**
SkyGP-Fast-gPoE($\bar{\mathcal{N}}=1$)	**0.62**	0.66	-	-	0.67	**0.62**	**0.62**	**0.62**
SkyGP-Fast-gPoE ($\bar{\mathcal{N}}=2$)	**0.46**	0.50	-	-	0.54	**0.46**	**0.46**	**0.46**
SkyGP-Fast-gPoE ($\bar{\mathcal{N}}=4$)	**0.36**	0.37	-	-	0.41	**0.36**	**0.36**	**0.36**
SkyGP-Dense-gPoE ($\bar{\mathcal{N}}=1$)	**0.54**	0.55	**0.54**	0.56	0.59	**0.54**	**0.54**	**0.54**
SkyGP-Dense-gPoE ($\bar{\mathcal{N}}=2$)	0.43	**0.42**	**0.43**	**0.43**	0.48	**0.43**	**0.43**	**0.43**
SkyGP-Dense-gPoE ($\bar{\mathcal{N}}=4$)	**0.38**	0.39	0.38	**0.37**	0.41	**0.38**	**0.38**	**0.38**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}}=1$)	**0.62**	0.66	-	-	0.67	**0.62**	**0.62**	**0.62**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}}=2$)	**0.42**	0.46	-	-	0.49	**0.42**	**0.42**	**0.42**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}}=4$)	**0.35**	0.37	-	-	0.40	**0.35**	**0.35**	**0.35**
SkyGP-Dense-rBCM($\bar{\mathcal{N}}=1$)	**0.54**	0.55	**0.54**	0.56	0.59	**0.54**	**0.54**	**0.54**
SkyGP-Dense-rBCM ($\bar{\mathcal{N}}=2$)	**0.39**	**0.39**	**0.39**	**0.39**	0.44	**0.39**	**0.39**	**0.39**
SkyGP-Dense-rBCM ($\bar{\mathcal{N}}=4$)	**0.38**	0.39	0.38	**0.37**	0.40	**0.38**	**0.38**	**0.38**

Table 9: Average MSLL on the KIN40K dataset. ✓indicates that the strategy is used; ✗indicates that it is not.

Strategy	KD		DC		AW		TD	
	✓	✗	✓	✗	✓	✗	✓	✗
SkyGP-Fast-MoE ($\bar{\mathcal{N}}=1$)	**-0.31**	-0.28	-	-	-0.26	**-0.31**	**-0.31**	**-0.31**
SkyGP-Fast-MoE ($\bar{\mathcal{N}}=2$)	**-0.43**	-0.39	-	-	-0.36	**-0.43**	**-0.43**	**-0.43**
SkyGP-Fast-MoE ($\bar{\mathcal{N}}=4$)	**-0.42**	-0.40	-	-	-0.36	**-0.42**	**-0.42**	**-0.42**
SkyGP-Dense-MoE ($\bar{\mathcal{N}}=1$)	**-0.40**	**-0.40**	**-0.40**	-0.38	-0.34	**-0.40**	**-0.40**	**-0.40**
SkyGP-Dense-MoE ($\bar{\mathcal{N}}=2$)	**-0.47**	-0.46	**-0.47**	**-0.47**	-0.40	**-0.47**	**-0.47**	**-0.47**
SkyGP-Dense-MoE ($\bar{\mathcal{N}}=4$)	**-0.39**	-0.36	-0.39	**-0.42**	-0.35	**-0.39**	**-0.39**	**-0.39**
SkyGP-Fast-gPoE($\bar{\mathcal{N}}=1$)	**-0.31**	-0.28	-	-	-0.26	**-0.31**	**-0.31**	**-0.31**
SkyGP-Fast-gPoE ($\bar{\mathcal{N}}=2$)	**-0.20**	-0.17	-	-	-0.09	**-0.20**	**-0.20**	**-0.20**
SkyGP-Fast-gPoE ($\bar{\mathcal{N}}=4$)	**-0.57**	-0.55	-	-	-0.49	**-0.57**	**-0.58**	**-0.58**
SkyGP-Dense-gPoE ($\bar{\mathcal{N}}=1$)	**-0.40**	**-0.40**	**-0.40**	-0.38	-0.34	**-0.40**	**-0.40**	**-0.40**
SkyGP-Dense-gPoE ($\bar{\mathcal{N}}=2$)	**-0.28**	-0.25	-0.28	**-0.29**	-0.20	**-0.28**	**-0.28**	**-0.28**
SkyGP-Dense-gPoE ($\bar{\mathcal{N}}=4$)	**-0.57**	-0.55	-0.57	**-0.61**	-0.52	**-0.57**	**-0.57**	**-0.57**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}}=1$)	**-0.31**	-0.28	-	-	-0.26	**-0.31**	**-0.31**	**-0.31**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}}=2$)	**-0.52**	-0.49	-	-	-0.44	**-0.52**	**-0.53**	**-0.53**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}}=4$)	**-0.59**	-0.58	-	-	-0.35	**-0.50**	**-0.59**	**-0.59**
SkyGP-Dense-rBCM($\bar{\mathcal{N}}=1$)	**-0.40**	**-0.40**	**-0.40**	-0.38	-0.34	**-0.40**	**-0.40**	**-0.40**
SkyGP-Dense-rBCM ($\bar{\mathcal{N}}=2$)	**-0.59**	**-0.59**	-0.59	**-0.60**	-0.51	**-0.59**	**-0.59**	**-0.59**
SkyGP-Dense-rBCM ($\bar{\mathcal{N}}=4$)	**-0.61**	-0.60	-0.61	**-0.62**	-0.56	**-0.61**	**-0.61**	**-0.61**

Table 10: Average SMSE on the ELECTRIC dataset. ✓indicates that the strategy is used; ✗indicates that it is not.

Strategy	KD		DC		AW		TD	
	✓	✗	✓	✗	✓	✗	✓	✗
SkyGP-Fast-MoE ($\bar{\mathcal{N}} = 1$)	**0.08**	0.10	-	-	**0.08**	**0.08**	**0.08**	**0.08**
SkyGP-Fast-MoE ($\bar{\mathcal{N}} = 2$)	**0.13**	0.15	-	-	**0.13**	**0.13**	**0.13**	**0.13**
SkyGP-Fast-MoE ($\bar{\mathcal{N}} = 4$)	**0.17**	0.22	-	-	**0.17**	**0.17**	**0.17**	**0.17**
SkyGP-Dense-MoE ($\bar{\mathcal{N}} = 1$)	**0.08**	0.10	**0.08**	**0.08**	**0.08**	**0.08**	**0.08**	**0.08**
SkyGP-Dense-MoE ($\bar{\mathcal{N}} = 2$)	**0.12**	0.15	**0.12**	0.13	**0.12**	**0.12**	**0.12**	**0.12**
SkyGP-Dense-MoE ($\bar{\mathcal{N}} = 4$)	**0.18**	0.22	**0.18**	**0.18**	**0.18**	**0.18**	**0.18**	**0.18**
SkyGP-Fast-gPoE($\bar{\mathcal{N}} = 1$)	**0.08**	0.11	-	-	**0.08**	**0.08**	**0.08**	**0.08**
SkyGP-Fast-gPoE ($\bar{\mathcal{N}} = 2$)	**0.09**	**0.09**	-	-	**0.09**	**0.09**	**0.09**	**0.09**
SkyGP-Fast-gPoE ($\bar{\mathcal{N}} = 4$)	**0.08**	0.09	-	-	**0.08**	**0.08**	**0.08**	**0.08**
SkyGP-Dense-gPoE ($\bar{\mathcal{N}} = 2$)	**0.07**	0.09	**0.07**	**0.07**	**0.07**	**0.07**	**0.07**	**0.07**
SkyGP-Dense-gPoE ($\bar{\mathcal{N}} = 4$)	**0.07**	0.08	**0.07**	0.08	**0.07**	**0.07**	**0.07**	**0.07**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}} = 1$)	**0.08**	0.11	-	-	**0.08**	0.09	**0.08**	**0.08**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}} = 2$)	**0.09**	**0.09**	-	-	0.09	**0.08**	**0.09**	**0.09**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}} = 4$)	**0.07**	0.09	-	-	**0.07**	**0.07**	**0.07**	**0.07**
SkyGP-Dense-rBCM($\bar{\mathcal{N}} = 1$)	**0.08**	0.10	**0.08**	**0.08**	**0.08**	0.09	**0.08**	**0.08**
SkyGP-Dense-rBCM ($\bar{\mathcal{N}} = 2$)	**0.07**	0.09	**0.07**	**0.07**	**0.07**	0.08	**0.07**	**0.07**
SkyGP-Dense-rBCM ($\bar{\mathcal{N}} = 4$)	**0.07**	0.08	**0.07**	0.08	**0.07**	**0.07**	**0.07**	**0.07**

Table 11: Average MSLL on the ELECTRIC dataset. ✓indicates that the strategy is used; ✗indicates that it is not.

Strategy	KD		DC		AW		TD	
	✓	✗	✓	✗	✓	✗	✓	✗
SkyGP-Fast-MoE ($\bar{\mathcal{N}} = 1$)	**-3.30**	-3.12	-	-	-3.29	**-3.30**	**-3.30**	**-3.30**
SkyGP-Fast-MoE ($\bar{\mathcal{N}} = 2$)	**-2.73**	-2.45	-	-	**-2.73**	**-2.73**	**-2.73**	**-2.73**
SkyGP-Fast-MoE ($\bar{\mathcal{N}} = 4$)	**-2.40**	-2.09	-	-	**-2.40**	**-2.40**	**-2.40**	**-2.40**
SkyGP-Dense-MoE ($\bar{\mathcal{N}} = 1$)	**-3.32**	-3.18	**-3.32**	-3.31	**-3.32**	**-3.32**	**-3.32**	**-3.32**
SkyGP-Dense-MoE ($\bar{\mathcal{N}} = 2$)	**-2.78**	-2.50	-2.78	**-2.79**	-2.78	**-2.79**	**-2.78**	**-2.78**
SkyGP-Dense-MoE ($\bar{\mathcal{N}} = 4$)	**-2.38**	-2.06	-2.38	**-2.41**	**-2.38**	**-2.38**	**-2.38**	**-2.38**
SkyGP-Fast-gPoE($\bar{\mathcal{N}} = 1$)	**-3.30**	-3.12	-	-	**-3.30**	-3.28	**-3.30**	**-3.30**
SkyGP-Fast-gPoE ($\bar{\mathcal{N}} = 2$)	**-3.16**	-2.99	-	-	**-3.17**	**-3.17**	**-3.16**	**-3.16**
SkyGP-Fast-gPoE ($\bar{\mathcal{N}} = 4$)	**-3.29**	-3.18	-	-	-3.29	**-3.32**	**-3.29**	**-3.29**
SkyGP-Dense-gPoE ($\bar{\mathcal{N}} = 1$)	**-3.32**	-3.18	**-3.32**	-3.31	**-3.31**	**-3.31**	**-3.32**	**-3.32**
SkyGP-Dense-gPoE ($\bar{\mathcal{N}} = 2$)	**-3.24**	-3.14	**-3.24**	-3.20	**-3.23**	**-3.23**	**-3.24**	**-3.24**
SkyGP-Dense-gPoE ($\bar{\mathcal{N}} = 4$)	**-3.30**	-3.26	-3.30	**-3.31**	-3.30	**-3.33**	**-3.30**	**-3.30**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}} = 1$)	**-3.30**	-3.11	-	-	**-3.30**	-3.28	**-3.30**	**-3.30**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}} = 2$)	**-3.30**	-3.20	-	-	-3.30	**-3.34**	**-3.30**	**-3.30**
SkyGP-Fast-rBCM ($\bar{\mathcal{N}} = 4$)	**-3.35**	-3.28	-	-	-3.35	**-3.39**	**-3.35**	**-3.35**
SkyGP-Dense-rBCM($\bar{\mathcal{N}} = 1$)	**-3.32**	-3.18	**-3.32**	**-3.32**	**-3.32**	-3.31	**-3.32**	**-3.32**
SkyGP-Dense-rBCM ($\bar{\mathcal{N}} = 2$)	**-3.36**	-3.30	**-3.36**	**-3.36**	**-3.36**	**-3.36**	**-3.36**	**-3.36**
SkyGP-Dense-rBCM ($\bar{\mathcal{N}} = 4$)	**-3.36**	-3.34	-3.36	**-3.37**	-3.36	**-3.39**	**-3.36**	**-3.36**