Robust Multi-task Modeling for Bayesian Optimization via In-Context Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Bayesian optimization is a sample-efficient optimization technique for black-box optimization, and leveraging historical information from related tasks can greatly improve its performance. Gaussian processes (GPs) are commonly used to model this multi-task data; however, they trade off complexity with expressivity. Jointly modeling all tasks can be computationally infeasible for GPs, while scalable approaches may fail to effectively utilize inter-task relationships. Moreover, these methods are often prone to negative transfer, where the inclusion of unrelated tasks degrades predictive performance. In this paper, we present Multi-Task Prior-Data Fitted Networks (MTPFNs), a multi-task model that efficiently and jointly models all tasks and data points. We show that MTPFNs serve as a compelling surrogate model that is robust to negative transfer, and their flexibility enables more efficient exploration. We demonstrate the effectiveness of our approach across a variety of synthetic and real-world benchmarks including hyperparameter optimization.

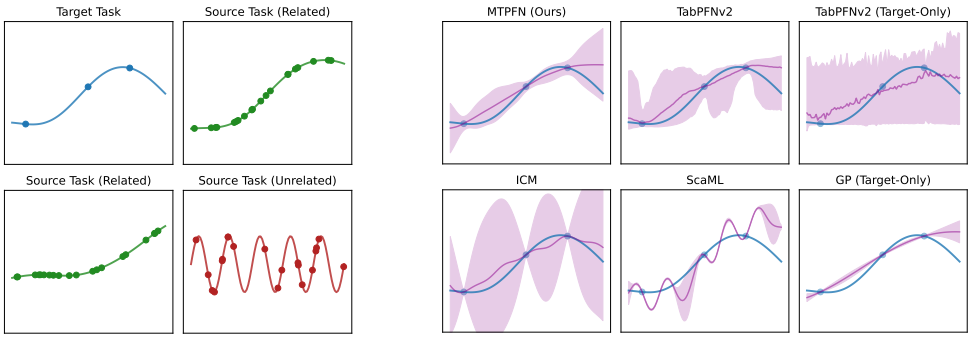
1 Introduction

Optimizing expensive black-box functions is critical across scientific and engineering domains, and important applications include as hyperparameter tuning for machine learning models [29], chemical reaction optimization [28], and engineering design problems [21]. For these resource-intensive settings, Bayesian optimization (BO) is a sample-efficient method that aims to find the global optimum with minimal function evaluations by using probabilistic surrogate models to select future evaluations [12].

Transfer learning can be used to further improve sample efficiency by extracting information from related tasks. Existing transfer-learning BO approaches often use a surrogate to jointly model the data from different tasks. However, these models face limitations. The commonly-used Gaussian process (GP) surrogate models often trade off with data efficiency with robustness: the models which jointly fit all of the data [4, 30, 35, 26, 18] may make strong assumptions about how different tasks are correlated and experience *negative transfer*, where unrelated tasks hinder performance; meanwhile, other approaches which fit separate GPs per task and ensemble their predictions [13, 10, 33, 7, 31] are more robust to negative transfer but cannot jointly capture cross-task information. Furthermore, multi-task GP may become computationally infeasible as we scale the number of tasks and data points, while more flexible variants may fail to efficiently utilize the full dataset.

Neural-network based approaches are an attractive alternative to Gaussian Processes. Transformer neural processes (TNP) [24] and prior fitted networks (PFNs) [22] are transformers trained to approximate the posterior predictive distribution given a data generating function, and they are capable of efficiently approximating the posterior for complex and bespoke priors. While TNPs and PFNs have successfully applied to Bayesian optimization in the single-task setting [23, 25], there has been no prior work which explores the use of in-context transfer of related tasks to accelerate

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.



(a) Four-task problem setting

(b) Model predictions on the target task

Figure 1: **MTPFNs effectively transfer information from related tasks while remaining robust to unrelated tasks.** Compared to joint models such as ICM, ScaML, and TabPFNv2 with a categorical task variable, our MTPFN demonstrates improved robustness to the unrelated source task (red). The MTPFN is also able to borrow strength from the related source tasks (green) and outperforms models which only consider the target task. We plot the mean and 95% confidence intervals for each model.

optimization. For more details, we provide a comprehensive background on Bayesian optimization, multi-task GPs, neural-network approaches, and other related work in Appendix A.

We propose Multi-Task Prior-data Fitted Networks (MTPFNs), which use a novel data generating process that samples data from both related and unrelated tasks. MTPFNs achieve better generalization on classical transfer tasks compared to previous GP-based approaches, while being more robust to negative transfer from irrelevant auxiliary sources. We demonstrate the efficacy of MTPFNs as a surrogate model in Bayesian optimization on synthetic problems and real-world AutoML benchmarks.

2 Method

In this section, we present the Multi-Task Prior-Data Fitted Network (MTPFN), a scalable model that transfers relevant knowledge from auxiliary tasks to model a target task via in-context learning.

2.1 Data Generation Process

PFNs are trained to approximate the posterior of a data generation process (DGP), and the design of this prior significantly influences predictive performance. While various DGPs have been proposed in previous works [e.g. 1, 15, 23], the multi-task setting poses unique challenges: there are complex relationships between tasks where information may transfer through shared latent structures, and real-world secnarios frequently contain unrelated tasks which should not negatively impact performance. To address these challenges, we propose a multi-task DGP that learns complex relationships between relevant tasks while mitigating corruption from irrelevant tasks.

Our approach combines two key insights: when tasks are truly related, strong transfer can be achieved by sharing statistical properties like lengthscales across tasks; however, since we cannot know a priori which auxiliary tasks will be helpful, the model must explicitly account for task irrelevance. Our DGP models task relationships through a shared covariance structure when tasks are related and introduces a probability $p \in [0, 1]$ that each auxiliary task is instead modeled independently when it would be irrelevant or harmful. In Figure 1, existing GP models and PFN priors perform poorly in this synthetic setting due to influence from the unrelated task in red. In contrast, our MTPFN is robust to the irrelevant task and accurately mirrors the true behavior, demonstrating the benefit of our robust prior generation procedure. We present the full algorithm in Algorithm A.1, with additional discussion and ablations for alternative DGPs in Appendix B.

2.2 Hierarchical Attention Mechanism

For multi-task regression problems, it is important to consider how the task itself should be encoded within the model. Two natural approaches include representing the associated task as a *categorical feature* or a continuous *task embedding*. However, these methods have distinct limitations: they

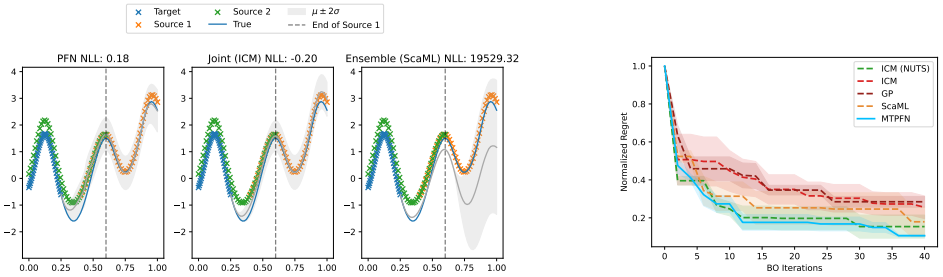


Figure 2: **MTPFNs jointly model the data from the target and all auxiliary tasks and perform fully Bayesian inference**. **(Left)**: MTPFNs perform similarly to other joint models like ICM and outperform ensemble-based models like ScaML. **(Right)**: MTPFNs have comparable performance to fully Bayesian methods like ICM with MCMC (NUTS) sampling.

require the maximum number of tasks to be specified during train-time, and they will not generalize to larger number of tasks. Furthermore, the model's representation of a task is fixed and is not influenced by the data points associated with that task.

To address these limitations, we propose a novel scalable attention mechanism for PFNs that effectively leverages the natural hierarchical structure of multi-task data, as shown in Figure A.3. Our approach applies hierarchical attention [34] to the multi-task regression setting, and we interleave intra-task and inter-task blocks. Our architecture introduces learnable "[Task]" tokens that summarize task-specific properties. The intra-task transformer blocks learn relationships between data points within each task while updating both point embeddings and the "[Task]" token summary, requiring $O(D^2)$ compute per task. The inter-task encoders then learn relationships between tasks by attending only to the summary "[Task]" embedding with $O(T^2)$ complexity. This hierarchical design reduces the overall attention complexity from the naive global setting of $O(D^2T^2)$ to $O(D^2T+T^2)$, enabling significantly longer contexts while still allowing for every data point to influence others.

Our hierarchical attention directly addresses many of the limitations of other task encoders. First, our attention mechanism naturally handles inputs of varying lengths, allowing the model to generalize to any number of tasks. Furthermore, our approach enables the model to dynamically learn data-dependent task representations, so tasks with similar patterns can develop similar representations, allowing the model to better capture the potentially complex relationships between tasks. We include further details in Appendix C.

3 Advantages of MTPFNs

In this section, we provide explicit demonstrations of the strengths of MTPFNs for multi-task learning. We compare against baselines: (1) ICM [14], a joint method which trains a multi-task GP on the combined target and auxiliary data; (2) ScaML [31], an ensemble method that fits individual GPs to each auxiliary task; and (3) a single-task GP which only uses the target task and ignores the auxiliary tasks. See Appendix D for additional details about the empirical evaluation and further results.

MTPFNs are robust to negative transfer In Appendix D.2, we evaluate the performance of multi-task models in settings with varying proportions of unrelated tasks. When only 1 of the 3 auxiliary tasks is unrelated, we find that all of the multi-task methods perform similarly. However, as we increase the number of unrelated tasks to 2 out of 3, we find that the MTPFNs trained on data with a higher proportion of corrupted tasks outperform the ICM, which is more sensitive to negative transfer. When we increase the number of unrelated auxiliary tasks 3, we find that the MTPFN trained with $p = 0.2$ is comparable to the single-task GP, which is the underlying DGP for this problem.

MTPFNs efficiently model inter-task relationships We design a regression problem in Figure 2 to highlight the importance of joint modeling. In this setting, all of the data points across all source tasks are drawn from the same function, and this function is highly correlated with the target task. We see that ensemble methods such as ScaML, which do not model the joint interactions between auxiliary tasks, are unable to effectively use the relevant information. However, traditional joint modeling methods like ICM are unable to scale to a large number of tasks and data points. In Appendix D.3, we benchmark the runtimes of multi-task models as we increase the number of tasks and the number

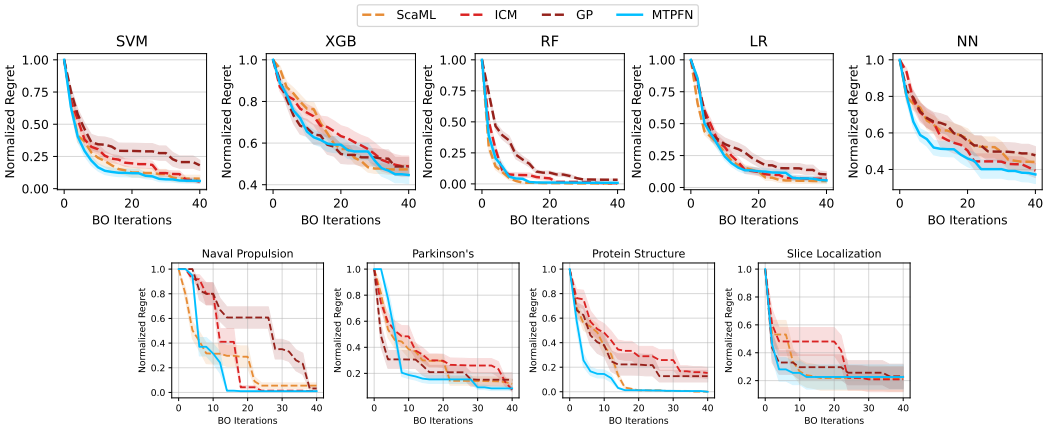


Figure 3: **MTPFNs are competitive across diverse real-world benchmarks.** Each plot shows the normalized regret for Bayesian optimization loop that was initialized with 3 auxiliary tasks, 20 observations from auxiliary task, and 5 observations from the target task.

of data points per task, and we find that this problem quickly becomes unmanageable for ICM. In contrast, MTPFNs are able to scale to large amounts of data while jointly modeling all interactions.

MTPFNs quickly perform fully Bayesian inference Müller et al. [22] demonstrate that PFNs perform fully Bayesian inference by implicitly learning the posterior predictive distribution that marginalizes over all possible samples from the prior which are consistent with the observed data. This ability extends to the multi-task setting: in Figure 2 (Right), we initialize the problem with 20 samples from three auxiliary tasks and 2 samples from the target task. MTPFNs are comparable to the fully Bayesian inference using MCMC sampling through NUTS; furthermore, the MTPFN is able to make predictions using one forward pass of the model in approximately 0.5 seconds on average, while NUTS takes orders of magnitudes longer at 352 seconds per iteration. We showcase further demonstrations of the importance of fully Bayesian inference in Appendix D.4.

MTPFNs can leverage domain data When making predictions with PFNs, domain data can be incorporated through fine-tuning or in-context learning. Fine-tuning updates model parameters to adapt to specific domains, enabling specialization but at high computational cost and risk of overfitting that hurts generalization. In Appendix D.5, we demonstrate these drawbacks on HPOBench's Logistic Regression dataset [8], which contains 25 training tasks with 4 held-out evaluation tasks. We find that fine-tuning improves target domain performance but severely hurts generalization on other domains like SVM and GP tasks due to overfitting, whereas MTPFNs and in-context learning achieve comparable target performance while maintaining strong generalization across all datasets.

MTPFNs are effective across real-world optimization benchmarks We compare the effectiveness of the methods on a set of hyperparameter optimization problems [8] for machine learning model described in Appendix D.6. Following Tighineanu et al. [31], we consider the hyperparameter optimization for five types of models and we share the results of our benchmark in the top panel of Figure 3. We also consider tabular FC-Net benchmarks: Slice Localization, Protein Structure, Naval Propulsion, and Parkinson's Telemonitoring. For each benchmark, we set the benchmark to be the target task, and we use the three other tabular dataset as the auxiliary tasks. MTPFNs are competitive across all of the settings. Specifically, we find that in instances where the meta-tasks contain helpful information (ScaML and ICM outperform GP), the MTPFNs are also able to effectively utilize this data. Furthermore, in cases like XGB where there is negative transfer for the ICM model, we find that MTPFNs are more robust and perform similarly to the standard single-task GP.

4 Discussion

In this work, we present MTPFNs, a scalable and robust surrogate model for Bayesian optimization. Our results highlight the effectiveness of leveraging domain data through in-context learning. MTPFNs are able to successfully capture the complex relationships between the information sources and thus can leverage auxiliary information without expensive model-fitting or fine-tuning procedures.

References

[1] Adriaensen, S., Rakotoarison, H., Müller, S., and Hutter, F. Efficient bayesian learning curve extrapolation using prior-data fitted networks, 2023. URL https://arxiv.org/abs/2310.20447.

[2] Balandat, M., Karrer, B., Jiang, D., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. Botorch: A framework for efficient monte-carlo bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.

[3] Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[4] Bonilla, E. V., Chai, K., and Williams, C. Multi-task gaussian process prediction. In Platt, J., Koller, D., Singer, Y., and Roweis, S. (eds.), *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper_files/paper/2007/file/66368270ffd51418ec58bd793f2d9b1b-Paper.pdf.

[5] Chalkidis, I., Dai, X., Fergadiotis, M., Malakasiotis, P., and Elliott, D. An exploration of hierarchical attention transformers for efficient long document classification. *arXiv preprint arXiv:2210.05529*, 2022.

[6] Chen, Y., Song, X., Lee, C., Wang, Z., Zhang, R., Dohan, D., Kawakami, K., Kochanski, G., Doucet, A., Ranzato, M., et al. Towards learning universal hyperparameter optimizers with transformers. *Advances in Neural Information Processing Systems*, 35:32053–32068, 2022.

[7] Dai, Z., Chen, Y., Yu, H., Low, B. K. H., and Jaillet, P. On provably robust meta-bayesian optimization. In *Uncertainty in Artificial Intelligence*, pp. 475–485. PMLR, 2022.

[8] Eggensperger, K., Müller, P., Mallik, N., Feurer, M., Sass, R., Klein, A., Awad, N., Lindauer, M., and Hutter, F. HPOBench: A collection of reproducible multi-fidelity benchmark problems for HPO. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=1k4rJYEwda-.

[9] Fan, Z., Han, X., and Wang, Z. Hyperbo+: Pre-training a universal prior for bayesian optimization with hierarchical gaussian processes. *arXiv preprint arXiv:2212.10538*, 2022.

[10] Feurer, M., Letham, B., and Bakshy, E. Scalable meta-learning for bayesian optimization using ranking-weighted gaussian process ensembles. In *AutoML Workshop at ICML*, volume 7, pp. 5, 2018.

[11] Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q., and Wilson, A. G. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.

[12] Garnett, R. *Bayesian Optimization*. Cambridge University Press, 2023.

[13] Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J., and Sculley, D. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1487–1495, 2017.

[14] Goovaerts, P. *Geostatistics for natural resources evaluation*, volume 483. Oxford University Press, 1997.

[15] 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.

[16] Hvarfner, C., Hellsten, E. O., and Nardi, L. Vanilla Bayesian optimization performs great in high dimensions. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 20793–20817. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/hvarfner24a.html.

[17] Jones, D. R., Schonlau, M., and Welch, W. J. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

[18] Joy, T. T., Rana, S., Gupta, S., and Venkatesh, S. A flexible transfer learning framework for bayesian optimization with convergence guarantee. *Expert Systems with Applications*, 115: 656–672, 2019.

[19] Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

[20] Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[21] Liao, X., Li, Q., Yang, X., Zhang, W., and Li, W. Multiobjective optimization for crash safety design of vehicles using stepwise regression model. *Structural and Multidisciplinary Optimization*, 35:561–569, 06 2008. doi: 10.1007/s00158-007-0163-x.

[22] Müller, S., Hollmann, N., Arango, S. P., Grabocka, J., and Hutter, F. Transformers can do bayesian inference. *arXiv preprint arXiv:2112.10510*, 2021.

[23] Müller, S., Feurer, M., Hollmann, N., and Hutter, F. Pfns4bo: In-context learning for bayesian optimization. In *International Conference on Machine Learning*, pp. 25444–25470. PMLR, 2023.

[24] Nguyen, T. and Grover, A. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling. In *International Conference on Machine Learning*, pp. 16569–16594. PMLR, 2022.

[25] Nguyen, T., Zhang, Q., Yang, B., Lee, C., Bornschein, J., Miao, Y., Perel, S., Chen, Y., and Song, X. Predicting from strings: Language model embeddings for bayesian optimization, 2024. URL `https://arxiv.org/abs/2410.10190`.

[26] Poloczek, M., Wang, J., and Frazier, P. Multi-information source optimization. *Advances in neural information processing systems*, 30, 2017.

[27] Rasmussen, C. E. *Gaussian Processes in Machine Learning*, pp. 63–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[28] Shields, B. J., Stevens, J., Li, J., Parasram, M., Damani, F., Alvarado, J. I. M., Janey, J. M., Adams, R. P., and Doyle, A. G. Bayesian reaction optimization as a tool for chemical synthesis. *Nature*, 590(7844):89–96, 2021.

[29] Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959, 2012.

[30] Swersky, K., Snoek, J., and Adams, R. P. Multi-task bayesian optimization. *Advances in neural information processing systems*, 26, 2013.

[31] Tighineanu, P., Grossberger, L., Baireuther, P., Skubch, K., Falkner, S., Vinogradska, J., and Berkenkamp, F. Scalable meta-learning with gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pp. 1981–1989. PMLR, 2024.

[32] Wang, Z., Dahl, G. E., Swersky, K., Lee, C., Nado, Z., Gilmer, J., Snoek, J., and Ghahramani, Z. Pre-trained gaussian processes for bayesian optimization. *Journal of Machine Learning Research*, 25(212):1–83, 2024.

[33] Wistuba, M., Schilling, N., and Schmidt-Thieme, L. Scalable gaussian process-based transfer surrogates for hyperparameter optimization. *Machine Learning*, 107(1):43–78, 2018.

[34] Wu, C., Wu, F., Qi, T., and Huang, Y. Hi-transformer: Hierarchical interactive transformer for efficient and effective long document modeling. *arXiv preprint arXiv:2106.01040*, 2021.

[35] Yogatama, D. and Mann, G. Efficient transfer learning method for automatic hyperparameter tuning. In *Artificial intelligence and statistics*, pp. 1077–1085. PMLR, 2014.

[36] Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

[37] Zhuang, B., Liu, J., Pan, Z., He, H., Weng, Y., and Shen, C. A survey on efficient training of transformers. *arXiv preprint arXiv:2302.01107*, 2023.

A Background and Related Work

A.1 Bayesian Optimization

Bayesian optimization (BO) [12] is a sample-efficient black-box optimization method. Given a function f and a compact search space $\mathcal{X} \subset \mathbb{R}^D$, BO aims to find the global maximum[1] $x^* = \arg\max_{x \in \mathcal{X}} f(x)$ by iteratively selecting points to evaluate, conditional on observations made previously. Typically, the observations are corrupted by noise $y = f(x) + \epsilon(x)$ where $\epsilon(x)$ is a noise process. At each step t, a probabilistic model (often a Gaussian process [27]) parameterized by θ is fit to the data collected so far $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^t$. An acquisition function $a(x; \theta)$ (e.g. expected improvement [17]), which balances exploration and exploitation, then uses the model's posterior distribution $p(f|\mathcal{D}_t, \theta)$ to determine the next point to evaluate $x_{t+1} = \arg\max_x a(x; \theta)$. Finally, we evaluate the function for the selected point x_{t+1}, add the new observation to the set of function evaluations $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{(x_{t+1}, y_{t+1})\}$, and proceed to the next iteration. The surrogate model is the key determinant of how successful BO will be, and therefore improvements in the model typically lead to improvements in performance [16].

In the multi-task setting, we have access to auxiliary data from functions which may be similar to the black-box function we wish to optimize. Formally, we aim to find the global optimum of a target task (denoted by task ID 0) $x^* = \arg\max_{x \in \mathcal{X}} f_0(x)$ while having access to evaluations from auxiliary tasks $\{f_k\}_{k=1}^K$, which we assumed are defined over the same search space. Compared to single-task BO, each observation consists of an additional task index k, and the dataset is $\mathcal{D}_t = \{(x_i, y_i, k_i)\}_{i=1}^{n_t}$. The acquisition function $a(x; \theta)$ is then used to select the next point for the target task 0. Successful multi-task BO requires a surrogate model that is able to correctly infer the relationship between the tasks and use this information effectively.

A.2 Multi-Task Surrogate Models

One approach to model multi-task data for BO is to jointly model the full collection of target and auxiliary data using a single GP with a multi-task kernel [4, 30, 35, 26, 18]. We will refer to these models as multi-task GPs (MTGPs). While these joint modeling approaches are effective in the low-data regime, they become computationally infeasible as we scale the number of tasks and data points due to their cubic complexity in the total number of data evaluations.

The intrinsic coregionalization model ICM, 14 is a common choice of MTGP due to its simplicity and was proposed in Swersky et al. [30] for multi-task BO. The ICM models the functions with a kernel that decomposes into two components

$$k((x, t), (x', t')) = k_{\text{inputs}}(x, x') \cdot k_{\text{tasks}}(t, t'),$$

where k_{inputs} is a kernel (often RBF or Matérn) that represents the covariance between inputs and k_{tasks} captures the covariance between tasks. Because this model assumes a single shared latent function across all tasks, it can efficiently transfer knowledge between similar tasks; however, the ICM model may perform poorly when this assumption does not hold and the tasks have distinct characteristics, e.g., when they should be modelled with different lengthscales.

There are other multi-task models which require weaker assumptions. For example, the linear model of coregionalization LMC, 14 is a generalization of the ICM, which uses multiple latent functions that are linearly combined for each task, instead of assuming a single latent function for all tasks. However, although this model is more flexible than the ICM, it has increased computational complexity and may lead to overfitting.

Alternative approaches which focus on scaling to large multi-task datasets have also been proposed. Many methods fit separate GPs to each auxiliary task and ensemble their predictions to inform the target task [13, 10, 33, 7]. Although these approaches are scalable, they are not able to jointly capture information across the related tasks and instead rely on heuristics to determine the relevance of each GP. Other methods use the auxiliary data to learn a better prior over GP hyperparameters for the target task [32, 9]; however, these methods are unable to utilize the specific information within each task. Tighineanu et al. [31] propose a scalable joint modeling approach between the target task and auxiliary tasks; however, this does not model the correlations between the auxiliary tasks, thereby not

[1] Or equivalently the global minimum. Without loss of generality we consider maximization.

taking advantage of the entire dataset. In contrast, we propose a scalable method that jointly models the full interaction between all data points and tasks.

A.3 Bayesian Optimization with Transformers

For single-task Bayesian optimization, there has been a growing interest in using neural-network based approaches. OptFormer [6] leverages a transformer directly trained on data collected from BO loops. In this setup the model does not only model y, like in classical BO, but directly predicts proposals for the next x. This method requires access to large amounts of domain data during training, which may not be possible in many settings.

Transformer neural processes (TNP) [24] and prior fitted networks (PFNs) [22] are transformers trained to approximate the posterior predictive distribution given a data generating function (prior). These models approximate posterior predictive distribution for a prior specified over a hypothesis space $\mathcal{H}$ where each hypothesis $h \in \mathcal{H}$ defines a relationship between inputs x and outputs y. This work builds off of the PFN framework due to PFN's strong empirical performance in Bayesian optimization tasks [23] and prediction for tabular data [15].

A PFN, denoted by f_θ, takes as input a dataset $\mathcal{D}$ and test point x_{test} and outputs a distribution over the target variable $p(y_{\text{test}}|x_{\text{test}}, \mathcal{D})$. To train f_θ to approximate the posterior predictive distribution, we repeatedly sample datasets by first sampling a hypothesis $h \sim p(h)$ which defines a datasets' input-output relationship, and then sampling a dataset $\mathcal{D} \sim p(\mathcal{D}|h)$. The PFN parameters θ are optimized by minimizing the negative log-likelihood on held-out test examples across datasets, expressed as $\mathcal{L}_{\text{NLL}} = \mathbb{E}_{\mathcal{D}\sim p(\mathcal{D}|h)}[-\log f_\theta(y_{\text{test}}|x_{\text{test}}, \mathcal{D}_{\text{train}})]$, where $\mathcal{D}$ is split into $\mathcal{D}_{\text{train}} \cup \{(x_{\text{test}}, y_{\text{test}})\}$.

While TNPs and PFNs have successfully applied to Bayesian optimization in the single-task setting [23, 25], there has been no prior work which explores the use of in-context transfer of related tasks to accelerate optimization. While the PFN's pre-training can already be interpreted as meta-learning, we take it a step further by training them to do Bayesian inference over several related tasks in-context; in this sense, the multi-task PFN acts as a *meta-meta-learning* model.

A.4 Long Contexts

Because of the increased number of tasks and data-points required for multi-task Bayesian optimization, the underlying architecture needs to support significantly longer context windows compared to the single-task setting. Various approaches have been proposed to extend the attention mechanisms in transformers to longer contexts, such as sparse attention [3, 36], hierarchical attention [34, 5], and others [19, 20]. See Zhuang et al. [37] for a survey of efficient methods.

B Data Generation Processes for PFNs

MTPFNs are flexible and have the capacity to incorporate various data generation processes during training. In this section, we explore the impacts of various data generation processes for MTPFNs, each designed to capture distinct inductive biases which improve model performance across different types of tasks.

B.1 Robust Isotropic Full-Rank ICM

Algorithm A.1 Data Generation Using a Robust Isotropic Full-Rank ICM

Require: Sequence length n, number of tasks T, unrelated task probability p

▷ **Input Sampling and Task Assignment**

Sample inputs $\{x_i\}_{i=1}^n \sim \text{Uniform}([0,1]^d)$
Sample task proportions $\pi \sim \text{Dirichlet}(\alpha)$
for $i = 1$ to n **do**
    Sample task ID $t_i \sim \text{Categorical}(\pi)$
end for

▷ **Isotropic ICM Covariance Structure**

Sample task covariance matrix $K_T \sim \text{LKJ}(\eta = 1)$
Sample input lengthscale $\ell \sim \text{Gamma}(3, 6)$
Define input covariance K_X on $\{x_i\}_{i=1}^n$ as RBF kernel with lengthscale ℓ
Compute full ICM kernel $K = K_T \otimes K_X$
Sample $y \sim \mathcal{N}(0, K)$

▷ **Sampling Unrelated Tasks**

for each source task j **do**
    With probability p:
        Sample new lengthscale $\ell_j \sim \text{Gamma}(3, 6)$
        Define RBF kernel $K_X^{(j)}$ on $\{x_i : t_i = j\}$ using ℓ_j
        Resample $y^{(j)} \sim \mathcal{N}(0, K_X^{(j)})$
end for

In the main text, we train PFNs with the data generation process described in Algorithm A.1: we sample datapoints across tasks from a full-rank isotropic ICM model. This approach assumes that all the input dimensions share identical lengthscales; this assumption imposes a strong prior on the relationship between tasks, and enables effective information transfer across related tasks when the assumption is met. This data generation process enables us to learn information across related tasks, since the full-rank isotropic model makes strong assumptions.

To improve our model's robustness to negative transfer, we also incorporate an additional hyperparameter $p \in [0, 1]$ which dictates the relatedness of the tasks during training. Specifically, p is the probability that any given source task is drawn independently from the target task, and thus may have completely different behaviors and lengthscales. Our data generation procedure enables the model to see a diverse group of datasets which consist of a mix of related and unrelated source tasks.

This p hyperparameter plays a crucial part in the robustness of the model against negative transfer: because the model is able to see many examples of unrelated tasks during training, it becomes more robust to seeing unrelated tasks during inference time and is less likely to be negatively impacted from irrelevant information.

We first introduce inter-task relationships by sampling from an ICM MTGP, where the ICM's assumption of a shared lengthscale across tasks enables strong transfer when the tasks are related. Specifically, we sample an inter-task covariance matrix from an LKJ prior with a concentration of 1.0, which provides us with a diverse set of relationships between tasks, and we sample the shared RBF kernel lengthscale from a Gamma (3, 6) prior following the default lengthscale prior in BoTorch v1.11 [2]. To prevent negative transfer, our DGP explicitly encodes the belief that each source task may be irrelevant to the target task by introducing a probability $p \in [0, 1]$ that the task is instead modeled independently using a separate RBF GP with its own lengthscale. In the following sections, we present results under a simple and transparent DGP, but different priors and more sophisticated DGPs can easily be accommodated within the PFN framework. See Appendix B for additional discussions.

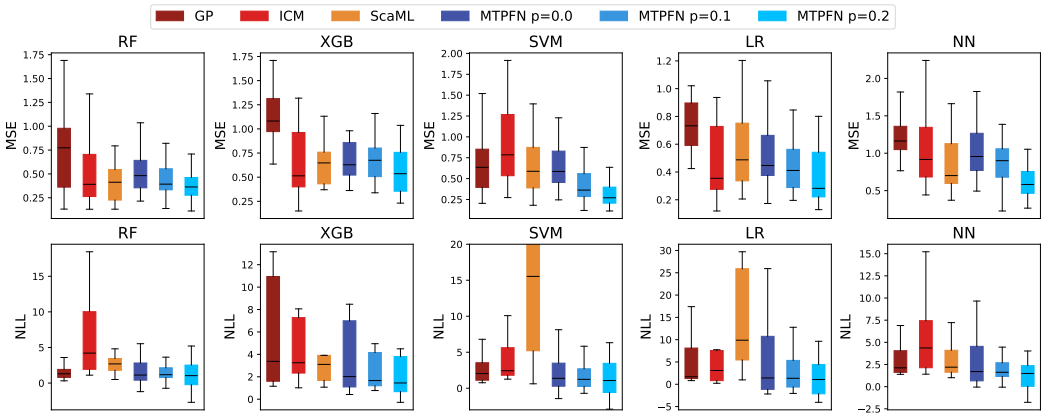


Figure A.1: As we increase p (the probability that each source task is unrelated to the target task during data generation), the model becomes more robust to negative transfer and achieves better performance on real-world benchmarks. We visualize the model's predictive performance on the HPOBench dataset, where we sample 5 data points from the target task and 20 data points each from three source tasks. We plot the average MSE and NLL on holdout data from the target task across 25 trials.

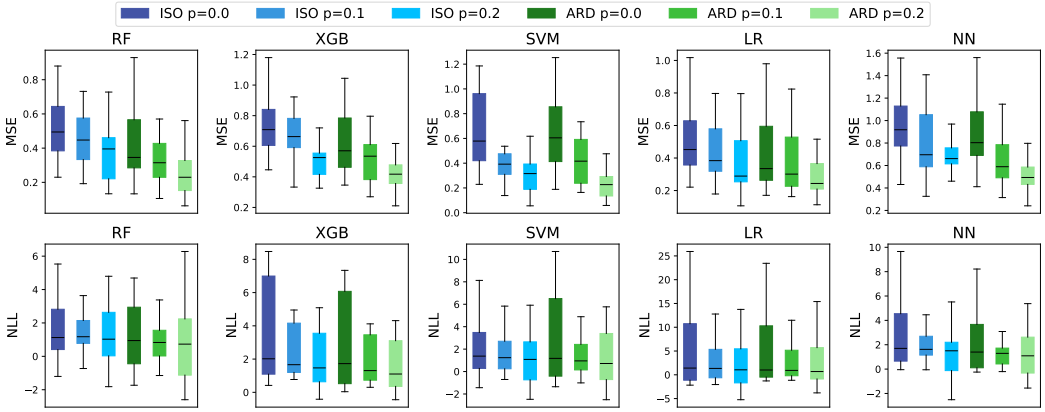


Figure A.2: MTPFNs trained with the ARD data generation process tend to outperform MTPFNs trained with the isotropic process (ISO) and achieve lower MSE and NLLs on HPOBench problems.

In Figure A.1, we study the impact of p on the model's ability to accurately predict the empirical data from HPOBench. Specifically, for each model type (SVM, LR, XGB, NN, and RF), we randomly sample one task to be the target task, and we sample 3 auxiliary tasks from the metadata. The target task is randomly initialized with 5 samples, and we also sample 20 points for each of the auxiliary tasks. We measure the mean squared error (MSE) and the negative log-likelihood (NLL) of each surrogate model on heldout examples from the target task, and we repeat this procedure 25 times and plot the average MSE and NLL for each trial.

We find that increasing p, which increases the diversity of the data that the model sees during training, leads to improved model performance on real-world benchmarks. We see that the MTPFN trained with $p = 0.2$ consistently outperforms other MTPFNs trained with lower values of p, and this MTPFN also outperforms baselines such as the standard ICM model, which assumes that all tasks share the same lengthscale.

B.2 Full-Rank ICM with Automatic Relevance Determination

We can also relax the assumption that all of the input dimensions share the same lengthscale, and instead sample datapoints from an ICM model with Automatic Relevance Determination (ARD), where we assume that each input dimensions has an independent lengthscale. This enables the PFNs

Algorithm A.2 Data Generation Using a Robust ARD Full-Rank ICM

Require: Sequence length n, number of tasks T, unrelated task probability p

▷ **Input Sampling and Task Assignment**

Sample inputs $\{x_i\}_{i=1}^n \sim \text{Uniform}([0,1]^d)$

Sample task proportions $\pi \sim \text{Dirichlet}(\alpha)$

for $i = 1$ to n **do**

  Sample task ID $t_i \sim \text{Categorical}(\pi)$

end for

▷ **ARD ICM Covariance Structure**

Sample task covariance matrix $K_T \sim \text{LKJ}(\eta = 1)$

Sample independent input lengthscales: $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_d) \sim \text{Gamma}(3,6)^d$

Define input covariance K_X on $\{x_i\}_{i=1}^n$ as an RBF kernel with ARD lengthscales $\boldsymbol{\ell}$

Compute full ICM kernel $K = K_T \otimes K_X$

Sample $y \sim \mathcal{N}(0, K)$

▷ **Sampling Unrelated Tasks**

for each source task j **do**

  With probability p:

    Sample new lengthscale $\ell_j \sim \text{Gamma}(3,6)$

    Define RBF kernel $K_X^{(j)}$ on $\{x_i : t_i = j\}$ using ℓ_j

    Resample $y^{(j)} \sim \mathcal{N}(0, K_X^{(j)})$

end for

to have more flexibility and fit more complex problems; however, this weaker assumption may reduce the model's ability to effective transfer information compared to the isotropic settings. We describe this data generation process in Algorithm A.2 and highlight the differences from the isotropic data generation in green.

In Figure A.2, we compare the performance of the MTPFN trained with isotropic lengthscales (ISO) to the performance of the MTPFNs trained with the ARD lengthscales. This experiment follows an identical setup to Figure A.1, where we sample 5 points from a target task and 20 points each from 3 source tasks, and evaluate the MTPFNs on held-out data from the target task. We plot the experiments across 25 trials.

We find that the improved flexibility of the ARD lengthscale generally enables the model to have better performance on the testing data, with the ARD outperforming ISO across many datasets. However, in some settings such as SVM, we find that the model performance of the isotropic ICM and the ARD ICM are comparable. This similar performance may be because the assumption of the shared lengthscale across input dimension is satisfied in this setting, so the additional flexibility of the ARD is unnecessary.

C Hierarchical Attention Details

For multi-task regression problems, the encoding of the task can influence how the model integrates information from the various sources and impact its ability to learn helpful relationships and differentiate between tasks with irrelevant characteristics. In this section, we describe two straight-forward methods to represent the task information, and we discuss their implementations and also describe their limitations for our problem setting.

Categorical Feature For a data point $(\mathbf{x}_i, y_i)$, its associated task can be represented as a categorical feature $t_i \in \{1, 2, \ldots, T\}$, where T is the number of distinct tasks. We can represent t_i using a one-hot encoding $\mathbf{1}_{t_i} \in \{0, 1\}^T$, and the final input is formed by concatenating the original feature vector with the one-hot encoding $\mathbf{x}'_i = [\mathbf{x}_i; \mathbf{1}_{t_i}]$. Although simple, this approach does not provide the model with information related to the task itself. Furthermore, the maximum number of tasks must be specified at train-time, and the model is also unable to generalize to a larger number of tasks at test-time since the categorical feature has a fixed number of dimensions.

Task Embedding Rather than directly using the one-hot encoding of the task, we can use a task encoder to map the task t_i to a continuous embedding vector $\mathbf{e}_{t_i} \in \mathbb{R}^d$, where d is the embedding dimension of the model. This embedding is jointly learned with the model parameters, allowing the task representation to adapt to task-specific characteristics. The original input $(\mathbf{x}_i, y_i)$ is first transformed into a feature $\mathbf{z}_i = \phi(\mathbf{x}_i, y_i)$, and then this feature is combined with the task embedding $\mathbf{z}'_i = \mathbf{z}_i + \mathbf{e}_{t_i}$. This approach integrates the task information directly into the feature space; however, the representation for the task is still learned independently of the information within each task, and the model remains unable to generalize to more tasks at test-time.

C.1 Hierarchical Attention

To address these limitations, we propose a novel scalable attention mechanism for PFNs that effectively leverages the natural hierarchical structure of multi-task data, as shown in Figure A.3. Our approach applies hierarchical attention [34] to the multi-task regression setting and uses specialized transformer blocks to separately model intra-task and inter-task relationships.

For intra-task encoding, we introduce a learnable "[Task]" token to each task that summarizes task-specific properties. The intra-task transformer blocks are responsible for learning the relationships of the data points within each task, By performing attention over these points, the intra-task block updates the embeddings for each data point and also updates the "[Task]" token with a summary embedding for the task, requiring $O(D^2)$ total compute per task. Then, the inter-task encoders, responsible for learning the relationship between tasks, attend to these summary "[Task]" embeddings, with $O(T^2)$ complexity. This hierarchical design reduces the overall attention complexity from the naive global setting of $O(D^2T^2)$ to $O(D^2T + T^2)$, enabling significantly longer contexts while still allowing for every data point to influence others. We interleave the intra-task and inter-task blocks in our architecture, although Chalkidis et al. [5] show that other topologies may also be effective

Our hierarchical attention directly addresses many of the limitations of other task encoders. First, our attention mechanism naturally handles inputs of varying lengths, allowing the model to generalize to any number of tasks. This flexibility ensures that even if the model encounters more tasks at test time than it did during training, it can still meaningfully integrate new task representations. Furthermore, our approach enables the model to dynamically learn task representations which depend on the data from the task, and its representation of each task evolves through the many layers of attention. This enables tasks with similar patterns to develop similar representations, allowing the model to better capture the potentially complex relationships between tasks.

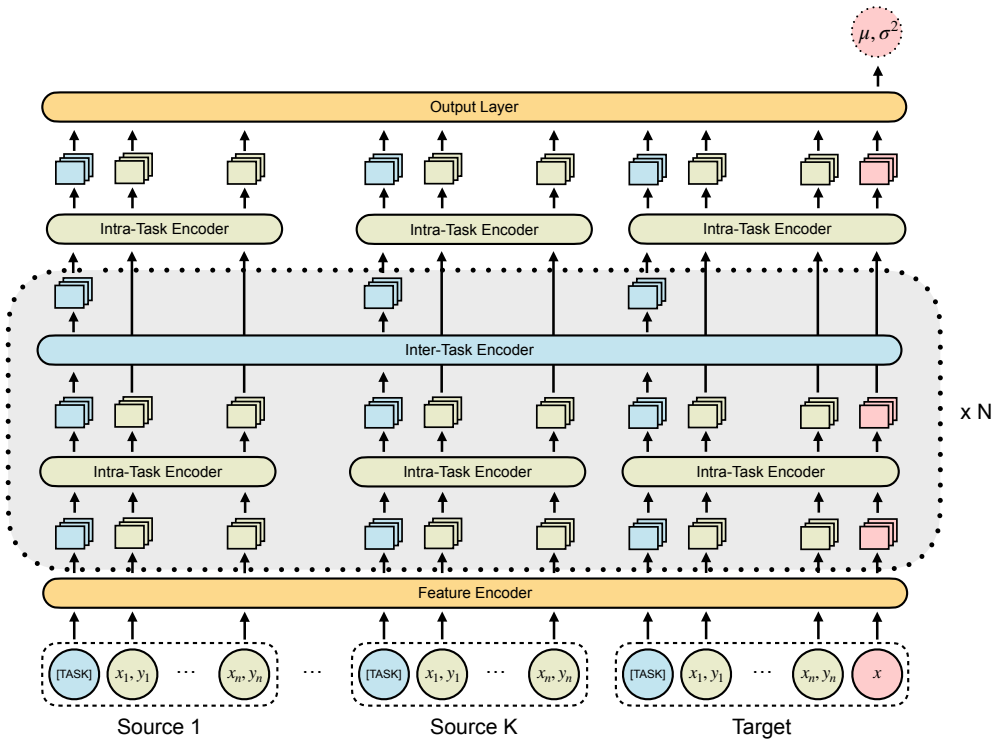


Figure A.3: MTPFNs use hierarchical attention and jointly model data across information sources.

D Empirical Results and Details

D.1 Setup Details

For our empirical results, we use a transformer backbone with 23 attention layers, where twelve intra-task attention layers are interwoven between eleven inter-task layers. Each attention layer has 4 attention heads with a hidden size of 512. The model is trained on approximately 50 million synthetically generated datasets as described in Section 2.1, with a batch size of 16 and AdamW with a learning rate of 1e-4 and cosine annealing.

We compare our method, MTPFN, to several baselines: (1) ICM [14], a joint method which trains a multi-task GP on the combined target and auxiliary data; (2) ScaML [31], an ensemble method that fits individual GPs to each auxiliary task; and (3) a single-task GP which only uses the target task and ignores the auxiliary tasks. Our Bayesian optimization results were implemented using BoTorch [2] and GPyTorch [11], and we will provide access to our code upon acceptance.

D.2 Robust to Negative Transfer

We explore the impact of p, the probability that any task is unrelated to target task from the data generation procedure. We train three MTPFNs with varying values of p to understand its effect on the robustness of the model to negative transfer. See Appendix B for further ablations.

For our test settings, we use an ICM to generate 5 different multi-task datasets with 3 input dimensions, each with 2 samples from the target task and 20 samples from each of the 3 auxiliary tasks. The auxiliary tasks have varying amounts of correlations with the target task. We then perform 5 runs of Bayesian inference for each multi-task dataset, and plot the mean and standard error of the mean. We find that MTPFNs trained with higher proportions of unrelated tasks are more robust to negative transfer and outperform other baselines.

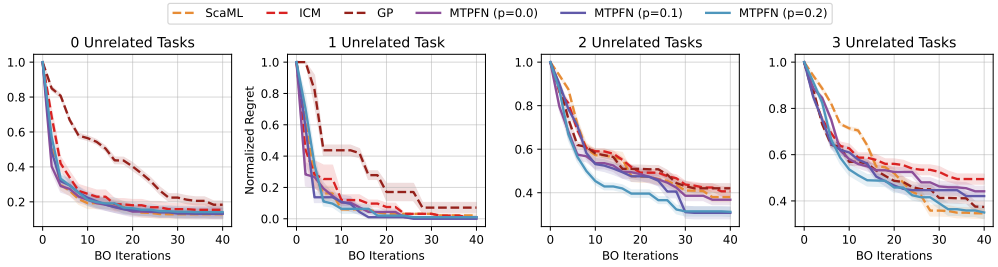


Figure A.4: **MTPFNs are robust to negative transfer from unrelated tasks**. We evaluate BO across four multi-task settings, where the target task is related to $\{0, ..., 3\}$ out of 3 auxiliary tasks. We compare the performance of MTPFNs trained with different p, where p represents the probability that an auxiliary task was drawn independently from the target task during training. As we increase the number of unrelated tasks during evaluation, the MTPFNs which were exposed to unrelated tasks during training ($p > 0$) outperform the ICM model, which suffers from negative transfer. We plot the mean and standard error of the mean over 5 trials.

D.3 Efficient Modeling of Inter-Task Relationships

MTPFNs are able to do Bayesian inference with a single forward pass. Furthermore, our proposed hierarchical attention mechanism enables the MTPFN to scale in $O(TD^2 + T^2)$, where D is the number of data points per task and T is the number of tasks. We compare the runtime of MTPFNs to joint-modeling methods such as ICM and ensemble-based methods such as ScaML in Figure A.5. We see that MTPFNs are able to perform inference on an order of magnitude more data points and tasks compared to traditional GP methods.

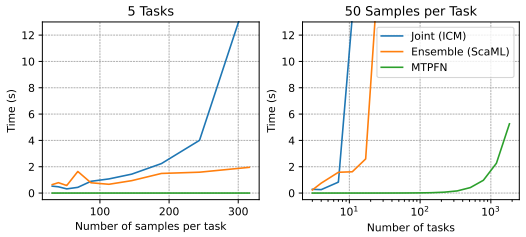


Figure A.5: MTPFNs are significantly faster than alternative GP-based methods.

D.4 Fully Bayesian Inference

When trained on a data-generation process that draws samples from a multi-task GP with an ICM kernel, we see in Figure A.6 that the MTPFN and the MTGP (ICM kernel with MAP estimation) have comparable behavior across varying levels of correlations. Furthermore, in low-data settings demonstrated by Figure A.7, we find that the MTPFN outperforms the MTGP because it considers the uncertainty over the task covariance matrix. This demonstrates that fully Bayesian inference may be preferable to MAP estimation.

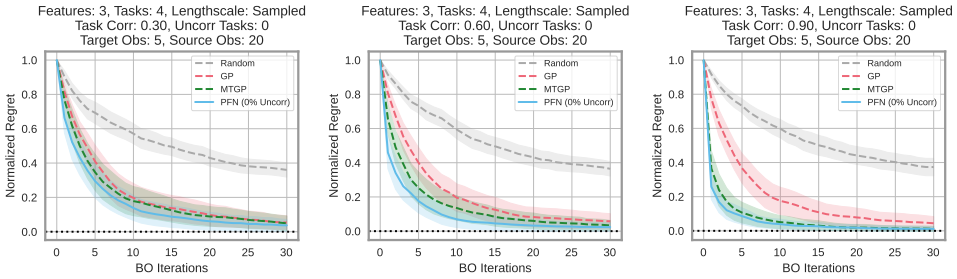


Figure A.6: ICM PFNs are comparable to MTGPs across varying levels of correlations between tasks.

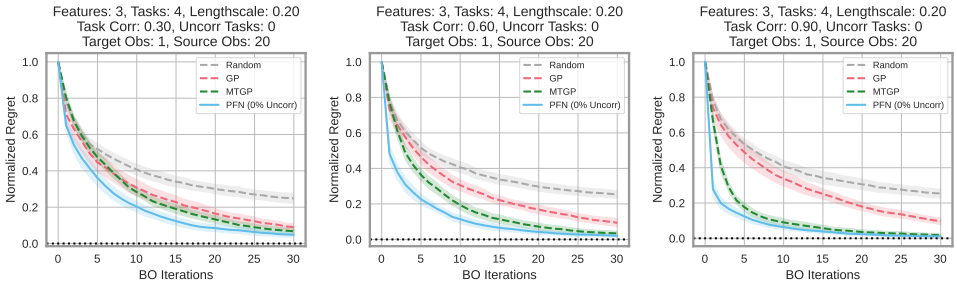


Figure A.7: In low-data settings, ICM PFNs, which approximate fully Bayesian inference, outperform MTGPs with MAP estimation. ICM PFNs are comparable to MTGPs across varying levels of correlations between tasks.

D.5 Leverage Domain Data

In Figure A.8, we demonstrate the benefits of in-context learning compared to fine-tuning on the HPOBench dataset [8] for Logistic Regression (LR), which contains 25 tasks with 4 held out for evaluation. We compare three approaches: (1) the "Original Single-Task", a general-purpose single-task model trained on Gaussian process draws with an RBF kernel; (2) the "Fine-Tuned Single-Task", the same base model after fine-tuning on data from the 25 LR training tasks; and (3) "MTPFNs", our method that uses only the 4 hold-out tasks in-context during inference, without any fine-tuning on the 25 training tasks. We evaluate all models by measuring negative log likelihood on the LR evaluation set as well as on other domains (SVM hyperparameter optimization and GP draws) to assess generalization.

The results clearly demonstrate the benefits of MTPFNs over fine-tuned approaches. While fine-tuning does improve performance on the target LR domain, it comes at a severe cost to generalization: as we fine-tune on more LR samples, the performance on SVM and GP domains deteriorates significantly due to overfitting. In contrast, MTPFNs, which use in-context learning, achieve comparable performance on the target domain while maintaining strong generalization across all evaluated datasets. This approach is also computationally efficient, requiring no parameter updates.

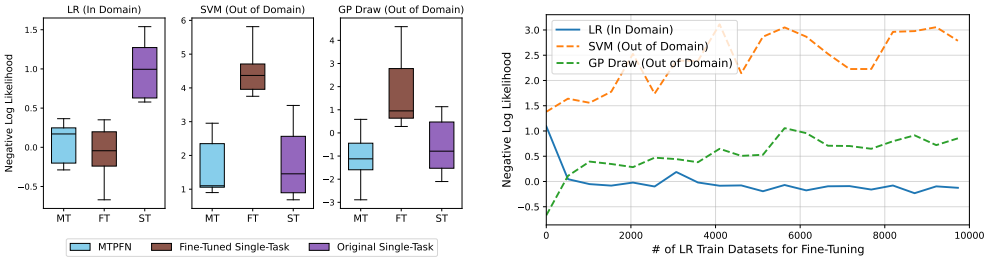


Figure A.8: **MTPFNs, which use domain data through in-context learning, match the performance of Fine-Tuned PFNs on in-domain data while generalizing better to other domains.** (**Left**): MTPFNs have comparable NLLs to Fine-Tuned PFNs on in-domain data (LR) and outperform Fine-Tuned PFNs on other domains (SVM and GP Draws). (**Right**): As we fine-tune on more in-domain data, the NLL for Fine-Tuned PFNs significantly worsens for other domains.

Original Single-Task PFN Details The data generation process for the single-task PFN randomly samples inputs x from the unit cube, and then samples the corresponding outputs y by drawing a sample from a GP with an RBF kernel with a lengthscale sampled from Gamma(3, 6). For this experiment, we use a fixed feature size of 2. We train an 8-layer standard transformer (not hierarchical attention) with an embedding size of 256 on this data generation process for 4 million sampled datasets, with a batch size of 16, and AdamW with a learning rate of 1e-4 and cosine annealing.

Fine-Tuned Single-Task PFN Details To fine-tune on the LR dataset, we develop a subsampling data-generation procedure: On the 20 training tasks, we subsample within one task to get 50 x, y. We uniformly select some number of them to be used as ICL training, and the remaining to be used as the test. We fine-tune the base model described above with a batch size of 16, and AdamW with a learning rate of 1e-4 and cosine annealing.

D.6 Real-World Optimization Problems

We consider the hyperparameter optimization for five types of models: support vector machines (SVM), logistic regression (LR), XGBoost (XGB), neural networks (NN), and random forest (RF). For each setting, we randomly sample one task to be the target function, and we sample 3 auxiliary tasks from the meta-data. We randomly sample 5 points from the target task and 20 points from each of the auxiliary tasks to use as the initialization for Bayesian inference. We measure the normalized regret $(f^* - f_{\text{best}})/(f^* - f_0)$ where f^* is the optimal value, f_{best} is the best value so far, and f_0 is the initial value. We run 100 replicates, each with a different combination of target task and auxiliary task initializations, and we plot the mean and one standard error.

We share the results of our benchmark in the top panel of Figure 3, and MTPFNs are competitive across all of the model types. Specifically, we find that in instances where the meta-tasks contain

helpful information (ScaML and ICM outperform GP), the MTPFNs are also able to effectively utilize this data. Furthermore, in cases like XGB where there is negative transfer for the ICM model, we find that MTPFNs are more robust and perform similarly to the standard single-task GP.

We also consider tabular FC-Net benchmarks from: Slice Localization, Protein Structure, Naval Propulsion, and Parkinson's Telemonitoring. For each benchmark, we set the benchmark to be the target task, and we use the three other tabular dataset as the auxiliary tasks. For instance, the results for Slice Localization use Protein Structure, Naval Propulsion, and Parkinson's Telemonitoring as auxiliary data sources. We initialize our Bayesian optimization problem with a random sample of 5 points from the target task and 20 points from each auxiliary task. We report the average normalized regret over 20 trials in the bottom panel of Figure 3. MTPFNs work well on these tabular datasets, often outperforming the other baselines.