NeurIPS 2023 Workshop on Adaptive Experimental Design and Active Learning in the Real World

Data-driven Prior Learning for Bayesian Optimisation

Sigrid Passano Hellan S.P.HELLAN@ED.AC.UK
Christopher G. Lucas C.LUCAS@ED.AC.UK
Nigel H. Goddard NIGEL.GODDARD@ED.AC.UK

School of Informatics
University of Edinburgh, UK

Abstract

Transfer learning for Bayesian optimisation has generally assumed a strong similarity between optimisation tasks, with at least a subset having similar optimal inputs. This assumption can reduce computational costs, but it is violated in a wide range of optimisation problems where transfer learning may nonetheless be useful. We replace this assumption with a weaker one only requiring the shape of the optimisation landscape to be similar, and analyse the recent method *Prior Learning for Bayesian Optimisation* — PLeBO — in this setting. By learning priors for the hyperparameters of the Gaussian process surrogate model we can better approximate the underlying function, especially for few function evaluations. We validate the learned priors and compare to a breadth of transfer learning approaches, using synthetic data and a recent air pollution optimisation problem as benchmarks. We show that PLeBO and prior transfer find good inputs in fewer evaluations.

Keywords: Bayesian optimisation, Transfer learning, Priors, Markov chain Monte Carlo

1. Introduction

Our goal in Bayesian optimisation (BO) is to find high-performing input values using few function evaluations. A powerful approach is to transfer knowledge from previous optimisation tasks. For example, when optimising many similar problems we can choose to disregard input ranges that consistently yield suboptimal outputs. Within the BO framework there are many ways of transferring knowledge (Bai et al., 2023), for instance through a shared model (Swersky et al., 2013), through the initial input values which we evaluate (Feurer et al., 2015), or through limiting the search space (Perrone et al., 2019). In this paper we develop the idea of prior transfer further, and introduce an extension to a recent prior transfer method (Hellan et al., 2022), which we call Prior Learning for BO (PLeBO).

Direct transfer: Most transfer learning for Bayesian optimisation rests on the assumption that inputs that are optimal for (some of) the tuning tasks will also be near-optimal for the new test task. For instance, a common technique is to evaluate a few good input values from previous tasks on a new task before doing Bayesian optimisation.

Prior transfer: An alternative to direct transfer is to assume only that the optimisation tasks have similar response surfaces, i.e. the shapes of their optimisation landscapes. A simple example is to assume that all optimisation tasks can be modelled using a single shared set of Gaussian process (GP) hyperparameter (HP) values, as is done in Wang et al.

(2022). Another approach is to learn a feature transformation which is applied to the input to the GP before the covariance kernel. This is done in Few-Shot BO, an example of deep kernel learning where a neural network is used for this feature transformation (Wistuba and Grabocka, 2021). Hand-crafted HP priors can also be seen as prior transfer: with PLeBO we automate this process. Müller et al. (2023) also learn HP priors from related data, but pick the best from a randomly sampled set instead of learning a posterior distribution.

Fig. 1 illustrates the difference between direct and prior transfer in two idealised settings. In the top row the optima for the tuning and test tasks coincide, and only direct transfer is able to perform useful transfer. In the bottom row the optima are far apart but the shapes are the same. In this setting prior transfer works perfectly while direct transfer does not work at all. For direct transfer we use a strategy where we evaluate the best point from the tuning data. We illustrate prior transfer with a method where we assume the shapes are identical — in practice we just assume the GP hyperparameters are similar.

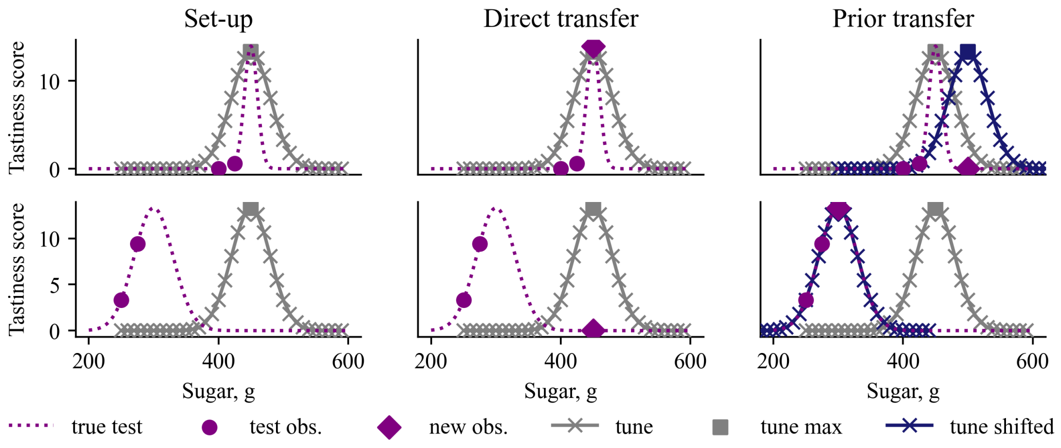

Figure 1: Comparison of direct and prior transfer. In the top row the past and new tasks share optima but have different shapes. In the bottom row they instead have the same shape. Direct transfer works well for shared optima, prior transfer for shared shapes.

Most work on Bayesian optimisation and transfer learning has been applied to hyperparameter optimisation for machine learning (HPO), e.g. Feurer et al. (2015); Perrone et al. (2019); Wistuba and Grabocka (2021). Extending the range of applications motivates the development of new methods, as assumptions that make sense for one application do not necessarily make sense for a different application. For HPO we can expect similar learning rates to work well for the same model trained on different data sets. But in the air pollution context we do not expect the same coordinates in different cities to be the most polluted — e.g. three kilometres north of the city centre. Therefore, we expect prior transfer to work well in situations unsuited to direct transfer, such as air pollution monitoring.

An earlier version of PLeBO was presented as part of Hellan et al. (2022). Here we improve on the method and compare it to other transfer learning approaches. We also provide code for a new and simpler implementation, and analyse the quality of the resulting priors. We show the advantage of prior transfer over direct transfer on both synthetic and real-world data, the latter being air pollution measurements.

2. Bayesian optimisation

Bayesian optimisation (Garnett, 2023) is a blackbox optimisation technique, which utilises a surrogate model, typically a Gaussian process, to model the underlying optimisation task and come up with informed choices of input values. It is particularly suited to problems with a complex unknown structure, where we do not know the gradients, and which are expensive to evaluate. Gaussian processes (Rasmussen and Williams, 2006) are probabilistic machine learning models which are popular in the regime of small data. They are defined by their choice of mean and covariance functions. The former is often set to zero (De Ath et al., 2020), while the latter comes with hyperparameters that need to be set. For instance, the RBF kernel $k_{RBF}(\boldsymbol{\tau}) = \sigma_r^2 \exp\left(-\frac{\boldsymbol{\tau}^T\boldsymbol{\tau}}{2l^2}\right)$ has two hyperparameters: the lengthscale l and the signal variance σ_r^2. The former defines how quickly the function changes, the latter how large its absolute values are. We also need an acquisition function, which uses the surrogate model to determine the next sampling location. We use expected improvement (EI, Jones et al. (1998)) unless otherwise noted.

The problems we want to optimise in Bayesian optimisation are of the form $\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}} f(\boldsymbol{x})$. To fit the hyperparameters of the Gaussian process we use the marginal log likelihood. A simple approach is to use gradient descent, possibly including a prior term on the hyperparameters. Alternatively, we can approximate the posterior distribution of the hyperparameters, for instance using Markov chain Monte Carlo (MCMC) (Lalchand and Rasmussen, 2020). Then we take the uncertainty of our hyperparameters into account in our model. With transfer learning we assume access to historical evaluations of other tasks, changing the problem to $\boldsymbol{x}_j^* = \arg\max_{\boldsymbol{x}} f_j(\boldsymbol{x}) \mid \{\{\boldsymbol{x}_{n,i}, y_{n,i}\}_{i=1}^{N_n}\}_{n=1}^{N}$ where n indexes tasks, i indexes samples and N_n is the number of sample evaluations for task n.

3. PLeBO

The PLeBO method is an example of prior transfer, intended to work even with a small amount of tuning data. We assume a hierarchical structure where each optimisation task has its own set of GP hyperparameters $\boldsymbol{\theta}$, and these hyperparameters come from a shared distribution $\boldsymbol{\eta}$. The method has two parts: in the preprocessing step, we use MCMC to learn $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ given observations $\boldsymbol{\mathcal{D}}$, and then sample new values of $\boldsymbol{\theta}$ based on $\boldsymbol{\eta}$ to generate a set of candidate hyperparameters. In the optimisation phase we use importance weighting to fit these candidates to the observed test data $\boldsymbol{\mathcal{D}}_j$. This structure is illustrated in Fig. 2.

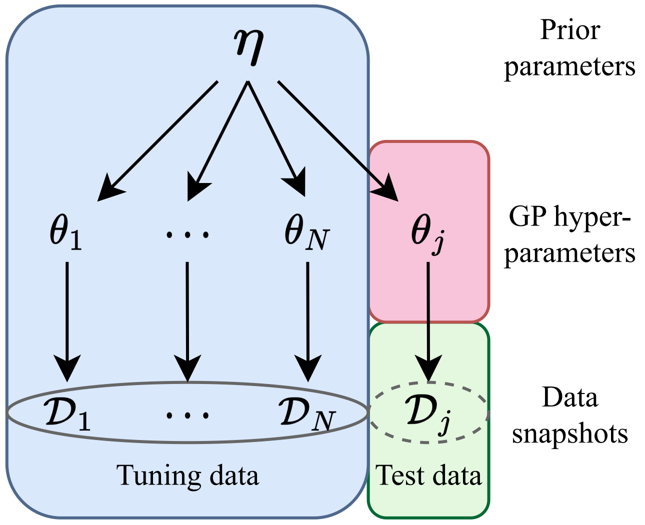

Figure 2: Hierarchical structure and inference in PLeBO.

Formally, the structure is $p(\boldsymbol{\eta}, \boldsymbol{\theta}, \boldsymbol{\mathcal{D}}) = p(\boldsymbol{\eta})p(\boldsymbol{\theta}|\boldsymbol{\eta})p(\boldsymbol{\mathcal{D}}|\boldsymbol{\theta}) = p(\boldsymbol{\eta}) \prod_n p(\boldsymbol{\theta}_n|\boldsymbol{\eta})p(\boldsymbol{\mathcal{D}}_n|\boldsymbol{\theta}_n)$. The modelling distributions can be chosen separately for each hyperparameter, we use gamma distributions $p(\boldsymbol{\theta}|\boldsymbol{\eta}) = \Gamma(\boldsymbol{\theta}; \boldsymbol{\eta})$. That means we need to learn N sets of $\boldsymbol{\theta}_n$ and $2k$ parameters of $\boldsymbol{\eta}$, where k is the dimensionality of $\boldsymbol{\theta}_n$ and N is the number of tuning tasks. For simplicity, we restrict ourselves to 2 HPs in $\boldsymbol{\theta}_n$, so learn 4 parameters of $\boldsymbol{\eta}$. We also need to choose the number H of candidate samples; we use 200 to get a good approximation to taking the expectation given $\boldsymbol{\eta}$, but future work should explore reducing H.

Preprocessing: We learn distributions for the surrogate model hyperparameters, using Markov chain Monte Carlo to collect samples from the distributions of $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$. We replace Hellan et al.'s (2022) custom-built implementation with a new one (`https://github.com/sighellan/plebo`) based on NumPyro (Phan et al., 2019) and using the NUTS (Hoffman and Gelman, 2014) sampler. It is more stable and makes modelling changes, e.g. changing $p(\boldsymbol{\theta}|\boldsymbol{\eta})$, much simpler. As the sampling chains do not always find samples with non-zero likelihood we run multiple chains in parallel before a simple filtering postprocessing step.

Optimising: We use the candidate samples within the optimisation loop to calculate the acquisition function, $a_i(\boldsymbol{x}) \approx \frac{1}{W} \sum_{h=1}^{H} w_h \, a_i^{\theta}(\boldsymbol{x}, \boldsymbol{\theta}_h^{\text{cand}})$ where $w_h = p(\boldsymbol{\mathcal{D}}_j^{:i}|\boldsymbol{\theta}_h^{\text{cand}})$, $W = \sum_{h=1}^{H} w_h$ and $a_i^{\theta}(\cdot)$ is the base acquisition function, for which we use EI. At each optimisation step we calculate the likelihood of every hyperparameter candidate, and use it to weight the corresponding base acquisition function. More details are in Appendix A.2.

4. Experiments

We consider both a synthetic benchmark and real-world air pollution data (Copernicus, 2018). To simplify analysis we use the RBF kernel and limit ourselves to two hyperparameters: the lengthscale and the signal variance. Our base metric is the best observed value at each iteration i, normalised by the max value of the task, $y_{\max}^{(j)}$: $r_i^{(j)} = \max \boldsymbol{Y}_{:N_{\text{start}}+i}^{(j)} / y_{\max}^{(j)}$ where $\boldsymbol{Y}_{:N_{\text{start}}+i}^{(j)}$ are the observations and N_{start} is the number of starting evaluations.

We compare PLeBO to a collection of baselines with no, direct and prior transfer:

- No transfer
  - **RandomSearch:** Randomly select the inputs at each iteration.
  - **BoTorch:** Default settings from the BoTorch library (Balandat et al., 2020).
  - **EI:** BO without transfer using expected improvement (Jones et al., 1998).
  - **UCB:** BO without transfer using upper confidence bound (Srinivas et al., 2012).
- Direct transfer
  - **DirectTrans:** Shared GP for past and new tasks. Cap at 100 past evaluations.
  - **Initial:** First evaluate the best point from each previous task, then do **EI**.
- Prior transfer
  - **PLeBO:** The new prior learning method detailed in Section 3.
  - **TruePLeBO:** Using EI with the true hyperparameters (only for synthetic data).
  - **Gamma:** Use the mean prior $\boldsymbol{\eta}$ from PLeBO in gradient descent for the HPs.
  - **Shared:** A single set of HPs learned from the tuning tasks (Wang et al., 2022).

The **synthetic benchmark** allows us to evaluate the extracted priors. We use the same hierarchical model as in PLeBO, see Fig. 2. We use GPs with RBF kernels and a known low level of noise. We define gamma distributions on the two hyperparameters: the lengthscale $l \sim \Gamma(5, 0.01)$, and the signal variance $\sigma_r^2 \sim \Gamma(2, 2)$. We use the shape, scale parameterisation of $\Gamma(\cdot)$. We generate a set of 10 tuning optimisation tasks and 100 test tasks. The tuning tasks have 20 evaluations each, and we start the test tasks with ten known evaluations. The tasks are discretised to match the other benchmarks. See Fig. 3.

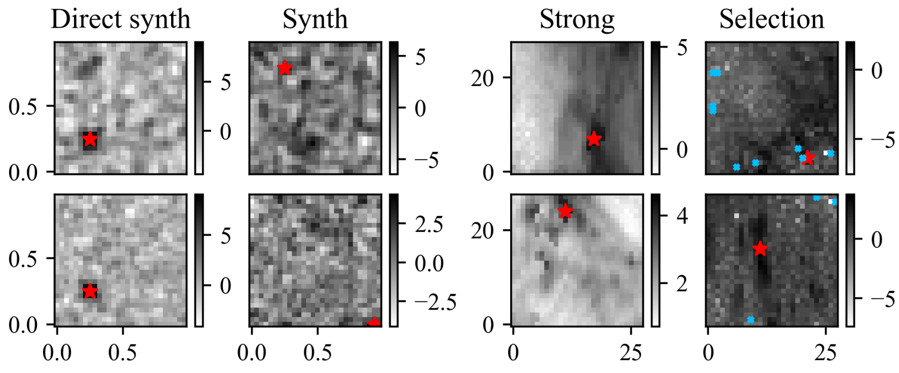

Figure 3: Example optimisation tasks. The left column shows the standard assumption that optima are near each other. Stars indicate optima and blue crosses missing values.

We also use two **air pollution benchmarks**: the satellite optimisation tasks used in Hellan et al. (2020). Each task comes from a snapshot of NO_2 air pollution levels taken from the Sentinel-5P satellite of the Copernicus programme (Copernicus, 2018). The two benchmarks come from grouping the snapshots based on the maximum pollution present. The Strong benchmark consists of 50 test tasks and 10 tuning tasks of similar problems. The Selection benchmark consists of 100 test tasks and 10 tuning tasks with maximal pollution levels at the middle of those of the test tasks. It therefore checks whether prior transfer still works if the task distributions change between the tuning and test data. The data has been preprocessed by log transform and standardisation, see Hellan et al. (2020).

5. Results and Discussion

Fig. 4 compares the performance of PLeBO to the other methods. We see that the transfer methods generally find good input values in fewer iterations. EI only does slightly worse than PLeBO. This highlights the difficulty of prior transfer, as we still need to explore the search space. We also see that prior transfer requires fewer iterations than direct transfer. The differences are smaller among the prior transfer methods. PLeBO is better than Shared and Gamma on the synthetic and Strong benchmarks, but for the Selection benchmark Shared works best. That TruePLeBO works best on the synthetic data supports our intuition that the optimisation can be sped up by better surrogate model HP estimates.

To validate PLeBO we also analyse the inferred priors. In Appendix A.4 we compare the inferred and true values for $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$. While the true and estimated priors are not identical they are similar, and an exact reconstruction could not be expected from only 10 tuning tasks. We also analysed the fits on the individual tasks: the true and inferred values had very similar likelihoods, and sometimes the inferred values had higher likelihood.

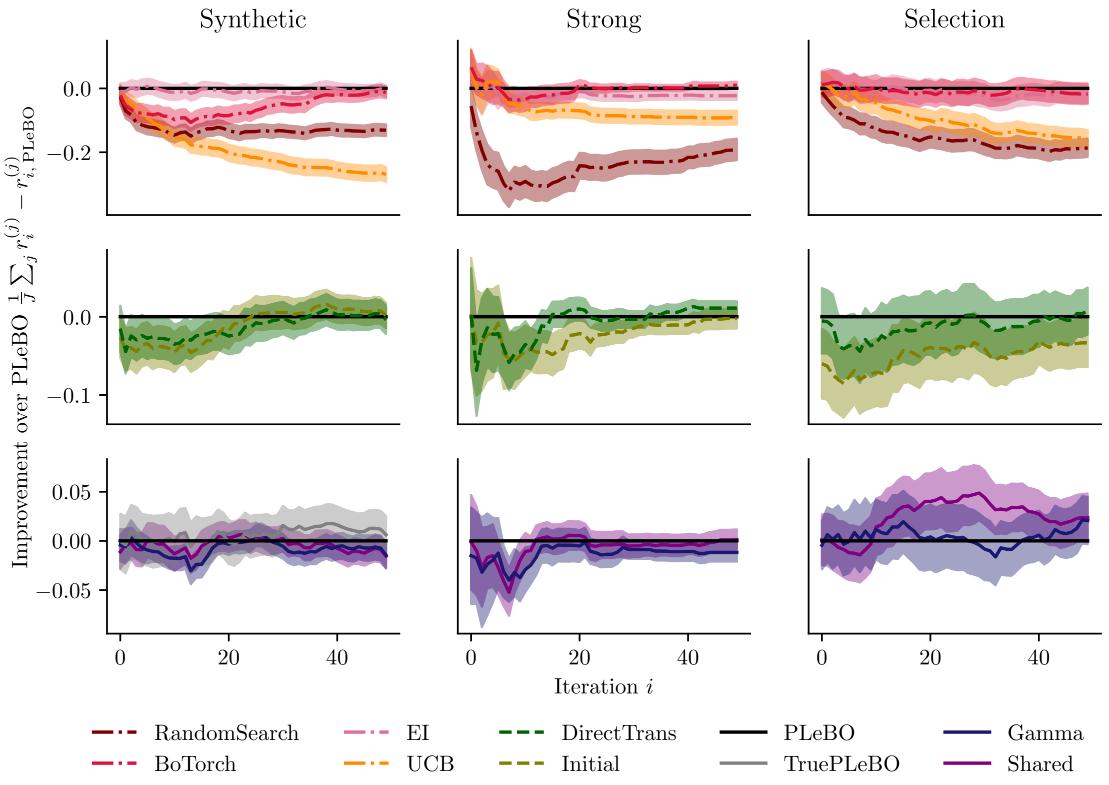

Figure 4: Performance compared to PLeBO at same iteration (above zero is improvement), mean ± one standard error. J is the number of test tasks. The top row compares PLeBO to no transfer, the middle to direct transfer and the bottom row to prior transfer methods.

PLeBO is computationally more expensive than the baselines. Its preprocessing step has a variable runtime due to the NUTS sampler. The runtime was 9 minutes for the synthetic, and 2.6–3.4 hours for the air pollution benchmarks on a standard desktop. The preprocessing is only done once, so a higher computational expense can be tolerated. In the optimisation step we replace gradient descent with calculating the acquisition function for each of H candidate HP sets. This scales as $\mathrm{O}(H(i + N_{\text{start}})^3)$. On average, PLeBO takes about 6.7 seconds per optimisation step, compared to 0.13 seconds for EI. For very expensive optimisation tasks this is still only a fraction of the total cost, e.g. when installing an air pollution sensor. Runtimes for all methods are given in Appendix A.3. PLeBO's runtime can be adjusted through H based on the available budget and task evaluation speed.

We have presented an improved prior transfer method, PLeBO, and evaluated it on a real-world benchmark for air pollution monitoring. We showed that the generated priors align with the tuning data, and can be exploited on new tasks. The approach is modular, and is not tied to a specific acquisition function. While more general, prior transfer is also weaker than direct transfer as we do not learn about the optima directly: we showed some improvement over EI, but not a massive difference. Future work should attempt to exploit the correct HP choices more, and evaluate PLeBO on a wider range of benchmarks.

Acknowledgments

We thank Tiffany Vlaar and Linus Ericsson for comments on the paper, and G-Research for generous financial support towards conference attendance.

This work was supported by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh.

References

Tianyi Bai, Yang Li, Yu Shen, Xinyi Zhang, Wentao Zhang, and Bin Cui. Transfer learning for Bayesian optimization: A survey. *arXiv preprint arXiv:2302.05927*, 2023.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. In *Advances in Neural Information Processing Systems 33*, 2020.

Copernicus. Sentinel-5P TROPOMI Level 2 Nitrogen Dioxide total column products. Version 01, 2018. URL `https://sentinels.copernicus.eu/web/sentinel/data-products/-/asset_publisher/fp37fc19FN8F/content/sentinel-5-precursor-level-2-nitrogen-dioxide`. Processed by ESA. DOI: 10.5270/S5P-s4ljg54.

George De Ath, Jonathan E Fieldsend, and Richard M Everson. What do you mean? The role of the mean function in bayesian optimisation. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, pages 1623–1631, 2020.

Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Initializing Bayesian hyperparameter optimization via meta-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

Roman Garnett. *Bayesian optimization*. Cambridge University Press, 2023.

Sigrid Passano Hellan, Christopher G. Lucas, and Nigel H. Goddard. Optimising placement of pollution sensors in windy environments. *NeurIPS 2020 Workshop on AI for Earth Sciences*, 2020.

Sigrid Passano Hellan, Christopher G. Lucas, and Nigel H. Goddard. Bayesian optimisation for active monitoring of air pollution. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(11):11908–11916, 2022.

Matthew D Hoffman and Andrew Gelman. The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1): 1593–1623, 2014.

Donald R Jones, Matthias Schonlau, and William Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

Vidhi Lalchand and Carl Edward Rasmussen. Approximate inference for fully Bayesian Gaussian process regression. *Symposium on Advances in Approximate Bayesian Inference*, pages 1–12, 2020. PMLR.

Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. PFNs4BO: In-context learning for Bayesian optimization. In *International Conference on Machine Learning*. PMLR, 2023.

Valerio Perrone, Huibin Shen, Matthias W Seeger, Cedric Archambeau, and Rodolphe Jenatton. Learning search spaces for Bayesian optimization: Another view of hyper-parameter transfer learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Du Phan, Neeraj Pradhan, and Martin Jankowiak. Composable effects for flexible and accelerated probabilistic programming in NumPyro. *arXiv preprint arXiv:1912.11554*, 2019.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *IEEE Transactions on Information Theory*, 58(5):3250–3265, 2012.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task Bayesian optimization. *Advances in neural information processing systems*, 26, 2013.

Zi Wang, George E Dahl, Kevin Swersky, Chansoo Lee, Zelda Mariet, Zachary Nado, Justin Gilmer, Jasper Snoek, and Zoubin Ghahramani. Pre-training helps Bayesian optimization too. *arXiv preprint arXiv:2207.03084*, 2022.

Martin Wistuba and Josif Grabocka. Few-shot Bayesian optimization with deep kernel surrogates. In *International Conference on Learning Representations*, 2021.

Appendix A. Supplementary material

A.1 Ethics statement

The work uses no personal data and the goal of the application is to reduce the impact of air pollution on human health. The authors have no ethical concerns.

A.2 PLeBO optimisation step

Algorithm 1 gives the procedure for calculating the acquisition function which is followed at each optimisation step. This is also expressed in Eq. (1), which shows how this corresponds to the expected acquisition function when drawing the hyperparameters from their posterior distribution. $\boldsymbol{\mathcal{D}}_{1:N}$ is the set of tuning observations. $\boldsymbol{\mathcal{D}}_j^{:i}$ is the test observations available at iteration i.

Algorithm 1 PLeBO acquisition calculation

```
Given $\boldsymbol{\theta}^{\text{cand}}$, $\boldsymbol{\mathcal{D}}_j^{:i}$, $\boldsymbol{X}$, $a_i^{\theta}(\cdot)$
$\boldsymbol{a}, W \leftarrow \mathbf{0}, 0$
for $h$ in $1, \ldots, H$ do
    $w \leftarrow p(\boldsymbol{\mathcal{D}}_j^{:i} | \boldsymbol{\theta}_h^{\text{cand}})$
    $\boldsymbol{a} \leftarrow \boldsymbol{a} + w\, a_i^{\theta}(\boldsymbol{X}, \boldsymbol{\theta}_h^{\text{cand}})$
    $W \leftarrow W + w$
end for
Return $\boldsymbol{a}/W$
```

$$a_i(\boldsymbol{x}) = \underset{p(\boldsymbol{\theta}_h^{\text{cand}}|\boldsymbol{\mathcal{D}}_{1:N}, \boldsymbol{\mathcal{D}}_j^{:i})}{\mathbb{E}}[a_i^{\theta}(\boldsymbol{x}, \boldsymbol{\theta}_h^{\text{cand}})] \approx \frac{1}{W} \sum_{h=1}^{H} w_h\, a_i^{\theta}(\boldsymbol{x}, \boldsymbol{\theta}_h^{\text{cand}}), \;\; \boldsymbol{\theta}_h^{\text{cand}} \sim p(\boldsymbol{\theta}_h^{\text{cand}} | \boldsymbol{\eta}) \quad (1)$$

A.3 Runtime

Table 1 gives the runtime for the preprocessing step of PLeBO for each benchmark. For the air pollution benchmarks we use 100 evaluations from each of the tuning tasks. Table 2 gives the mean duration of each optimisation step for all the considered methods. We see that PLeBO is much slower than the other methods. But for expensive optimisation tasks, where each step corresponds to e.g. installing a pollution sensor or training a neural network, this cost is negligible. We have capped the number of past evaluations available to DirectTrans at 100, otherwise it would be much slower than the listed times.

The optimisation step of PLeBO scales as $\mathrm{O}(H(i + N_{\text{start}})^3)$, as opposed to $\mathrm{O}(N_{\text{grad}}(i + N_{\text{start}})^3)$ for EI where N_{grad} is the number of gradient steps (Rasmussen and Williams, 2006, p. 19). H is a hyperparameter of the method which we can adjust to balance computational cost with the quality of the posterior estimate of the GP hyperparameters. At one extreme we could set H to 1, which should give a similar runtime to Shared.

Benchmark	Preprocessing time
Synthetic	9 min 11 sec
Strong	2 h 38 min
Selection	3 h 22 min

Table 1: PLeBO preprocessing times.

Method	Synthetic	Strong	Selection
RandomSearch	0.006	0.005	0.005
BoTorch	0.20	0.31	0.27
EI	0.16	0.11	0.11
UCB	0.17	0.11	0.11
DirectTrans	1.35	1.20	1.20
Initial	0.16	0.10	0.11
PLeBO	6.73	6.60	6.89
TruePLeBO	0.05	-	-
Gamma	0.18	0.13	0.12
Shared	0.05	0.04	0.04

Table 2: Mean durations in seconds of an optimisation step.

A.4 Prior quality

Fig. 5 compares the true and inferred values of $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$ for the lengthscale and signal variance. The learned distributions are more peaked than the true ones, but are similar and reasonable approximations given that we only use 10 tuning tasks.

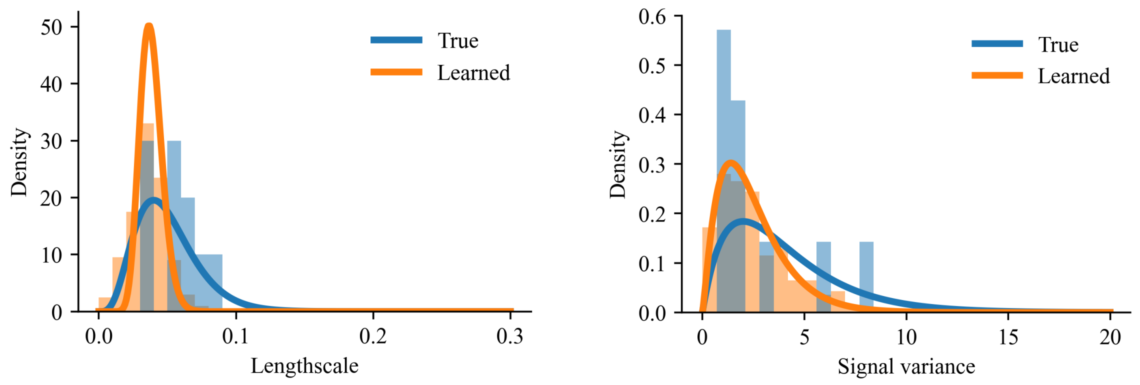

Figure 5: Comparing true and inferred hyperparameter priors for the synthetic benchmark.