GIT-BO: High-Dimensional Bayesian Optimization with Tabular Foundation Models

Rosen Ting-Ying Yu [1 2] **Cyril Picard** [1] **Faez Ahmed** [1 2]

Abstract

Existing high-dimensional Bayesian optimization (BO) methods typically leverage low-dimensional embeddings or structural assumptions to mitigate the curse of dimensionality, yet these approaches frequently incur considerable computational overhead due to iterative surrogate retraining and fixed assumptions. To address these limitations, we propose Gradient-Informed Bayesian Optimization using Tabular Foundation Models (GIT-BO), an approach that uses a pre-trained tabular foundation model (TFM) as a surrogate, leveraging its gradient information to adaptively identify low-dimensional subspaces for optimization. We propose a way to exploit internal gradient computations from the TFM's forward pass by creating a gradient-informed diagnostic matrix that reveals the most sensitive directions of the TFM's predictions, enabling optimization in a continuously re-estimated active subspace without the need for repeated model retraining. Extensive empirical evaluation across 23 synthetic and real-world benchmarks demonstrates that GIT-BO consistently outperforms four state-of-the-art Gaussian process-based high-dimensional BO methods, showing superior scalability and optimization performances especially as dimensionality increases up to 500 dimensions.

1. Introduction & Background

Bayesian optimization (BO) is widely recognized as an effective and sample-efficient technique for black-box optimization, essential in machine learning (Dewancker et al., 2016; Snoek et al., 2012), engineering design (Kumar et al., 2024; Wang & Dowling, 2022; Zhang et al., 2020; Yu et al., 2025), and hyperparameter tuning (Klein et al., 2017; Wu et al., 2019). However, BO can struggle in high-dimensional spaces (especially when dimension $D > 100$) due to the curse of dimensionality, making it difficult for conventional Gaussian process (GP)-based methods to effectively explore complex objective functions (Malu et al., 2021; Wang et al., 2023; Hvarfner et al., 2024; Xu et al., 2025; Letham et al., 2020). Existing high-dimensional BO approaches mitigate these challenges by exploiting low-dimensional subspaces, but still rely heavily on iterative model retraining and structural assumptions, incurring substantial computational overhead (Rana et al., 2017; Nayebi et al., 2019; Eriksson et al., 2019; Letham et al., 2020; Eriksson & Jankowiak, 2021).

Recent advances in tabular foundation models (TFM) have proposed using Prior-Data Fitted Networks (PFNs) to bypass surrogate refitting in Bayesian Optimization. Initially demonstrated for simple low-dimensional BO problems (Müller et al., 2023), the PFN-based BO was further validated for constrained engineering problems (Yu et al., 2025), using the parallel processing capabilities of the transformer architecture to outperform other methods in both speed and performance. TabPFN v2, a recent TFM model, extends the capabilities of TabPFN to handle inputs up to 500 dimensions, demonstrating superior performance in zero-shot classification, regression, and time series prediction tasks (Hollmann et al., 2025).

While TabPFN v2 presents an exciting opportunity as a high-dimensional BO surrogate due to its computational efficiency and predictive accuracy, its fixed-parameter foundation model architecture restricts its ability to dynamically adapt kernel hyperparameters. This is in contrast to traditional GP models, which adjust their kernel parameters during optimization to effectively identify principal searching directions in high-dimensional spaces (Eriksson & Jankowiak, 2021; Hvarfner et al., 2024; Xu et al., 2025). Addressing this limitation requires an approach that can leverage the strengths of foundation models while introducing adaptive mechanisms for subspace exploration, a gap we aim to bridge in this work.

Therefore, we introduce Gradient-Informed Bayesian Optimization using Tabular Foundation Models (GIT-BO)

[1]Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA [2]Center of Computational Science and Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA. Correspondence to: Rosen Ting-Ying Yu <rosenyu@mit.edu>.

Proceedings of the 1st ICML Workshop on Foundation Models for Structured Data, Vancouver, Canada. 2025.

algorithm— a novel framework leveraging pre-trained tabular models for scalable, high-dimensional optimization. GITBO leverages the gradient-informed subspace identification technique that adaptively learns promising search directions in high-dimensional spaces, exploiting the TFM's adaptive gradient knowledge during the optimization. Furthermore, we benchmark GIT-BO against four popular state-of-the-art (SOTA) GP high-dimensional BO methods across twenty-three diverse benchmarks, including synthetic functions and real-world problems, introducing foundation model surrogates as a viable alternative for complex Bayesian optimization tasks.

2. GIT-BO Overview

We summarize the GIT-BO algorithm in this section and provide the full details of the algorithm in the appendix. The framework consists of four main components: (1) Initial observed samples are collected in the high-dimensional space $\mathbb{R}^D$. (2) TabPFN v2, a fixed-weight tabular foundation model generates predictions of the objective space at inference time using in-context learning. (3) Gradient information from TabPFN's forward pass ($\nabla\hat{\mu}(x)$) is used to identify a low-dimensional gradient-informed (GI) subspace. We leverage the approximated diagnostic (Fisher information) matrix $H = \mathbb{E}_\mu[\nabla_x\mu_n(x)\,\nabla_x\mu_n(x)^\top]$ to identify the reduced subspace spanned by its principal eigenvectors (Zahm et al., 2022; Li et al., 2024a;b; Ly et al., 2017). The TabPFN v2 one-shot predictions $\hat{\mu}(x)$ and $\hat{\sigma}^2(x)$ are used for Thompson sampling acquisition calculations. (4) The next sample point (x_{next}) is selected within this GI subspace with the highest acquisition value and appended to the data set for iterative search until the stopping criteria are met. Figure 1 visually shows the GIT-BO methods, and Algorithm 1 demonstrates the algorithm.

3. Experiment

We benchmark GIT-BO against random search (Bergstra & Bengio, 2012) and five high-dimensional BO methods, including SAASBO (Eriksson & Jankowiak, 2021), TURBO (Eriksson et al., 2019), the Vanilla GP for high-dimensional BO that recently integrated in Botorch (Hvarfner et al., 2024), HESBO (Nayebi et al., 2019), and ALEBO (Letham et al., 2020). For HESBO and ALEBO, we varied $d_{\text{embedding}} \in \{10, 20\}$ and denote the choice as HESBO/ALEBO (d=$\{10, 20\}$). We evaluate the algorithms on a total of 63 benchmark problems, both synthetic and real-world tasks, at a fixed-iteration cutoff. Each algorithm run is repeated 20 times using fixed seeds, with statistical ranking and analysis calculated, and the average algorithm runtime t_{avg} is measured. Details of the experiment setup, implementation, and metric are in the appendix.

Algorithm 1 Gradient-Informed Bayesian Optimization using Tabular Foundation Models (GIT-BO)

Require: objective f, domain $\mathcal{X} \subset \mathbb{R}^D$, init. size n_0, iteration budget I, subspace dim. r

Draw n_0 LHS points x_i
Set $y_i = f(x_i)$; $D \leftarrow \{(x_i, y_i)\}_{i=1}^{n_0}$
for $i = 1$ to I **do**
  Fit TabPFN on D to obtain mean μ_n and variance σ_n^2
  Calculate gradient $\nabla_x\mu_n(x_i)$ for all $(x_i, y_i) \in D$
  $H = \mathbb{E}_\mu[\nabla_x\mu_n(x)\,\nabla_x\mu_n(x)^\top]$ diagnostic matrix;
  $V_r \leftarrow$ top-r eigvectors of H
  Set $x_{\text{cand}} \leftarrow x_{\text{ref}} + V_r z$, with $x_{\text{ref}} \leftarrow \bar{x}$ and $z \sim \mathrm{U}([-1, 1]^r)$
  $x_{\text{next}} \leftarrow \arg\max_j \mathrm{ThompsonSampling}_n(x_{\text{cand}})$
  Evaluate $y_{\text{next}} = f(x_{\text{next}})$and append the query point data $D \leftarrow D \cup \{(x_{\text{next}}, y_{\text{next}})\}$
end for
return $x^\star = \arg\min_{(x,y)\in D} y$

4. Results and Evaluations

This section reports a subset of the results of the 63 benchmark optimization experiments. The complete optimization results for all synthetic and real-world engineering problems are detailed in the Appendix.

4.1. Overall Statistical Ranking and Algorithm Runtime Tradeoffs

Across all problem variants, Figure 2 (a) shows that GIT-BO achieves the top statistical performance rank based on the optimization results, followed by SAASBO, ALEBO (d=10), and Vanilla GPEI. Even when we enforce a strict iteration limit at 100 iterations, GIT-BO still outperforms other GP-based algorithms in the statistical rank demonstrated in Figure 2 (b). A detailed rank-iteration evolution curve in Figure 2 (c) shows that GIT-BO becomes dominant within the first 25 iterations and keeps that lead. Figure 2 (d) plots the average run time t_{avg} of each algorithm versus the statistical rank. GIT-BO requires $\approx 10^3$ sec per trial, two orders of magnitude faster than SAASBO and ALEBO. With the first place in rank and third place in t_{avg}, GIT-BO is the Pareto frontier of speed and quality.

4.2. Scalable Synthetic Benchmarks (100 $\leq$ D $\leq$ 500)

The convergence results for the 45 scalable synthetic problems can be summarized in three categories (nine experiments) in Figure 3 (a). First, GIT-BO has absolute domination over several problems (e.g., Ackley), outperforming all other algorithms in all dimensions. The second category, the most common observed ones, are that ALEBO (d=10) or SAASBO match or outperform GIT-BO in $D = 100$; however, GIT-BO maintains its top optimization efficiency in

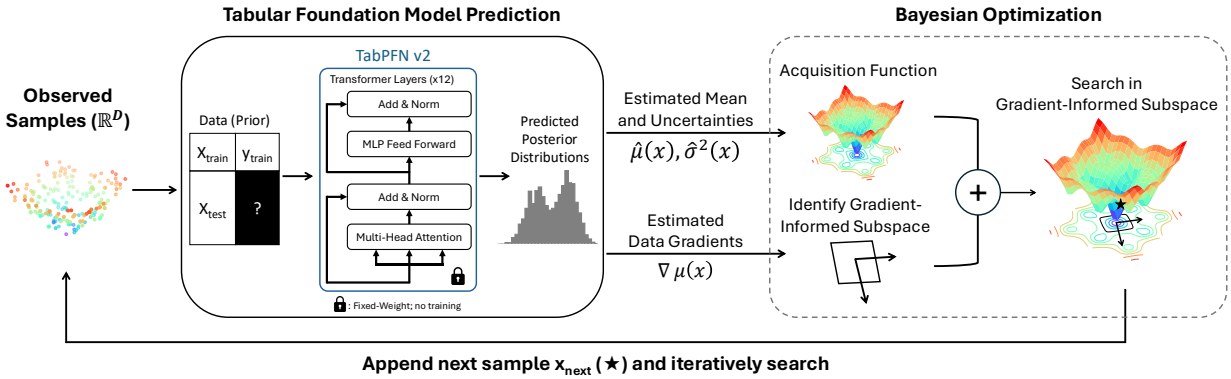

Figure 1. GIT-BO algorithm overview.

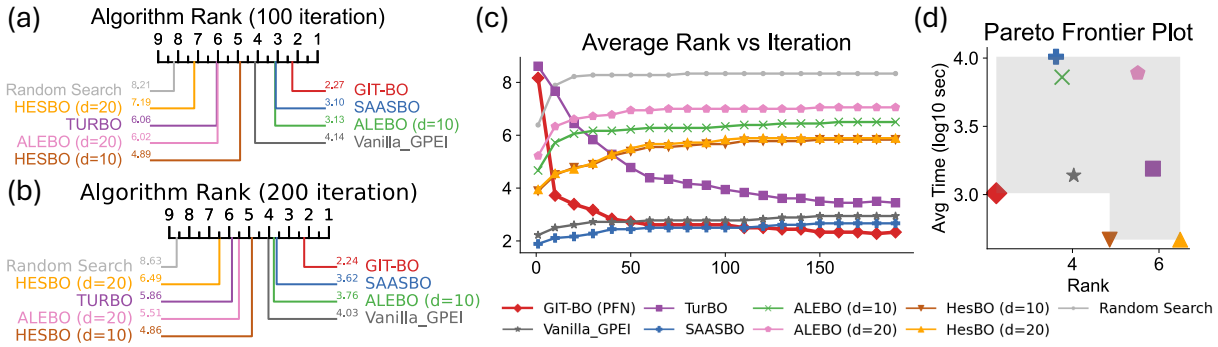

Figure 2. (a) and (b): Statistical ranking of the overall performance across 63 benchmark problems. GIT-BO ranked the top at optimization results at both fixed 100- and 200-iteration evaluations. (c): Plot of average algorithm rank at each iteration. Due to the compute limit, we only run ALEBO and SAASBO for 100 iterations. We compare their final results after 100 iterations with other algorithms run for 200 iterations. The plot shows that GIT-BO achieves a fast convergence at ranking performance as it remains the top of all algorithms after 25 iterations. (d): Plot of average time vs overall statistical rank. The shorter time and smaller rank perform better (bottom left corner), so we show GIT-BO at the Paerto front as the best algorithm.

the higher dimensions $D \geq 300$ (e.g., Levy). Finally, GIT-BO struggles with very few problems (e.g., Michalewicz), where the current SOTA methods with the trust region and linear embedding finding strategies dominate.

4.3. Real-world Problem

Figure 3 (b) illustrates the convergence results for a subset of six real-world problems. GIT-BO outperforms all baselines on the four 100+D CEC engineering tasks on power system optimization while achieves the second and third best in the LassoDNA problem and car problems (MOPTA08 Car and Mazda), respectively. One possible explanation is that these engineering benchmarks data might differ greatly from the training distribution of our TFM, TabPFN v2. Although SAASBO and Vanilla GPEI achieves the best for these problems, it takes on average two to three hours to solve them, while the second best, GIT-BO, takes only fifteen minutes. This showcases that GIT-BO offers a compelling trade-off for resource-constrained engineering optimization tasks where time-to-solution is critical.

5. Discussion

5.1. Novelties and Strengths

Our results demonstrate that a TFM can serve as a top-class surrogate in high-dimensional BO. By combining TabPFN's one-shot regression with a data-driven, gradient-informed subspace updating after every evaluation, GIT-BO converges faster than carefully tuned GP baselines on sixty-three benchmarks, maintaining its lead even as dimensionality grows to 500D. The approach removes the computational bottleneck of GP retraining and latent space searching while preserving the exploration–exploitation balance (up to two orders of magnitude speed up). Taken together, these properties position GIT-BO as an accessible "drop-in" optimizer for engineers who already rely on large language models and are now seeking equally powerful tools for structured data and design optimization.

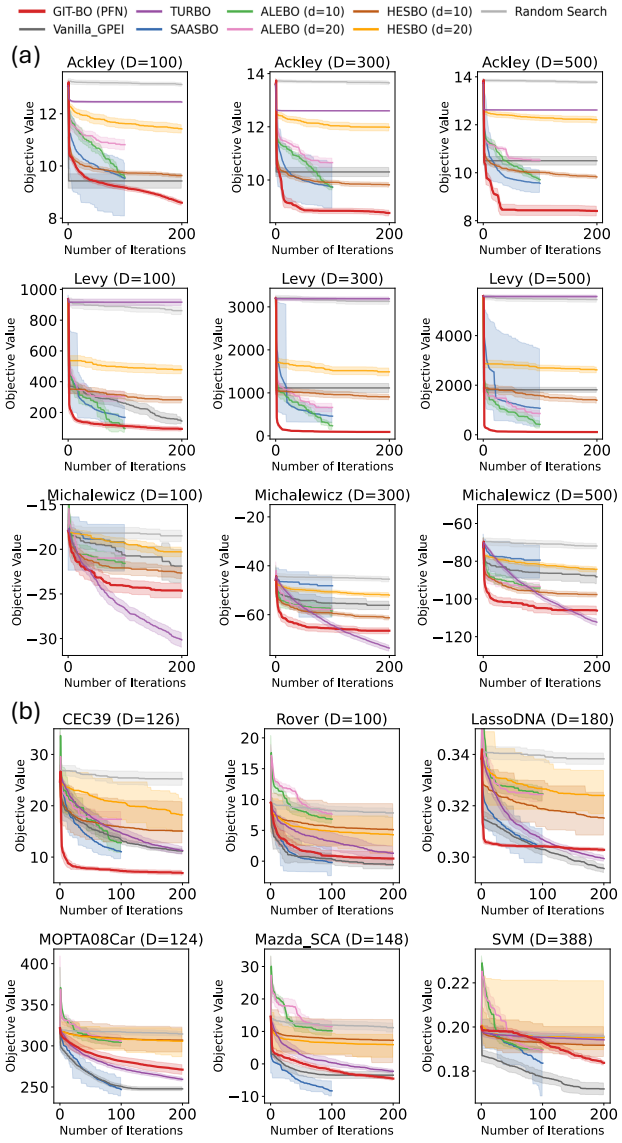

Figure 3. Optimization results on the (a) synthetic and (b) real-world benchmarks comparing GIT-BO against baseline algorithms. The solid line represents the median best function value achieved over 20 trials, with shaded regions indicating the 95% confidence interval. Out of the fifteen experiments here, GIT-BO ranks the top for six and the second for seven, showing better performance at higher-dimensional problems and faster convergence for the first 100 iterations. GIT-BO also has a lower variance compared to other baselines, meaning GIT-BO converges faster in expectation and delivers reliably strong results without requiring multiple restarts, reducing computational cost and risk. Full statistical tests and per-problem plots are provided in the Appendix.

5.2. Limitations

Our approach, although effective, faces several practical limitations. First, the computational requirements of TabPFN v2 present significant hardware constraints. As a large foundation model, TabPFN requires GPU acceleration for efficient inference. This limits accessibility for users without GPUs. Second, the current implementation imposes a hard dimensionality cap of 500D, as the foundation model was trained specifically with this input dimension limit. Problems exceeding this dimensionality cannot be directly addressed without architectural modifications or using ensembles. Third, although GIT-BO is largely parameter-free, the rank r of the GI subspace and the choice of reference point x_{ref} for back-projection could influence the algorithm's performance. We provide an initial sensitivity analysis in the Appendix. However, future work will explore using an automated heuristic or meta-learning of these hyperparameters.

5.3. Conclusion and Future Work

In this paper, we introduced Gradient-Informed Bayesian Optimization using Tabular Foundation Models (GIT-BO), a method integrating the computational efficiency of tabular foundation models with an adaptive, gradient-informed subspace identification strategy for Bayesian optimization. Our comprehensive evaluations across synthetic and real-world benchmarks demonstrate that GIT-BO consistently surpasses state-of-the-art Gaussian process-based methods in high-dimensional optimization scenarios (up to 500 dimensions). By eliminating iterative surrogate retraining and dynamically identifying critical subspaces, GIT-BO also accelerates convergence and reduces computational costs by orders of magnitude compared to traditional GP methods.

The capability to effectively handle previously intractable high-dimensional optimization tasks broadens the practical applicability of Bayesian optimization. This advancement is particularly impactful for automated machine learning pipelines, scientific experimentation, and complex engineering design problems where dimensionality has historically impeded efficient optimization. As tabular foundation models continue to grow in model size and improve the prediction accuracy, we anticipate even greater optimization performance from approaches like GIT-BO, further extending Bayesian optimization to even more challenging real-world scenarios. Future work can leverage our method in multiple directions, such as constrained, mixed-integer, and multi-objective optimization, by modifying the acquisition functions and embedding techniques.

Software and Data

We will soon migrate the code to reproduce every experiment to a public GitHub repository under an open-source license.

References

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21524–21538, 2020.

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012.

Brockhoff, D. and Tušar, T. Gecco 2023 tutorial on benchmarking multiobjective optimizers 2.0. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pp. 1183–1212, 2023.

Dewancker, I., McCourt, M., and Clark, S. Bayesian optimization for machine learning: A practical guidebook. *arXiv:1612.04858 [cs.LG]*, 2016.

Eriksson, D. and Jankowiak, M. High-dimensional bayesian optimization with sparse axis-aligned subspaces. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161, pp. 493–503. PMLR, 2021.

Eriksson, D., Pearce, M., Gardner, J., Turner, R. D., and Poloczek, M. Scalable global optimization via local bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

Hansen, N., Auger, A., Brockhoff, D., and Tušar, T. Anytime performance assessment in blackbox optimization benchmarking. *IEEE Transactions on Evolutionary Computation*, 26(6):1293–1305, 2022.

Hollmann, N., Müller, S., Purucker, L., Krishnakumar, A., Körfer, M., Hoo, S. B., Schirrmeister, R. T., and Hutter, F. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

Hvarfner, C., Hellsten, E. O., and Nardi, L. Vanilla Bayesian optimization performs great in high dimensions. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 20793–20817. PMLR, 2024.

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.

Klein, A., Falkner, S., Bartels, S., Hennig, P., and Hutter, F. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pp. 528–536. PMLR, 2017.

Kohira, T., Kemmotsu, H., Akira, O., and Tatsukawa, T. Proposal of benchmark problem based on real-world car structure design optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 183–184, 2018.

Kumar, A., Wu, G., Ali, M. Z., Mallipeddi, R., Suganthan, P. N., and Das, S. A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. *Swarm and Evolutionary Computation*, 56:100693, 2020.

Kumar, N., Mim, M. S., Dowling, A., and Zartman, J. J. Reverse engineering morphogenesis through bayesian optimization of physics-based models. *npj Systems Biology and Applications*, 10(1):49, 2024.

Letham, B., Calandra, R., Rai, A., and Bakshy, E. Re-examining linear embeddings for high-dimensional bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1546–1558, 2020.

Li, M. T., Cui, T., Li, F., Marzouk, Y., and Zahm, O. Sharp detection of low-dimensional structure in probability measures via dimensional logarithmic sobolev inequalities. *arXiv:2406.13036 [stat.ML]*, 2024a.

Li, M. T., Marzouk, Y., and Zahm, O. Principal feature detection via ϕ-sobolev inequalities. *Bernoulli*, 30(4): 2979–3003, 2024b.

Ly, A., Marsman, M., Verhagen, J., Grasman, R. P., and Wagenmakers, E.-J. A tutorial on fisher information. *Journal of Mathematical Psychology*, 80:40–55, 2017.

Malu, M., Dasarathy, G., and Spanias, A. Bayesian optimization in high-dimensional spaces: A brief survey. In *2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA)*, pp. 1–8. IEEE, 2021.

Müller, S., Feurer, M., Hollmann, N., and Hutter, F. PFNs4BO: In-context learning for Bayesian optimization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 25444–25470. PMLR, 2023.

Nayebi, A., Munteanu, A., and Poloczek, M. A framework for Bayesian optimization in embedded subspaces. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 4752–4761. PMLR, 2019.

Papenmeier, L., Nardi, L., and Poloczek, M. Increasing the scope as you learn: Adaptive bayesian optimization in nested subspaces. *Advances in Neural Information Processing Systems*, 35:11586–11601, 2022.

Picard, C. and Ahmed, F. Untrained and unmatched: Fast and accurate zero-training classification for tabular engineering data. *Journal of Mechanical Design*, 146(9), 2024.

Rana, S., Li, C., Gupta, S., Nguyen, V., and Venkatesh, S. High dimensional Bayesian optimization with elastic Gaussian process. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 2883–2891. PMLR, 2017.

Šehić, K., Gramfort, A., Salmon, J., and Nardi, L. Lassobench: A high-dimensional hyperparameter optimization benchmark suite for lasso. In *Proceedings of the First International Conference on Automated Machine Learning*, volume 188, pp. 2/1–24. PMLR, 2022.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104 (1):148–175, 2015.

Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 25, 2012.

Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., De Cola, G., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., KG, A., et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv:2407.17032 [cs.LG]*, 2024.

Wang, K. and Dowling, A. W. Bayesian optimization for chemical products and functional materials. *Current Opinion in Chemical Engineering*, 36:100728, 2022.

Wang, X., Jin, Y., Schmitt, S., and Olhofer, M. Recent advances in bayesian optimization. *ACM Computing Surveys*, 55(13s):1–36, 2023.

Wang, Z., Gehring, C., Kohli, P., and Jegelka, S. Batched large-scale bayesian optimization in high-dimensional spaces. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84, pp. 745–754. PMLR, 2018.

Wolpert, D. H. and Macready, W. G. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

Wu, J., Chen, X.-Y., Zhang, H., Xiong, L.-D., Lei, H., and Deng, S.-H. Hyperparameter optimization for machine learning models based on bayesian optimization. *Journal of Electronic Science and Technology*, 17(1):26–40, 2019.

Xu, Z., Wang, H., Phillips, J. M., and Zhe, S. Standard gaussian process is all you need for high-dimensional bayesian optimization. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yu, R., Picard, C., and Ahmed, F. Fast and accurate bayesian optimization with pre-trained transformers for constrained engineering problems. *Structural and Multidisciplinary Optimization*, 68(3):66, 2025.

Zahm, O., Cui, T., Law, K., Spantini, A., and Marzouk, Y. Certified dimension reduction in nonlinear bayesian inverse problems. *Mathematics of Computation*, 91(336): 1789–1835, 2022.

Zhang, Y., Apley, D. W., and Chen, W. Bayesian optimization for materials design with mixed quantitative and qualitative variables. *Scientific reports*, 10(1):4924, 2020.

A. The GIT-BO Algorithm Details

In this section, we detail our GIT-BO high-dimensional BO framework. The framework consists of four main components: the surrogate model (TabPFN v2), the gradient-based subspace identification, the choice of acquisition function, and the algorithm that ties these together.

A.1. Surrogate Modeling with TabPFN

We use TabPFN, specifically the 500-dimensional TabPFN v2 TFM model from (Hollmann et al., 2025), as our surrogate model to predict the objective function values using in-context learning. The optimization objective is defined as minimizing an unknown function $f(x)$ over a domain $x \in \mathcal{X} \subset \mathbb{R}^D$. At any BO iteration, let $D_n = (x_i, y_i)_{i=1}^n$ be the dataset of points sampled so far and their observed function values $y_i = f(x_i)$. In a standard BO setting, one would fit a GP posterior $p(f|\mathcal{D}n)$ to these data. Here instead, we leverage TabPFN to obtain a predictive model. We provide the dataset D_n to TabPFN as input context (taking a set of labeled examples as part of its input sequence) and query the model for its prediction at any candidate point x. TabPFN then returns an approximate posterior predictive distribution PFN$(y \mid x, D_n)$ informed by both the new point x and the context of observed samples. In practice, TabPFN produces a prediction in a single forward pass as an approximation to the Bayesian posterior mean $\mu_n(x)$ for $f(x)$ as this predictive mean given the provided data. In our method, we review the previous series of PFN work (Müller et al., 2023) and implement the calculation of predictive mean $\mu_n(x)$ and predictive variance $\sigma_n^2(x)$ with the latest TabPFN v2 regressor model.

A.2. Gradient-Informed Subspace Identification and Sampling

For active subspace identification, we leverage gradient information $\nabla_x \mu_n(x)$ obtained through one-step backpropagation on TabPFN's predictive mean values, which naturally varies across data points at each iteration. Inspired by studies on dimension reduction in nonlinear Bayesian inverse problems, which employs techniques ϕ-Sobolev inequalities and gradient-based approaches, we approximate the diagnostic (Fisher information) matrix $H = \mathbb{E}_\mu[\nabla_x \mu_n(x)\, \nabla_x \mu_n(x)^\top]$ (Zahm et al., 2022; Li et al., 2024a;b; Ly et al., 2017). In this study of high-dimensional problems with only $D > 100$, we select the top r $(D >> r)$ dominant eigenvectors in H as the principal vectors V_r that span our gradient-informed active subspace (GI subspace). Next, we restrict the next exploration to this subspace and map the search candidates back to the original space by $x_{\text{cand}} = \bar{x}_{ref} + V_r z,\ z \sim \mathrm{U}([-1, 1]^r)$, where the reference point $\bar{x}_{ref}$ is the centroid of the observed data. Finally, we pass x_{cand} to the acquisition function for determining the next sample to evaluate. In GIT-BO, we recalculate the GI subspace at each iteration as the subspace changes when observed data increases. For computational practicality and exploration-exploitation balance, we default to $r = 15$ in GIT-BO; a detailed sensitivity analysis on the choice of r is presented in the Appendix.

A.3. Acquisition Function

We implement the Thompson Sampling (TS) acquisition as our acquisition function due to its previous success in high-dimensional BO (Eriksson et al., 2019). For TS, we approximate it by sampling from the predictive distribution at each sample x and draw a fixed number of 512 random samples $\tilde{f}(\cdot)$ from the surrogate's posterior. The candidate x_{next} with the highest sampled value is selected as the next query $\tilde{f}(x_{\text{next}})$. This leverages the full predictive uncertainty and tends to balance exploration and exploitation implicitly (Shahriari et al., 2015; Eriksson et al., 2019). The key difference here is that we restrict x to our learned GI subspace. After selecting the next query point x_{next}, we evaluate the true objective function to obtain y_{next}.

A.4. GIT-BO Implementation Details

The GIT-BO algorithm was implemented using Python 3.12 with the TabPFN v2.0.6 implementation and model (link: `https://github.com/PriorLabs/TabPFN` and `https://huggingface.co/Prior-Labs/TabPFN-v2-reg`, license: Prior Lab License (a derivative of the Apache 2.0 license (`http://www.apache.org/licenses/`))). Instead of using the off-the-shelf TabPFN code for GIT-BO, we make a wrapper that converts the numpy calculations to PyTorch for `torch.backward()` gradient calculations. The system uses BoTorch v0.12.0 and PyTorch 2.6.0+cu126 for the underlying optimization framework. For the gradient-informed subspace identification, a rank of 15 was selected for the GI subspace matrix. All experiments were conducted on a GNU/Linux 6.5.0-15-generic x86_64 system running Ubuntu 22.04.3 LTS as the operating system, ensuring a consistent computational environment across all benchmark tests.

B. Experiment Details

This section describes how we evaluate GIT-BO's performance on a diverse collection of synthetic optimization benchmarks and real-world engineering problems.

B.1. Experiment Setups

This section outlines our empirical approach to evaluating and comparing different high-dimensional Bayesian optimization algorithms, specifically focusing on assessing their performance across various complex synthetic and engineering benchmarks.

B.1.1. Benchmark Algorithms

We benchmark GIT-BO against random search (Bergstra & Bengio, 2012) and four high-dimensional BO methods, including SAASBO (Eriksson & Jankowiak, 2021), TURBO (Eriksson et al., 2019), the Vanilla GP for high-dimensional BO that recently integrated in Botorch (Hvarfner et al., 2024), HESBO (Nayebi et al., 2019), and ALEBO (Letham et al., 2020). For HESBO and ALEBO, we varied $d_{\text{embedding}} \in \{10, 20\}$ and denote the choice as HESBO/ALEBO (d=$\{10, 20\}$). The implementation of SAASBO and TURBO is taken from BoTorch tutorials (Balandat et al., 2020). The implementation of HESBO and ALEBO are taken from their original published paper and code (Nayebi et al., 2019; Letham et al., 2020). Additional details of the algorithm implementation are listed in the supplemental material.

B.1.2. Test Problems

This study incorporates a diverse set of high-dimensional optimization problems, including 11 synthetic problems and 12 real-world benchmarks. Synthetic and scalable problems include: Ackley, Rosenbrock, Dixon-Price, Levy, Powell, Griewank, Rastrigin, Styblin-Tang, and Michalewicz. Additionally, we tested on two Synthetic problems in LassoBench (LassoSyntheticMedium and LassoSyntheticHigh) and one real-world hyperparameter optimization (HPO) problem LassoDNA. The rest of the application problems are collected from previous optimization studies and conference benchmarks: the power system optimization problems in CEC2020, Rover, SVM HPO, MOPTA08 car problem, and two Mazda car problems. As this study focus on the high-dimensional characteristic of problem, we make all our benchmark problems single and unconstrained for testing. Therefore, we have applied penalty transforms to all real-world problems with constraints and perform average weighting to the two multi-objective Mazda problems. The details of benchmark transformation and implementation are detailed in the Appendix. Among the 23 benchmarks, 10 (Synthetic + Rover) are scalable problems. To evaluate the algorithms' performance with respect to dimensionality, we solve the scalable problems for $D = \{100, 200, 300, 400, 500\}$. Therefore, we have experimented with a total of $5 \times 10 + 13 = 63$ different variants of the benchmark problems. The benchmarks' details and implementations are listed in the Appendix.

B.1.3. Algorithm Tests

The algorithm evaluation aims to thoroughly compare GIT-BO to current SOTA Bayesian optimization techniques. This study focuses on minimizing the objective function for the given test problems. For each test problem, our experiment consists of 20 independent trials, each utilizing a distinct random seed. To ensure fair comparison, we initialize each algorithm with an identical set of 200 samples, generated through Latin Hypercube Sampling with consistent random seeds across all trials. During each iteration, each algorithm selects one sample to evaluate next.

To execute this extensive benchmarking process, we utilized a distributed server infrastructure featuring Intel Xeon Platinum 8480+ CPUs and NVIDIA H100 GPUs. Individual experiments were conducted with the same amount of compute allocated: a single H100 GPU node with 24 CPU cores and 250GB RAM.

B.2. Evaluation Metrics

B.2.1. Optimization Fixed-budget Convergence Analysis

Fixed-budget evaluations is a technique for comparing the efficiency of optimization algorithms by allotting specific computational resources for their execution (Hansen et al., 2022).

We apply a "fixed-iteration" budget approach in our study, initially running all algorithms (GIT-BO, TurBO, HESBO, SAASBO, and ALEBO) for 100 iterations. Additionally, we run GIT-BO, TurBO, and HESBO for an extra 100 iterations

(total of 200 iterations) because these methods are roughly two orders of magnitude faster than SAASBO and ALEBO, allowing us to explore their long-term convergence behavior more thoroughly.

B.2.2. Statistical Ranking

For comprehensively compare and evaluate the performance of the Bayesian optimization algorithms, statistical ranking techniques are employed instead of direct performance measurements of the optimization outcome. In this study, we define the optimization performance result as the median of the minimum result found (final incumbent) across the 20 optimization trials of each algorithm. By statistically ranking the results, we were able to standardize the comparisons across different problems, since various optimization challenges can produce objective values of vastly different magnitudes. Furthermore, using this ranking allowed us to reduce the distorting effects of unusual or extreme data points that might influence our evaluation.

We conduct our statistical analysis using the Friedman and Wilcoxon signed-rank tests, complemented by Holm's alpha correction. These non-parametric approaches excel at processing benchmarking result data without assuming specific distributions, which is critical for handling optimization results with outliers. These statistical methods effectively handle the dependencies in our setup, where we used the same initial samples and seeds to test all algorithms. The Wilcoxon signed-rank test addresses paired comparisons between algorithms, while the Friedman test manages problem-specific grouping effects. For multiple algorithm comparisons, we used Holm's alpha correction to control error rates (Ismail Fawaz et al., 2019; Brockhoff & Tušar, 2023; Picard & Ahmed, 2024).

B.2.3. Algorithm Runtime Recording

Each algorithm is timed for the total time it takes to run one experiment trial. We take the overall mean over the 20 trials and over all benchmark problems, resulting in a single value average time (t_{avg}) to represent the average runtime of each algorithm.

C. Benchmark Algorithms and Problems Implementation Details

C.1. GP-based Benchmark Algorithms Implementation Details

We benchmark GIT-BO against five high-dimensional BO methods, including SAASBO (Eriksson & Jankowiak, 2021), TURBO (Eriksson et al., 2019), Vanilla GPEI (Hvarfner et al., 2024), HESBO (Nayebi et al., 2019), and ALEBO (Letham et al., 2020).

- SAASBO: The implementation is taken from (Balandat et al., 2020) (link: `https://github.com/pytorch/botorch/blob/main/tutorials/saasbo/saasbo.ipynb`, license: MIT license, last accessed: May 1st, 2025)
- TURBO: The implementation is taken from (Balandat et al., 2020) (link: `https://github.com/pytorch/botorch/blob/main/tutorials/turbo_1/turbo_1.ipynb`, license: MIT license, last accessed: May 1st, 2025)
- Vanilla GPEI: The implementation is taken from (Balandat et al., 2020) version 13 (link: `https://github.com/pytorch/botorch/discussions/2451`, license: MIT license, last accessed: May 1st, 2025)
- HESBO: The implementation is taken from the original implementation (Nayebi et al., 2019) (link: `https://github.com/aminnayebi/HesBO`, license: none, last accessed: July 1st, 2025)
- ALEBO: The implementation is taken from the original implementation (Letham et al., 2020) (link: `https://github.com/facebookresearch/alebo`, license: CC BY-NC 4.0, last accessed: May 1st, 2025)

As most BO code is archived in the BoTorch or Ax Platform[1] library, we use Botorch 0.12.0 for SAASBO and TURBO. However, to successfully replicate ALEBO, we make another environment using Ax 0.1.17 and other incompatible packages.

C.2. Benchmark Problems Implementaion Details

The source and license details of our benchmark problems are described in the following paragraphs.

[1] `https://github.com/facebook/ax`

C.2.1. SYNTHETIC PROBLEMS:

The implementations for the nine synthetic functions are taken from Botorch (Balandat et al., 2020) (link: https://github.com/pytorch/botorch/blob/main/botorch/test_functions/synthetic.py, license: MIT license, last accessed: May 1st, 2025). The bounds of each problems are the default implementation in Botorch. Detailed equations for each problem can be found here: https://www.sfu.ca/~ssurjano/optimization.html.

C.2.2. CEC2020 POWER SYSTEM PROBLEMS:

We examine a subset of six problems, specifically those with design spaces exceeding 100 dimensions, from the CEC2020 test suite (Kumar et al., 2020) (link: https://github.com/P-N-Suganthan/2020-Bound-Constrained-Opt-Benchmark, license: no licence, last accessed: May 1st, 2025). The code is initially in MATLAB, and we translate it into Python, running pytest to ensure the implementations are correct. While these problems incorporate equality constraints ($h_j(x)$), they are transformed into inequality constraints ($g_j(x)$) using the methodology outlined in the original paper (Kumar et al., 2020), as constraint handling is not the primary focus of this research. These transformed constraints are subsequently incorporated into the objective function $f(x)$ as penalty terms.

$$g_j(x) = |h_j(x)| - \epsilon \leq 0 \ , \ \epsilon = 10^{-4} \ , \ j = 1 \sim C$$

$$f_{penalty}(x) = f(x) + \rho \sum_{j=1}^{C} max(0, g_j(x))$$

Each problem has different ρ penalty factor respectively to make the objective and constraints values have similar effect on $f_{penalty}(x)$.

Table 1. CEC2020 Benchmark Problems Penalty Transform Factor ρ

Problems	CEC34	CEC35	CEC36	CEC37	CEC38	CEC39
ρ	0.01	0.0002	0.001	0.04	0.02	0.04

C.2.3. LASSOBENCH BENCHMARKS:

The implementation is taken from the original implementation (Šehić et al., 2022) (link: https://github.com/ksehic/LassoBench, license: MIT and BSD-3-Clause license, last accessed: May 1st, 2025).

C.2.4. ROVER:

The implementation is taken from (Wang et al., 2018) (link: https://github.com/zi-w/Ensemble-Bayesian-Optimization, license: MIT license, last accessed: May 1st, 2025).

C.2.5. MOPTA08 CAR:

The MOPTA08 executables are taken from the paper (Papenmeier et al., 2022)'s personal website (link: https://leonard.papenmeier.io/2023/02/09/mopta08-executables.html, license: no license, last accessed: May 1st, 2025). The MOPTA08 Car's penalty transformation follows the formation of (Eriksson & Jankowiak, 2021)'s supplementary material.

C.2.6. MAZDA BECHMARK PROBLEMS:

The implementation is taken from (Kohira et al., 2018) (link: https://ladse.eng.isas.jaxa.jp/benchmark/, license: no license, last accessed: May 1st, 2025). The Mazda problem has two raw forms: a 4-objectives problem 148D (Mazda_SCA) and a 5-objectives 222D problem (Mazda), and both of them have inequality constraints. For both problems, we equally weight each objective to form a single objective and perform a penalty transform:

$$f_{multiobj\_penalty}(x) = \frac{1}{N}\sum_{i=1}^{N} f(x) + \rho \sum_{j=1}^{C} max(0, g_j(x))$$

where N is the number of objectives, C is the number of inequality constraints, and we use $\rho = 10$ for both variants of Mazda problem.

Table 2 summarizes the type of problems and their respective tested dimensions.

Table 2. High-Dimensional Benchmark Problems

Problems	Source	Type	Dimension (D) Tested
Ackley	Botorch (Balandat et al., 2020)	Synthetic	100, 200, 300, 400, 500
Dixon-Price	Botorch (Balandat et al., 2020)	Synthetic	100, 200, 300, 400, 500
Griewank	Botorch (Balandat et al., 2020)	Synthetic	100, 200, 300, 400, 500
Levy	Botorch (Balandat et al., 2020)	Synthetic	100, 200, 300, 400, 500
Michalewicz	Botorch (Balandat et al., 2020)	Synthetic	100, 200, 300, 400, 500
Powell	Botorch (Balandat et al., 2020)	Synthetic	100, 200, 300, 400, 500
Rastrigin	Botorch (Balandat et al., 2020)	Synthetic	100, 200, 300, 400, 500
Rosenbrock	Botorch (Balandat et al., 2020)	Synthetic	100, 200, 300, 400, 500
Styblinski-Tang	Botorch (Balandat et al., 2020)	Synthetic	100, 200, 300, 400, 500
CEC34	CEC2020 Benchmark Suite (Kumar et al., 2020)	Real-World	118
CEC35	CEC2020 Benchmark Suite (Kumar et al., 2020)	Real-World	153
CEC36	CEC2020 Benchmark Suite (Kumar et al., 2020)	Real-World	158
CEC37	CEC2020 Benchmark Suite (Kumar et al., 2020)	Real-World	126
CEC38	CEC2020 Benchmark Suite (Kumar et al., 2020)	Real-World	126
CEC39	CEC2020 Benchmark Suite (Kumar et al., 2020)	Real-World	126
LassoSyntMedium	LassoBench (Šehić et al., 2022)	Synthetic	100
LassoSyntHigh	LassoBench (Šehić et al., 2022)	Synthetic	300
LassoDNA	LassoBench (Šehić et al., 2022)	Real-World	180
Rover	Previous BO studies (Wang et al., 2018)	Real-World	100, 200, 300, 400, 500
MOPTA08 CAR	Previous BO studies (Eriksson et al., 2019)	Real-World	124
MAZDA	Mazda Car Bechmark (Kohira et al., 2018)	Real-World	222
MAZDA SCA	Mazda Car Bechmark (Kohira et al., 2018)	Real-World	148
SVM	Previous BO studies (Eriksson & Jankowiak, 2021)	Real-World	388
HalfCheetah	Gymnasium's Mujoco (Towers et al., 2024)	Real-World	102

D. Ablation Studies on Acquisition and Gradient Subspace

To rigorously assess the necessity of gradient-informed subspace exploration, we conducted an ablation study comparing GIT-BO with a variant employing the vanilla TabPFN v2 model without adaptive gradient-informed subspace identification. Two acquisition functions, expected improvement acquisition (EI) (used in the original PFNs4BO (Müller et al., 2023)) and Thompson Sampling (TS), are tested with the vanilla TabPFN v2 and TabPFN v2 + GI subspace, resulting in four algorithms being ablated: TabPFNv2 + EI, TabPFNv2 + TS, TabPFNv2 + EI + GI subspace, and TabPFNv2 + TS + GI subspace (which is our GIT-BO method).

The results of the acquisition funciton ablation convergence plot shown in Figure 4 clearly indicate that both vanilla TabPFN v2 with EI and TS have failed the high-dimensional search significantly compared to GIT-BO. Specifically, we observe a drastic degradation in convergence speed and final optimization outcomes. This confirms our hypothesis that vanilla TabPFN v2, regardless of the selection of the acquisition function, fails to capture crucial directions in high-dimensional spaces without the assistance of the GI subspace. This reinforces the critical role of our proposed gradient-based refinement strategy in achieving robust performance. With the GI subspace, the use of the TS or EI acquisition function has minor differences in performance for the engineering problems tested (CEC34 and Mopta08Car). However, looking at Ackley's results TS acquisition shows slight advantages for this optimization problem with many local minima. While we proceed on

using TS acquisition for the GIT-BO algorithm, we believe more research into more acquisition functions (probability of improvement, upper confidence bound, entropy search, ...) in the future would benefit specific use cases.

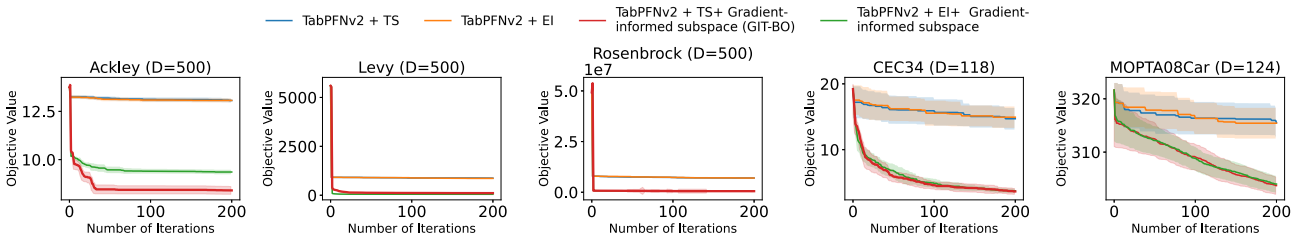

Figure 4. Ablation studies of vanilla TabPFNv2 BO with GIT-BO. GIT-BO, regardless of the acquisition functions, consistently outperform other algorithms without GI subspace search across all tested problems (8.6 times better regrets), indicating GIT-BO is the best approach and vanilla TabPFNv2 is not an ideal candidate for performing high-dimensional BO.

E. Ablation Studies over Gradient Subspace Sampling

We conducted an ablation study to evaluate the impact of three different gradient-informed (GI) subspace sampling methods on GIT-BO's optimization performance: uniform (default), random, and Sobol sampling. Figure 5 shows the comparative convergence results.

Our findings indicate mixed results without a universally optimal sampling strategy. Uniform sampling generally provided stable and reliable convergence, while random sampling occasionally achieved better outcomes but with greater variance, similar as Sobol sampling. These observations highlight the potential for adaptive strategies in selecting GI subspace sampling methods based on problem-specific characteristics, representing an important area for future exploration.

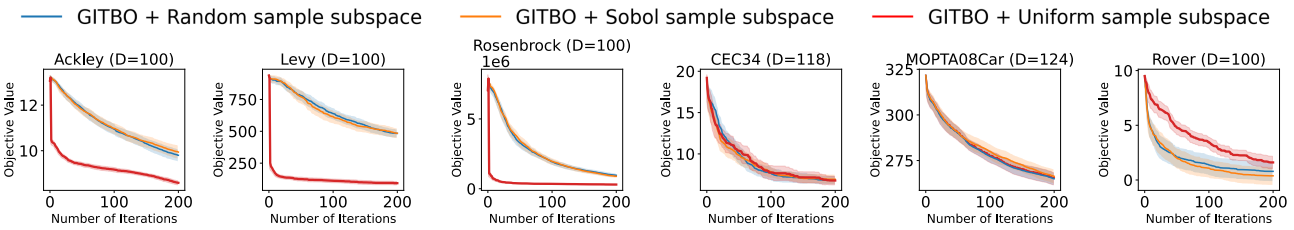

Figure 5. Comparative convergence of uniform, random, and Sobol sampling strategies within the GI subspace on selected benchmarks. Shaded regions represent 95% confidence intervals over 20 trials. Random and Sobol sampling can achieve similar or superior performance than uniform sampling GI subspace in engineering problems, while struggling at the synthetic tasks.

F. Parameter Sweep

We conducted a parameter sweep to evaluate the sensitivity of GIT-BO's optimization performance to the chosen dimensionality (r) of the gradient-informed active subspace The results are presented in Figure 7. The empirical findings suggest that very high-dimensional subspaces (e.g., $r = 40$ for 100-dimensional problems) result in significant performance degradation due to diminished effectiveness of the gradient-informed search direction. Conversely, lower-dimensional subspaces generally showed superior performance. Our default selection, $r = 15$, represents a balanced compromise between efficiency and thoroughness, positioned roughly in the middle of performance rankings. Investigating adaptive or context-dependent hyperparameter tuning strategies for r presents an exciting avenue for future work.

G. Full Results

We present the comprehensive optimization results for all benchmark problems investigated in the main paper. Figures 9 to 12 provide the complete convergence curves for synthetic and real-world benchmarks, respectively. The results consistently

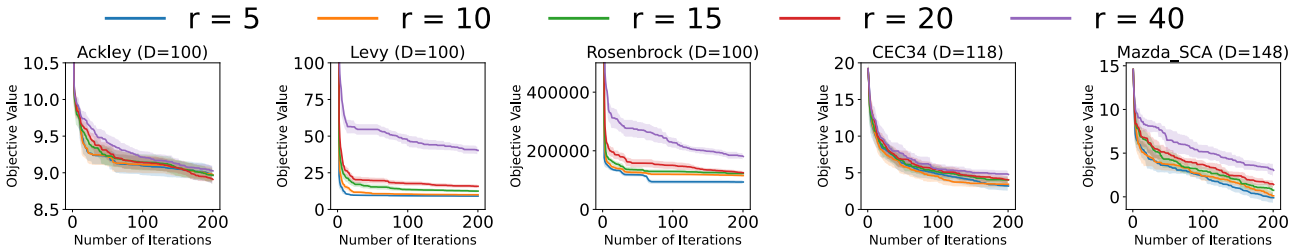

Figure 6. Performance comparison across various gradient-informed subspace dimensions (r) in GIT-BO. Median optimization results across 20 trials are shown, with shaded regions depicting 95% confidence intervals. High-dimensional subspaces (e.g., $r = 40$) exhibit notable performance deterioration, and lower subspace dimensions tend to perform consistently better.

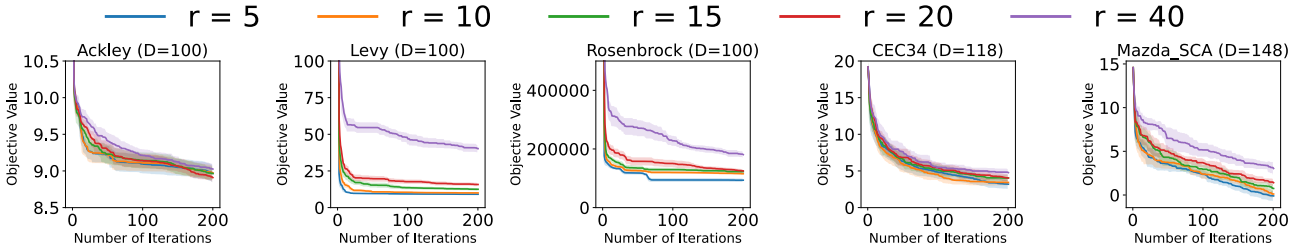

Figure 7. Performance comparison across various gradient-informed subspace dimensions (r) in GIT-BO. Median optimization results across 20 trials are shown, with shaded regions depicting 95% confidence intervals. High-dimensional subspaces (e.g., $r = 40$) exhibit notable performance deterioration, and lower subspace dimensions tend to perform consistently better.

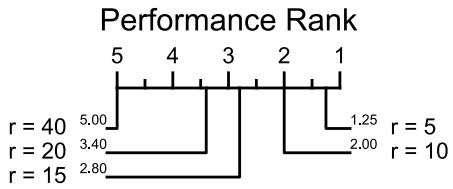

Figure 8. Rank of the final optimization solution performance comparison across various gradient-informed subspace dimensions (r) in GIT-BO.

demonstrate GIT-BO's robust performance across different classes of high-dimensional problems, clearly indicating its advantage in both convergence speed and final solution quality compared to baseline methods. Notably, GIT-BO achieves significant improvements on complex synthetic functions and demonstrates remarkable consistency on real-world engineering benchmarks, validating our approach's generalizability and practical efficacy.

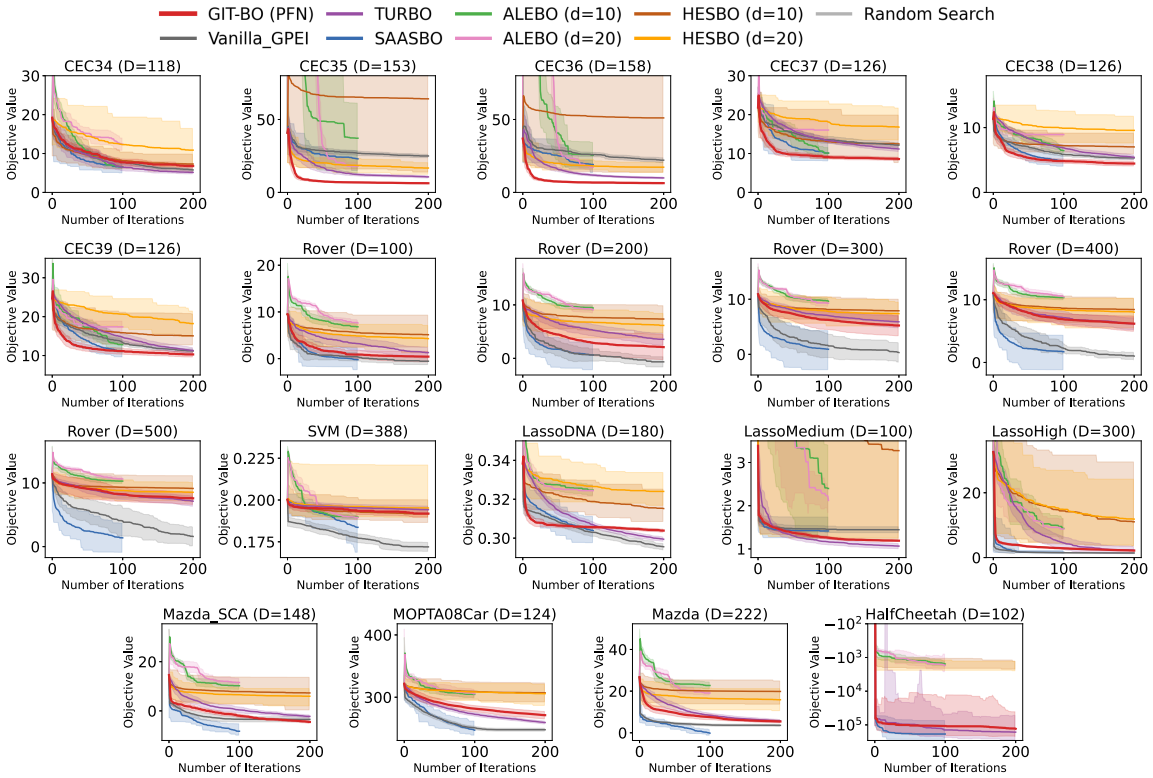

Figure 9. Full convergence curves on engineering benchmarks illustrating the median best solutions obtained across multiple trials. Confidence intervals (95%) are shown as shaded areas around the median lines. The minimal median best solution among all algorithms at the final iteration is marked in "x". The results demonstrate that SAASBO and Vanilla GPEI excels in the Rover and car (MOPTA08 and Mazdas) problems, and GIT-BO leads the CEC power system optimization problem. Though GIT-BO is second-best for real-world problems, engineers might benefit from its 100x faster runtime if time is a hard constraint.

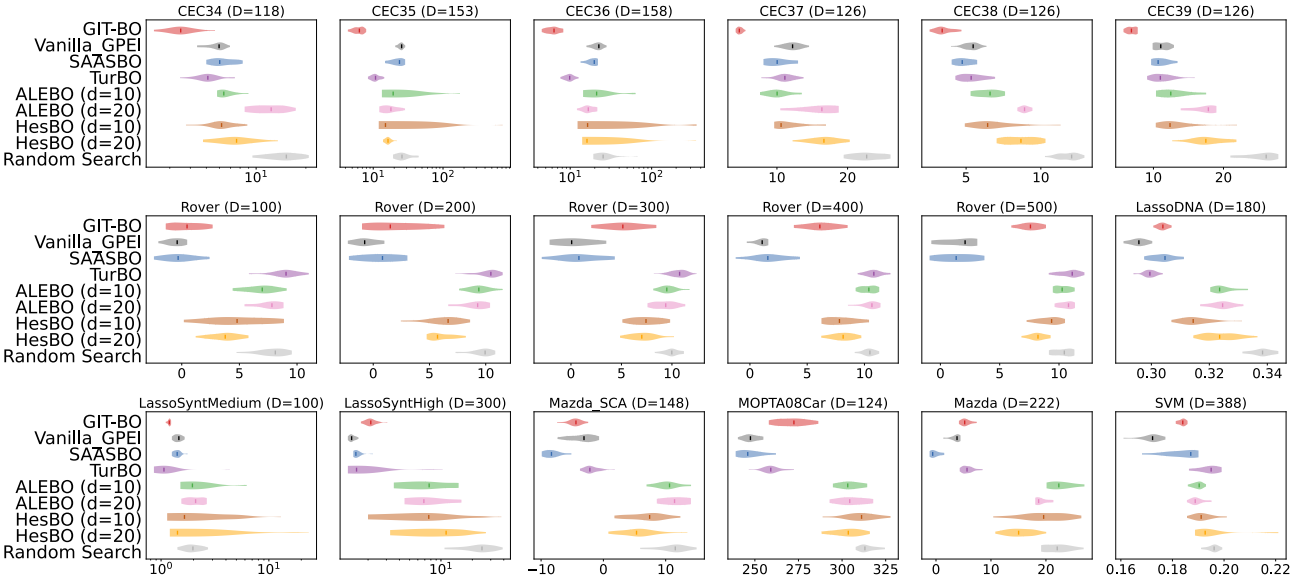

Figure 10. Violin plot statistical summary of final optimization performance for engineering functions, summarizing results over 20 trials. GIT-BO exhibits less variance and top-or-second median results on most high-dimensional problems (except for Rover).

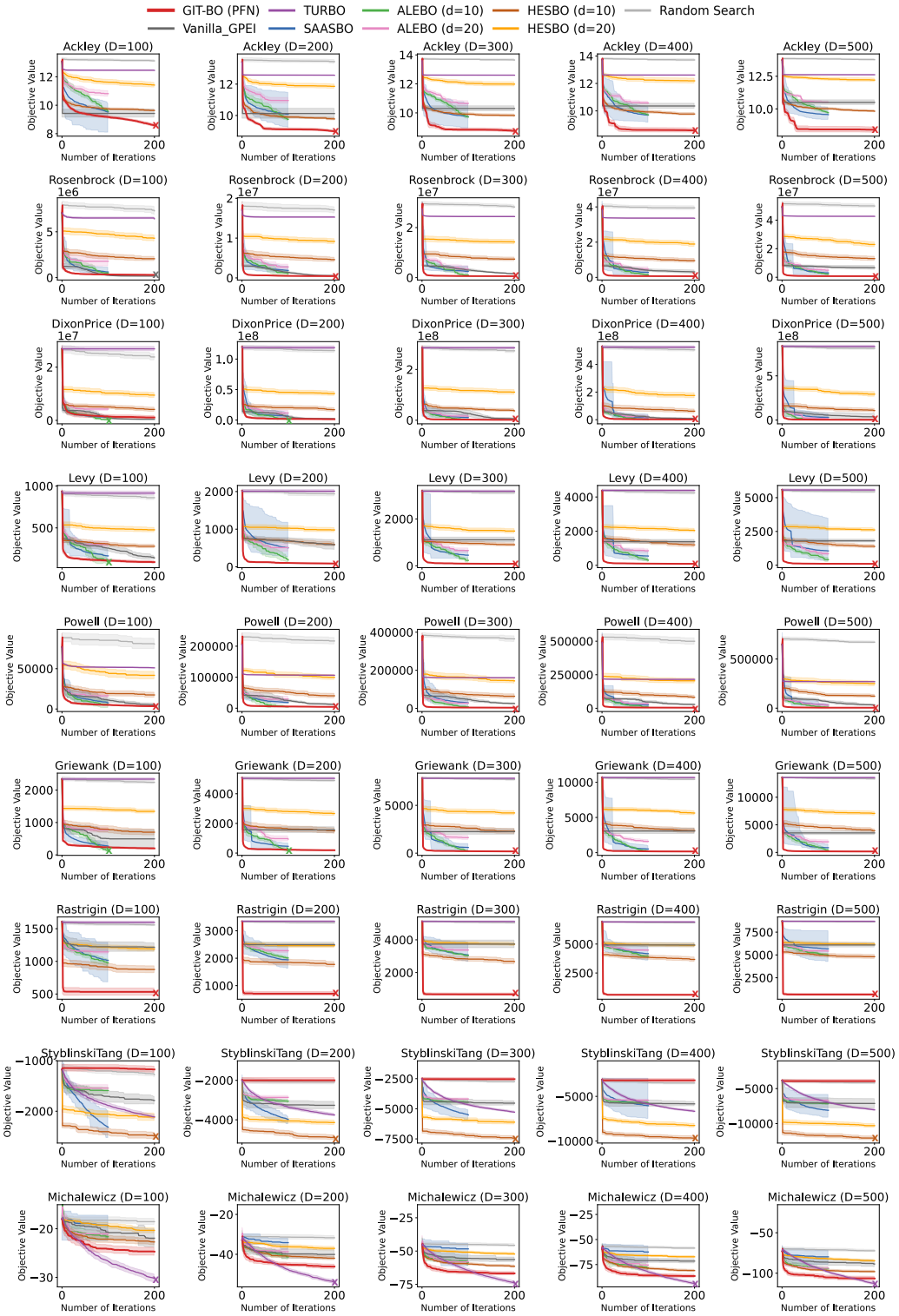

Figure 11. Detailed optimization results for synthetic benchmarks demonstrating the relative performance of GIT-BO versus competing algorithms. Median best-found function values are illustrated by solid lines, with shaded bands denoting 95% confidence intervals. The minimal median best solution among all algorithms at the final iteration is marked in "x". Results highlight the robustness of GIT-BO in achieving superior performance across diverse synthetic problems, especially in challenging higher dimensional settings (D >300). The only two exceptions that GIT-Bo fails dramatically in all Ds are Styblinski-Tang and Michalewicz, which can potentially be explained by the "No Free Lunch" theorem (Wolpert & Macready, 1997).

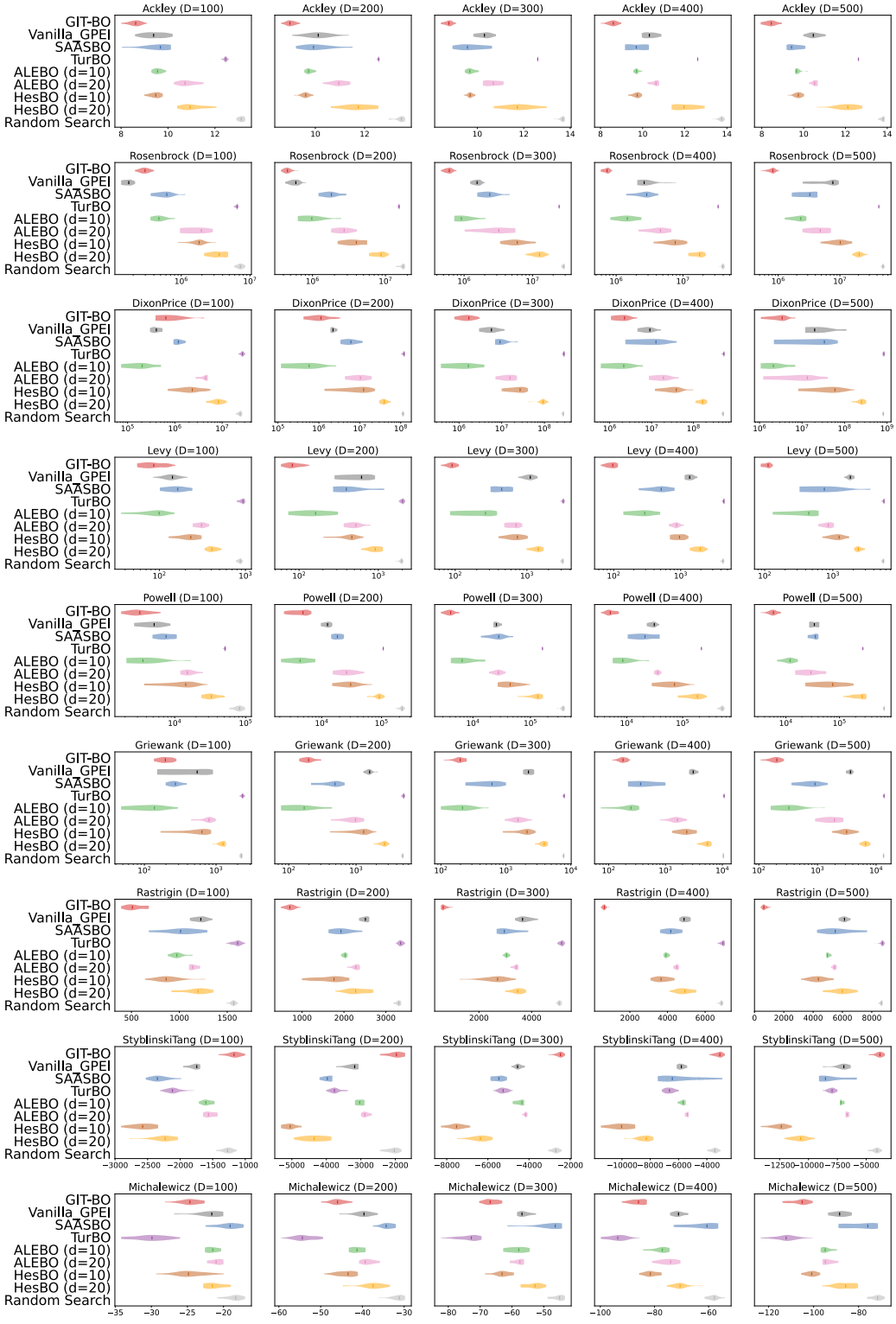

Figure 12. Violin plot representations of final optimization performance for synthetic functions, summarizing results over 20 trials. GIT-BO shows narrower variance compared to the second-ranked (SAASBO) method and improved median performance across most benchmarks, underscoring its effectiveness in reliably navigating complex synthetic landscapes.