Supplementary Information

Equipping data-driven experiment planning for Self-driving Laboratories with semantic memory: case studies of transfer learning in chemical reaction optimization

Riley J. Hickman,[a] Jurgis Ruža,[a] Loïc M. Roch,[a,*] Hermann Tribukait,[a,†] and Alberto García-Durán,[a]

[a] *Atinary Technologies Inc, 1006 Lausanne, VD, Switzerland*
[*] *loic@atinary.com*
[†] *ht@atinary.com*

S.1. NEURAL PROCESS MODEL DETAILS

Neural processes (NPs) [1] are a neural network based architecture which learn approximations of a stochastic process. Like Gaussian processes (GPs), NPs learn to model distributions over functions and are able to estimate uncertainty over their predictions conditioned on some set of context observations. Importantly, NPs can perform inference in a computationally efficient way. Given a trained NP, inference is essentially a forward pass of a neural network, which is an $\mathcal{O}(n+m)$ operation compared to the $\mathcal{O}\left((n+m)^3\right)$ scaling of traditional GPs, with n context and m target data points. Also, NPs inherently transcend functional design restrictions of GPs by inferring a kernel implicitly from data. We are interested in learning implicit kernels which strongly resemble specific concepts in chemistry and materials science, and apply them to related optimization problems.

The NP architecture consists of three models: a deterministic encoder, a latent encoder and a decoder. Each model is a fully connected multi-layer perceptron (MLP). An MLP with n hidden layers is constructed as

$$\phi_1 = \text{act hidden}(\boldsymbol{x} \cdot w_0 + b_0)\,, \tag{5}$$
$$\phi_2 = \text{act hidden}(\phi_1 \cdot w_1 + b_1)\,, \tag{6}$$
$$\vdots$$
$$\phi_n = \text{act hidden}(\phi_{n-1} \cdot w_{n-1} + b_{n-1})\,, \tag{7}$$
$$\phi_{\text{out}} = \text{act out}(\phi_n \cdot w_{\text{out}} + b_{\text{out}}). \tag{8}$$

The computational graph of the latent variable NP is shown in Fig. S1. Context points $\{(\boldsymbol{x}_i, y_i)\}_{i\in C}$ are passed through a deterministic and latent encoder. In the deterministic path, the encoder function produces the representation $\boldsymbol{r}_i$ for each context point. These are then aggregated using a permutation invariant operation (mean) to form the representation $\boldsymbol{r}_C$ which is intended to summarize the encoded inputs.

The latent path produces a similar representation, $\boldsymbol{s}_C$, which is passed through an additional MLP to estimate the parameters of the diagonal multivariate Gaussian distribution $\boldsymbol{z} \sim \mathcal{N}\left(\mu(\text{MLP}(\boldsymbol{r}), I\sigma(\text{MLP}(\boldsymbol{r}))\right)$. Finally, $\boldsymbol{r}_C$, are concatenated with samples from $\boldsymbol{z}$, along with targeted input locations $\boldsymbol{x}_*$ and passed through the decoder which parameterizes the Gaussian predictive posterior distribution $p\left(y_* | \boldsymbol{x}_*, \boldsymbol{r}_C, \boldsymbol{z}\right)$

During training, we use the Adam optimizer [2] to optimize the evidence lower bound (ELBO) to the log predictive likelihood. At each iteration of the training process, we randomly select a training task and randomly select n of its m input-output pairs as target points $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, and select a subset of these points to be context points $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{k}$, where $k < n \leq m$. The loss function is

$$\begin{aligned} \log p\left(y_T | \boldsymbol{x}_T, \boldsymbol{x}_C, y_C\right) &\geq \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{s}_T)}\left[\log p\left(y_T|\boldsymbol{x}_T, \boldsymbol{r}_C, \boldsymbol{z}\right)\right] - \text{KL}\left(q(\boldsymbol{z}|\boldsymbol{s}_T) || q(\boldsymbol{z}|\boldsymbol{s}_C)\right)\,, \\ &\geq \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{s}_T)}\left[\log p\left(y_T|\boldsymbol{x}_T, \boldsymbol{r}_C, \boldsymbol{z}\right)\right] + \text{KL} \log \frac{q(\boldsymbol{z}|\boldsymbol{s}_C)}{q(\boldsymbol{z}|\boldsymbol{s}_T)}\,. \end{aligned} \tag{9}$$

$\boldsymbol{x}_C$ and y_C are used to abbreviate the context inputs and outputs for the training step (same for $\boldsymbol{x}_T$ and y_T for the target points). $\boldsymbol{s}_C$ ($\boldsymbol{s}_T$) is the latent path representation produced using the context (target) points.

A. Attentive neural processes

Although the mean-aggregation step reduces the runtime of the NP, it is well known that it acts as an information bottleneck as taking the mean across all $\boldsymbol{r}_i$ gives equal weight to each context point. This makes it difficult for the

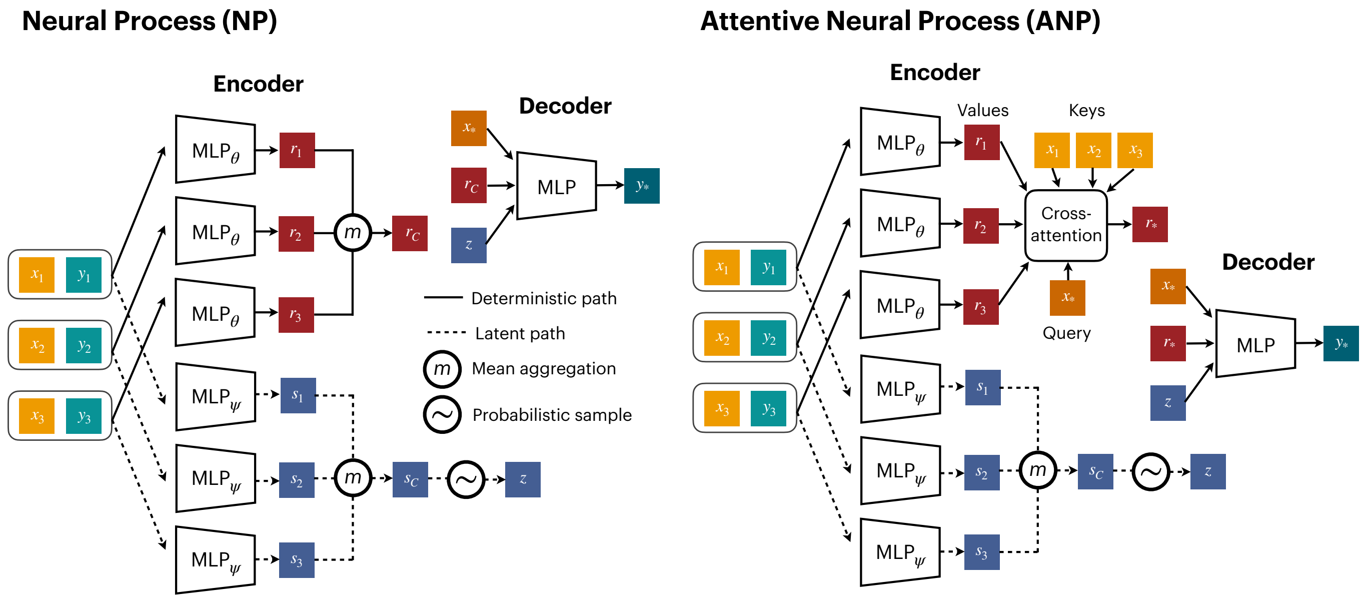

FIG. S1: Schematic depiction of the neural process (left) and attentive neural process (right) model architectures. The key difference between the two models is that in the attentive model, the mean aggregation step is replaced by a cross-attention mechanism which produces a query-specific representation code $\boldsymbol{r}_*$. In this work we do not use a self-attention mechanism in the encoder of the attentive neural process. Instead, we use same vanilla MLP as in the neural process model.

decoder to learn which of the context points provides relevant information about a given target location $\boldsymbol{x}_*$. Attentive neural processes (ANPs) address this bottleneck issue by replacing the mean-aggregation step with a cross-attention mechanism. In this way, the target query location $\boldsymbol{x}_*$ *attends* to the key-value pairs of context points $\{(\boldsymbol{x}_i, y_i)\}_{i\in C}$ and assigns weights w_i to each context pair to give a query specific representation code, $\boldsymbol{r}_C = \sum_i w_i \boldsymbol{r}_i$. The best performing attention mechanism reported in Ref. [3], multihead attention [4], is used. For each head, a learned linear transformation is applied individually to the keys, values and queries, then dot-product attention is applied to give head-specific values. These values are then concatenated and transformed linearly once more to give the final value. For queries Q, keys K, and values V, multihead attention with H heads can be summarized as

$$\text{Multihead}\,(Q, K, V) = (\text{head}_1, \ldots, \text{head}_H)\, W \in \mathbb{R}^{m\times d_v}\,, \tag{10}$$

where $(\cdot, \ldots, \cdot)$ denotes concatenation and

$$\text{head}_i = \text{DotProduct}\left(QW_i^Q, KW_i^K, VW_i^V\right) \in \mathbb{R}^{m\times d_v}\,. \tag{11}$$

$$\text{DotProduct}\,(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \in \mathbb{R}^{m\times d_v}\,. \tag{12}$$

Note that in Ref. [3], the authors also use a self-attention mechanism which replaces the vanilla MLP in the original NP encoder. We choose not to use self-attention in this work since the problems we address are all in the low-data regime.

B. Neural process hyperparameters

S.2. IMPLEMENTATION DETAILS OF RELATED META-LEARNING BAYESIAN OPTIMIZATION STRATEGIES

In this section, details of our implementations of related Bayesian optimization strategies with access to historical source data are given. All strategies are implemented in house using the BoTorch library [5],

Hyperparameter	Encoder	Decoder
hidden layers	3	2
hidden nodes	48	48
hidden activation	ELU	ELU
output activation	linear	linear
learning rate	5e4	
batch size	100	
batch norm momentum	0.98	
batch norm epsilon	0.002	

TABLE S3: Hyperparameters for the NP model used in this work. Hyperparameters were determined manually by assessing the generalizability of hyperparameter settings to a wide range of tasks (*i.e.* with varying input and output dimensions, number of data points, etc.).

A. Details of transfer acquisition function implementation (TAF)

Training on all of the source campaign data at once makes the assumption that each source task is equally important for transfer to the target task. Also, for GP-based models, this may pose computational scaling issues given the $\mathcal{O}\left(n^3\right)$ scaling of GPs with n training points. Transfer acquisition functions (TAFs) therefore propose to train an independent GP model on each of the N source campaigns, including an $N+1^{\text{th}}$ model which is iteratively trained on the target campaign observations $\mathcal{D}_T$. TAFs then propose to transfer the information from the source campaigns to the target campaign via the acquisition function,

$$\alpha(\boldsymbol{x}) = \frac{w_{N+1}\mathbb{E}\left[I_{N+1}(\boldsymbol{x})\right] + \sum_{i=1}^{N} w_i I_i(\boldsymbol{x})}{\sum_{i=1}^{N+1} w_i} \,. \tag{13}$$

The first term in the numerator corresponds to the $N+1^{\text{th}}$ GP model trained on $\mathcal{D}_T$, while the summation is over the N source campaigns. Weights w_i determine the influence each campaign has on the final acquisition function. The critical step in TAF strategies is determining the values of each w_i. The predicted improvement for the source tasks is

$$I_i(\boldsymbol{x}) = \max\left(y_i^{\min} - \hat{Y}_i(\boldsymbol{x}), 0\right) \,, \tag{14}$$

where $\hat{Y}(\boldsymbol{x}) \sim \mathcal{N}\left(\mu_i(\boldsymbol{x}), \sigma_i^2(\boldsymbol{x})\right)$ is sampled from the predictive posterior distribution of the i^{th} source campaign GP, and $y_i^{\min}$ is the best value achieved on that source campaign. The TAF is then a weighted average of the expected improvement on the ground-truth target campaign observations $\mathcal{D}_T$ and the predicted improvement on all source campaigns $\{\mathcal{D}_n\}_{n=1}^{N}$, where

$$\mathbb{E}\left[I_{N+1}(\boldsymbol{x})\right] = \mathbb{E}\left[\max\left(y_{\mathcal{D}_T}^{\min} - \hat{Y}(\boldsymbol{x})\right), 0 \mid \mathcal{D}_T\right] \,. \tag{15}$$

We implement the TAF-ME strategy, which is a simple baseline strategy which assumes that all source tasks contribute equally to the final acquisition, *i.e.* $w_i = 1/M + 1 \; \forall \; i$. All GP models use an isotropic Matern5/2 kernel function.

B. Details of rank-weighted Gaussian process ensemble implementation (RGPE)

Feurer *et al.* proposed the rank-weighted Gaussian process ensemble (RGPE) approach to do meta-learning across related optimization tasks [6, 7]. Similar to the TAF approaches, RGPE trains independent GPs on each of the source tasks, and estimates the target task objective as a weighted sum of the target and source models. For M source tasks and an additional target task, the GPs have posterior $f^i(\boldsymbol{x}|\mathcal{D})$ with mean $\mu_i(\boldsymbol{x})$ and variance $\sigma_i^2(\boldsymbol{x})$. The estimate of the target objective is

$$\bar{f}(\boldsymbol{x}|\mathcal{D}) = \sum_{i=1}^{M+1} w_i f^i(\boldsymbol{x}|\mathcal{D}) \,, \tag{16}$$

which, importantly, is also a GP

$$\bar{f}(\boldsymbol{x}|\mathcal{D}) \sim \mathcal{N}\left(\sum_{i=1}^{M+1} w_i \mu_i(\boldsymbol{x}), \sum_{i=1}^{M+1} w_i^2 \sigma_i^2(\boldsymbol{x})\right) . \tag{17}$$

The weights w_i for model i are computed based on the ranking loss between draws from the posterior of model i and the target task observations $\mathcal{D}_{M+1}$. Given $|\mathcal{D}_{M+1}| = n_{M+1} > 1$ target task measurements, the ranking loss is formulated as the number of misranked pairs

$$\mathcal{L}(f^i, \mathcal{D}_{M+1}) = \sum_{k=1}^{n_{M+1}} \sum_{l=1}^{n_{M+1}} \mathbb{1}\left((f(\boldsymbol{x}_k^{M+1}) < f(\boldsymbol{x}_l^{M+1})) \oplus (y_k^{M+1} < t_l^{M+1})\right) . \tag{18}$$

$\mathbb{1}$ is an indicator function, and $\oplus$ is the exclusive-or operation. The ranking-based loss assures that only the location of the optimum is important, as opposed to the actual values of the predictions. The loss for the target model is computed using leave-one-out cross validation, and is given by

$$\mathcal{L}(f^{M+1}, \mathcal{D}_{M+1}) = \sum_{k=1}^{n_{M+1}} \sum_{l=1}^{n_{M+1}} \mathbb{1}\left((f_{-k}^{M+1}(\boldsymbol{x}_k^{M+1}) <_{-k}^{M+1} (\boldsymbol{x}_l^{M+1})) \oplus (y_k^{M+1} < t_l^{M+1})\right) , \tag{19}$$

where f_{-k}^{M+1} denotes a model fit to all target task observations except for the k^{th} point. Now, the weights are assigned to each model according to the probability that it is the ensemble member with the lowest ranking loss. This probability is estimated by boostrapping S samples from the model predictions on the validation set, i.e. $\ell_{i,s} \sim \mathcal{L}(f^i, D_{M+1,s}^{\text{boot}})$. w_i is then computed as

$$w_i = \frac{1}{S} \sum_{i=1}^{S} \left(\frac{\mathbb{1}(i \in \text{argmin}_{i'}, \ell_{i',s})}{\sum_{j=1}^{M+1} \mathbb{1}(j \in \text{argmin}_{i'}, \ell_{i',s}))} \right) . \tag{20}$$

C. Details of Multi-task Gaussian process implementation (MTBO)

Multi-task Gaussian processes for Bayesian optimization (MTBO) were proposed by Swersky *et al.* [8]. This approach defines a covariance function $K((\boldsymbol{x}_k, i), (\boldsymbol{x}_l, j))$ between parameter points $\boldsymbol{x}$ and the task, and uses a single GP for all observations across all tasks, which are then learned jointly. Specifically, this approach is known as the intrinsic model of coregionalization, which transfroms a latent function to produce each output. Formally, the kernel function is

$$K((\boldsymbol{x}_k, i), (\boldsymbol{x}_l, j)) = K_{i,j}\left(i, j\right) \otimes K_x\left(\boldsymbol{x}_k, \boldsymbol{x}_l\right) , \tag{21}$$

where $\otimes$ is the Kronecker product, $K_{i,j}$ measures the relationship between the tasks, and K_x measures the relationship between the inputs. MTBO has a training complexity of $O((M+1)^3 n^3)$, where M is the number of total observations and n is the number of tasks, and requires a expensive hyperparameter tuning routine.

D. Details of the deep kernel transfer implementation (DKT)

Deep kernels are models which combine neural networks with kernels to provide scalable and expressive closed-form expressions for covariance [9, 10]. Given two input parameter instances $\boldsymbol{x}$ and $\boldsymbol{x}'$ and a function f, the kernel $k\left(\boldsymbol{x}, \boldsymbol{x}'\right)$ is a covariance function which describes the joint variability of the outputs as a function of their parameter space locations,

$$k\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \text{cov}\left(f(\boldsymbol{x}), f(\boldsymbol{x}')\right) . \tag{22}$$

In our experiments, we use the Matérn kernel: a stationary kernel which generalizes the squared and absolute exponential kernel. It is parameterized by ν, which we set to 2.5, giving twice differentiable functions. In deep kernel learning, input parameters $\boldsymbol{x}$ are mapped to an embedding vector $\boldsymbol{h}$ using a neural network with learnable weights ϕ, i.e. $\mathcal{F}_\phi(\boldsymbol{x}) \mapsto \boldsymbol{h}$. This embedding is then passed to a kernel

$$k\left(\boldsymbol{x}, \boldsymbol{x}'|\theta, \phi\right) = k'\left(\mathcal{F}_\phi(\boldsymbol{x}), \mathcal{F}_\phi(\boldsymbol{x}')|\theta\right) \,, \tag{23}$$

where the neural network parameters ϕ and kernel parameters θ are jointly learned using a single optimizer. Patacchiola *et al.* introduced *Deep kernel transfer* (DKT), a Bayesian model for the meta-learning/few-shot setting which uses deep kernels. More recently, Witsuba *et al.* extended the DKT method to a few-shot Bayesian optimization framework [11]. Our DKT approach follows closely to that of Ref. [11]. The feature extractor network $\mathcal{F}_\phi$ is constructed using 3 densely connected layers. The initial 2 layers have 48 nodes, and the final has 40 nodes, i.e. $|\boldsymbol{h}| = 40$. The ReLU activation function is used for all hidden layers. The model is trained end-to-end using the Adam optimizer, although we use a learning rate of 0.001 to adjust the neural network parameters, and a learning rate of 0.0001 to adjust the parameters of the kernel.

S.3. TARGET CAMPAIGN ACQUISITION FUNCTIONS

Three target acquisition functions, (α_T in main text) are considered in this work. Gryffin is a general purpose BO framework which is tailored to the needs of chemists and materials scientists [12]. Gryffin extends the ideas of the Phoenics algorithm to categorical and mixed categorical-continuous parameter spaces. Phoenics is a linear-scaling Bayesian optimization for continuous parameter spaces whose surrogate model relies on kernel density estimation [13]. For minimization problems, the acquisition function of Gryffin is defined as

$$\alpha_T(\boldsymbol{x}) = \frac{\sum_{k=1}^n f_k p_k(\boldsymbol{x}) + \lambda p_{\text{uniform}}(\boldsymbol{x})}{\sum_{k=1}^n p_k(\boldsymbol{x}) + p_{\text{uniform}}(\boldsymbol{x})} \,, \tag{24}$$

where $p_k(\boldsymbol{x})$ are the kernels of the kernel regression surrogate model, f_k are the observed objective values, and λ is a user defined sampling parameter which biases the sampling behavior between exploitation and exploration. In all our experiments, we alternate between $\lambda = 1$ and $\lambda = -1$ at each iteration. The acquisition function defined in Eq. 24 is to be minimized.

The *expected improvement* (EI) acquisition function is a commonly used function in the Bayesian optimization literature. Given a black-box function f to be optimized, let f' be the best value of this function observed so far. EI proposes to evaluate f at the parameter point at which we expect to improve on f' the most. The utility of a parameter point $\boldsymbol{x}$ is then $\max\left(0, f' - f(\boldsymbol{x})\right)$, and the EI acquisition function is the expectation value of this utility,

$$\alpha_T(\boldsymbol{x}) = \left(\mu(\boldsymbol{x}) - f' - \xi\right)\Phi\left(Z\right) + \sigma(\boldsymbol{x})\phi\left(Z\right) \,. \tag{25}$$

$$Z = \frac{\mu(\boldsymbol{x}) - f' - \xi}{\sigma_(\boldsymbol{x})} \,. \tag{26}$$

where $\mu(\boldsymbol{x})$ and $\sigma(\boldsymbol{x})$ are respectively the surrogate model's mean and standard deviation. Φ and ϕ are the CDF and PDF of the standard normal distribution. ξ is a tradeoff parameter which determines the amount of exploration during optimization. We set $\xi = 0.01$ in all experiments.

Lastly, the *upper confidence bound* (UCB) acquisition function is

$$\alpha_T(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \beta\sigma(\boldsymbol{x}) \,, \tag{27}$$

where $\mu(\boldsymbol{x})$ and $\sigma(\boldsymbol{x})$ are respectively the surrogate model's Gaussian mean and standard deviation. $\beta > 0$ is a tradeoff parameter that weights the importance of variance based sampling in the acquisition function. We set $\beta = 0.01$ in all experiments.

S.4. ANALYTICAL BENCHMARKS

In all analytical benchmark optimization experiments, the goal of the target campaign is to minimize the *reference* function. To generate source tasks for a continuous, d-dimensional reference function, we sample shift, $\boldsymbol{t} \sim \left[-0.1, 0.1\right]^d$

and scale, $s \sim [0.9, 1.1]$ perturbations which are applied to the reference surface. For d-dimensional categorical functions, we sample the parameters from $\boldsymbol{t} \sim [-0.3, 0.3]^d$ and $s \sim [0.8, 1.2]$. A set of 30 source tasks for each reference function are generated. For 2(3)-dimensional source campaigns we sample 50 (200) points using a Sobol sequence, which are used to train the meta-learning strategies. A similar approach to generating analytical source campaign datasets was employed previously by Volpp *et al.* [14]. Fig. S2 shows several samples for the source campaigns generated for the 2-dimensional *Goldstein-Price* function.

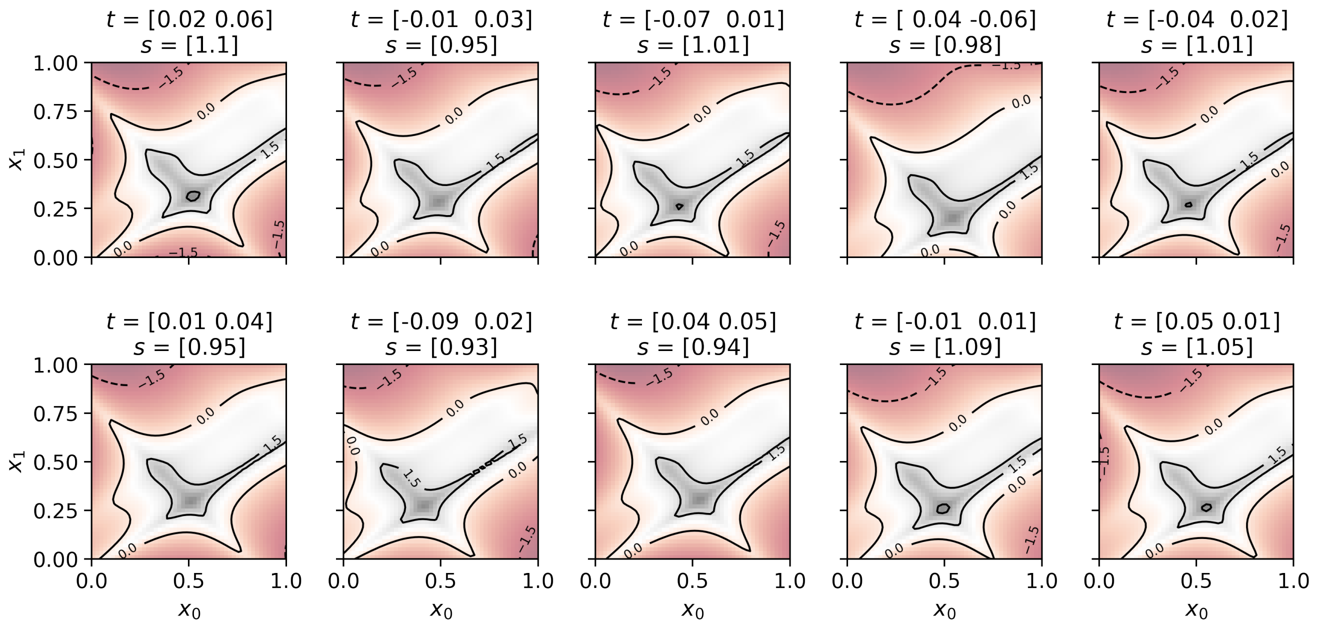

FIG. S2: Contour plots showing examples of the source campaigns for the 2-dimensional *Goldstein-Price* target campaign. The scale and shift parameters s and $\boldsymbol{t}$ are listed in subplot titles.

A. Continuous surfaces

We use three continuous-valued, d-dimensional analytical surfaces.

- *Goldstein-Price* ($d = 2$): This surface is evaluated in 2 dimensions on the square $x_i \in [-2, 2]\ \forall\ i = 1, 2$. The global minimum of the surface is at $f(\boldsymbol{x}^*) = 3$ at $\boldsymbol{x}^* = (0, -1)$.

- *AckleyPath* ($d = 3$): This surface is characterized by a flat outer region with a large hole in the middle, and has many local minima. The d-dimensional functional form is $f(\boldsymbol{x}) = -20 \exp\left(-0.2\sqrt{1/d \sum_{i=1}^{d} x_i^2}\right) - \exp\left(1/d \sum_{i=1}^{d} \cos(2\pi x_i)\right) + 20 + \exp(1)$, evaluated on the hypercube $x_i \in [-32.768, 32.768]\ \forall\ i = 1, \ldots, d$. The global optima is located at $\boldsymbol{x}^* = (0, \ldots, 0)$ with $f(\boldsymbol{x}^*)$

- *Hartmann* ($d = 3$): The 3-dimensional Hartmann function has 4 local optima and has the functional form $f(\boldsymbol{x}) = \sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{3} A_{i,j}(x_j, P_{i,j})^2\right)$, where $\alpha \in \mathbb{R}^4$, $\boldsymbol{A} \in \mathbb{R}^{3\times3}$ and $\boldsymbol{P} \in \mathbb{R}^{4\times3}$. The function is evaluated on the hypercube $x_i \in [0, 1]\ \forall\ i = 1, 2, 3$, and features a global minimum at $\boldsymbol{x}^* = (0.114614, 0.555649, 0.85251)$ with $f(\boldsymbol{x}^*) = -3.86278$.

The full results of the analytical tests on continuous surfaces are presented in Fig. S3 and S4. Fig. S3 compares the performance of single-task strategies (dashed traces) with that of their SeMOpt extensions by first training on the source campaign data and using the compound acquisition function of SeMOpt (solid traces of the same color). Regret is used as the performance metric. In most instances, using SeMOpt improves optimization performance with

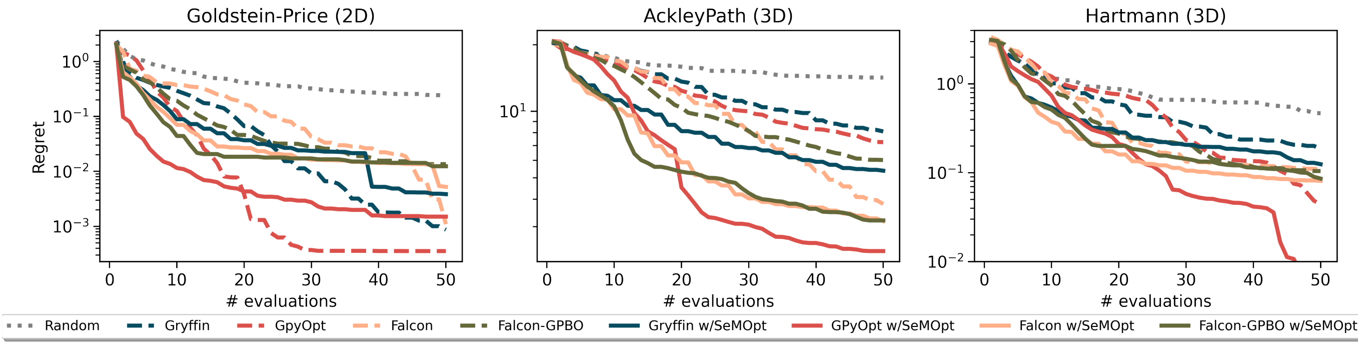

FIG. S3: Optimization traces for experiments on continuous surfaces comparing performance of single task strategies to their meta-learning analogues using the SeMOpt formalism. Dotted traces depict single task strategies, and meta-learning strategies are shown with solid traces of a matching color. Reported values represent the mean regret value over 40 independently seeded runs.

respect to the single task strategies. In light of these tests, it appears that the benefit of using SeMOpt is greater when the dimensionality of the parameter space is increased from 2 to 3.

Fig. S4 compares the optimization performance of techniques extended with SeMOpt (solid traces) to that of meta-learning techniques for Bayesian optimization with access to historical campaign data, including TAF-ME, RGPE, and DKT (alternating dotted and dashed traces). We also consider the ANP model, which both use the expected improvement acquisition function. In most cases, SeMOpt outperforms TAF-ME and ANP. On the 2-dimensional *Goldstein-Price* function, RGPE outperforms all other techniques by a significant margin. On the 3-dimensional *AckleyPath* function, single task optimizers extended with SeMOpt outperform all other methods. On the 3-dimensional *Hartmann* function, DKT, RGPE, and GPyOpt *w/*SeMOpt are the best performing methods.

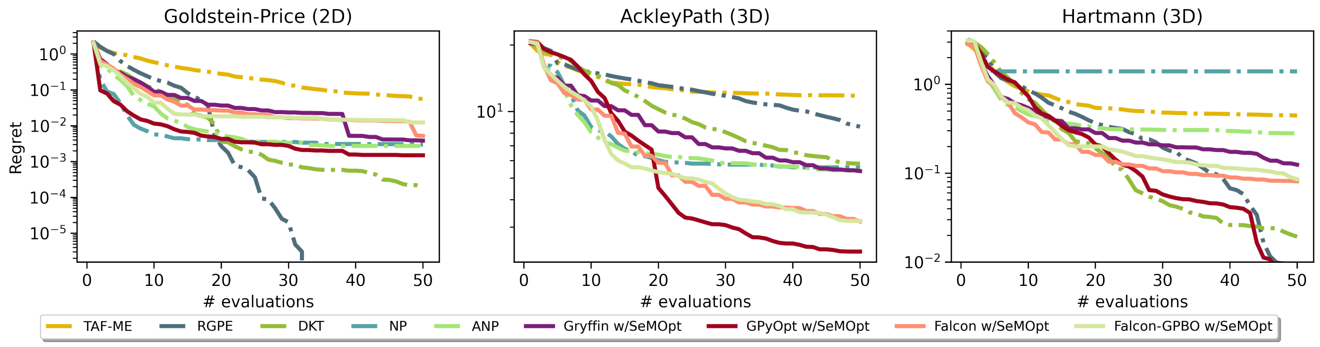

FIG. S4: Optimization traces for experiments on continuous surfaces comparing performance of SeMOpt strategies to other meta-learning optimization strategies. Solid traces depict SeMOpt strategies, while other meta-learning strategies are shown with alternative dot-dash traces. Reported values represent the mean regret value over 40 independently seeded runs.

B. Categorical surfaces

We also use two categorical-valued, d-dimensional analytical surfaces, with n options per dimension.

- *CatCamel* ($d = 2$, $n = 21$): This surface features a degenerate and pseudo-disconnected global minimum. In 2-dimensions, it has global minima at $(x_0, x_1) = (7, 11)$ and $(x_0, x_1) = (14, 10)$.

- *CatMichalewicz* ($d = 2$, $n = 21$): This surface features a sharper well where the global optimum is located. The number of psuedo-local minima scales factorially with the number of dimensions. It features a global minima at $(x_0, x_1) = (14, 10)$.

The results of the analytical tests on categorical functions are shown in Fig. S5 as overlayed box-and-whisker and strip plots, which represent distributions of the percentage of total parameter space needed for each method to identify the global optimum of the surface. A more efficient optimization strategy should be able to identify the global optimum with fewer evaluations. It should be noted that the *CatCamel* surface has two degenerate global optima, thus we measure the number of evaluations taken to identify one of the two. The best performing methods for each surface are indicated with an asterisk. For the *CatCamel* surface, DKT, along with all NP-based strategies have the best performance, needing to explore roughly 1% of the space ($4-5$ evaluations). On the *CatMichalewicz* surface, DKT and GPyOpt w/SeMOpt are the best performing methods, with much narrower distributions of explored space needed to reach the global optimum than the NP-based strategies. Numerical results from this experiment are summarized in Table S4. Median percentages of explored space are tabulated, along with the interquartile range in parentheses.

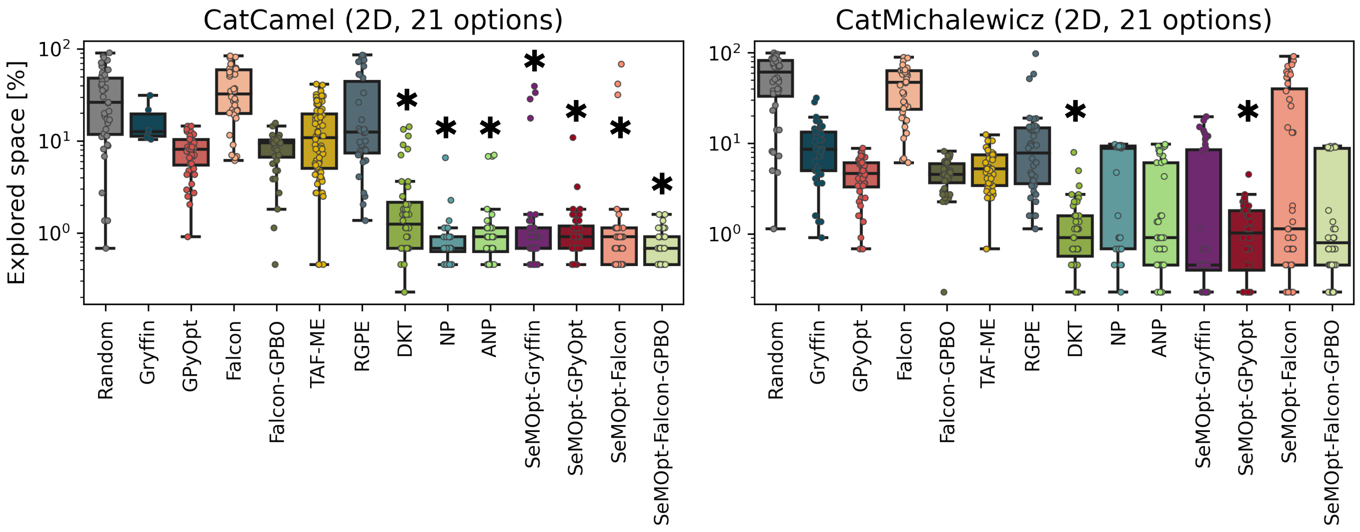

FIG. S5: Results of the analytical tests on categorical surfaces. Overlayed box-and-whisker and swarm plots show the distribution of the percent of total parameter space needed for each strategy to identify the (one of the) global optimum (optima) of the function. For each method we conducted 40 independently seeded optimization runs. Best performing techniques are indicated with an asterisk.

	CatCamel	CatMichalewicz
Random Search	26.24 (11.82, 48.13)	61.20 (33.03, 82.41)
Gryffin	12.67 (11.20, 19.57)	8.60 (4.98, 13.24)
GPyOpt	8.14 (5.49, 10.41)	4.64 (3.28, 6.11)
Falcon	32.47 (19.91, 59.28)	46.95 (19.85, 65.55)
Falcon-GPBO	9.50 (6.62, 10.18)	4.52 (3.68, 5.94)
TAF-ME	10.86 (5.03, 19.68)	5.20 (3.39, 7.47)
RGPE	12.44 (7.35, 44.63)	7.81 (3.56, 14.82)
DKT	1.24 (0.68, 2.15)	0.90 (0.57, 1.58)
NP	0.68 (0.62, 0.90)	8.82 (0.68, 9.28)
ANP	0.90 (0.62, 1.13)	0.90 (0.45, 6.11)
Gryffin *w*/SeMOpt	0.68 (0.68, 1.13)	0.45 (0.40, 8.48)
GPyOpt *w*/SeMOpt	0.90 (0.68, 1.19)	1.02 (0.40, 1.81)
Falcon *w*/SeMOpt	0.90 (0.45, 0.96)	10.18 (0.62, 49.32)
Falcon-GPBO *w*/SeMOpt	0.68 (0.45, 0.90)	0.79 (0.45, 8.82)

TABLE S4: Percentage of total parameter space needed for each optimization method to identify (one of) the global (optima) optimum on analytical categorical functions. The median percentage of space explored is reported, along with the interquartile range in parentheses. For each method we conducted 40 independently seeded optimization runs.

S.5. FULL RESULTS OF THE SIMULATED REACTION OPTIMIZATION EXPERIMENTS

In addition to the results presented in the main text, full results from our simulated reaction optimization experiments are presented here. Additionally, we test the pure NP and ANP strategies, which both use the expected improvement acquisition function. Optimization traces for the five reaction cases studied are shown in Fig. S6, and numerical cumulative regret values are tabulated in Table S5. Values represent the mean cumulative regret after 20 yield evaluations, averaged over 20 independently seeded runs (± standard errors on the mean). The best performing strategies are bolded, and statistical hypothesis testing was conducted using Welch's t-test.

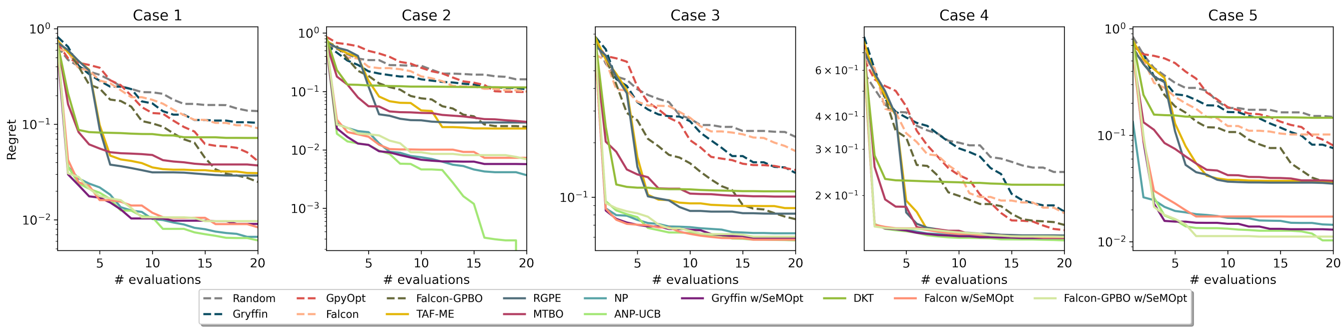

FIG. S6: Optimization traces for simulated reaction experiments. Traces show the best regret value (i.e. best yield value) achieved for each strategy, averaged over 20 independently seeded executions. Dashed traces represent "naïve" strategies (no access to source tasks), while solid traces represent meta-learning planners.

	Case 1	Case 2	Case 3	Case 4	Case 5
Random Search	5.04 ± 0.38	5.70 ± 0.49	6.56 ± 0.40	6.79 ± 0.38	5.37 ± 0.42
Gryffin	4.72 ± 0.55	4.14 ± 0.53	5.74 ± 0.54	6.60 ± 0.42	4.34 ± 0.64
GPyOpt	4.22 ± 0.56	6.40 ± 0.78	5.83 ± 0.78	5.83 ± 0.52	5.51 ± 0.61
Falcon	4.44 ± 0.46	4.62 ± 0.47	5.99 ± 0.55	5.74 ± 0.36	3.80 ± 0.46
Falcon-GPBO	3.43 ± 0.39	2.99 ± 0.27	4.16 ± 0.31	5.33 ± 0.30	3.21 ± 0.34
TAF-ME	2.79 ± 0.15	2.88 ± 0.26	3.76 ± 0.18	4.57 ± 0.14	2.81 ± 0.12
RGPE	2.70 ± 0.20	2.77 ± 0.21	3.57 ± 0.17	4.53 ± 0.17	2.47 ± 0.23
DKT	2.31 ± 0.51	3.27 ± 0.59	2.99 ± 0.29	5.00 ± 0.60	3.69 ± 0.83
MTBO	1.73 ± 0.16	1.69 ± 0.16	2.91 ± 0.29	3.70 ± 0.15	1.64 ± 0.11
NP	**1.09 ± 0.10**	**0.86 ± 0.10**	**1.87 ± 0.08**	**3.43 ± 0.08**	**0.95 ± 0.07**
ANP	**1.00 ± 0.08**	**0.92 ± 0.12**	**1.89 ± 0.09**	**3.36 ± 0.08**	**0.90 ± 0.08**
Gryffin *w*/SeMOpt	**0.84 ± 0.08**	**0.76 ± 0.09**	**1.90 ± 0.09**	**3.35 ± 0.07**	**0.96 ± 0.07**
Falcon *w*/SeMOpt	**0.90 ± 0.08**	**0.85 ± 0.10**	**1.87 ± 0.08**	**3.39 ± 0.07**	**1.02 ± 0.10**
Falcon-GPBO *w*/SeMOpt	**0.99 ± 0.06**	**0.95 ± 0.08**	**2.01 ± 0.07**	**3.46 ± 0.06**	**1.00 ± 0.06**

TABLE S5: Cumulative regret values over 20 optimization iterations averaged over 20 independently seeded runs for each simulated reaction case (reported with standard errors on the mean). The values for the best performing strategies in each case are bolded. Hypothesis testing was conducted using Welch's t-test.

S.6. FULL RESULTS AND DETAILS OF THE BUCHWALD-HARTWIG REACTION OPTIMIZATION EXPERIMENTS

In this section, full results of the Buchwald-Hartwig reaction optimization experiments are given, along with some additional details on the dataset. Fig. S7 shows the distributions of yield values reported by Ahneman *et al.* [15] independently for each aryl halide. Fig. S9-S11 show overlayed box-and-whisker and strip plots of the distributions of number of yield evaluations taken by each optimization strategy to identify a top-10, top-5 and top-2 yield, respectively. The best performing strategies for each aryl halide are denoted with an asterisk. Table S6 organizes the mean number of evaluations needed to reach a top-10, top-5 and top-2 yield for each strategy on each of the five aryl halide tasks (with ± standard error on the mean). Numerical values are averages over 40 independently seeded runs. The best performing strategies are bolded. Statistical hypothesis tests are conducted using a Wilcoxon signed-rank test.

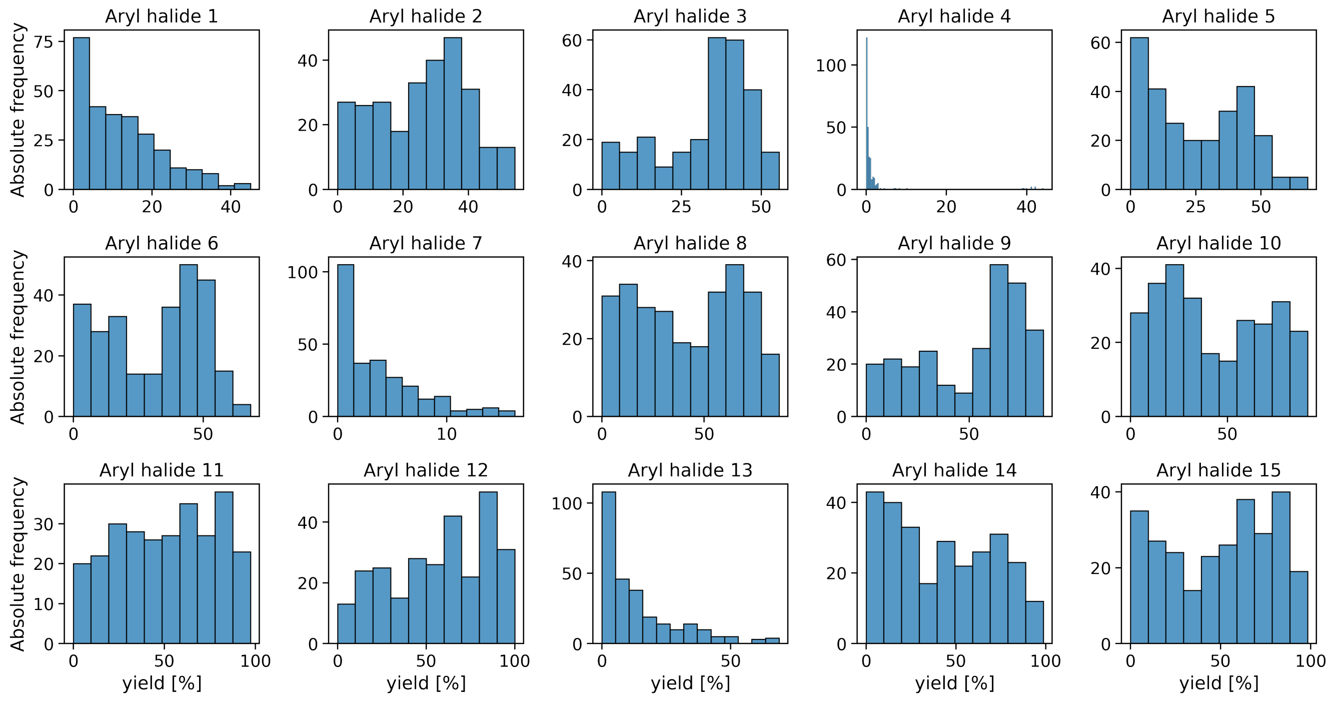

FIG. S7: Histogram of yield values for each aryl halide reported in Ahneman *et al.* [15].

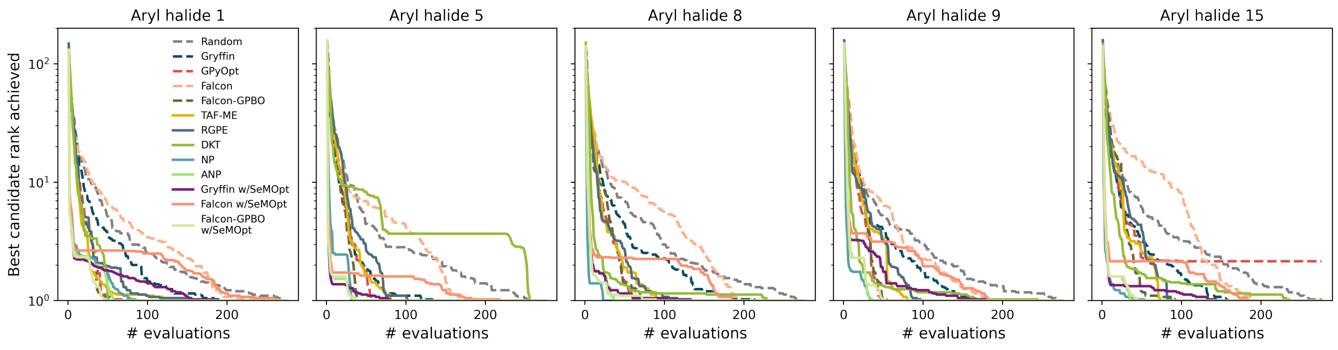

FIG. S8: Rank of the best performing candidate found by each of the studied optimization strategies averaged over 40 independently seeded optimization runs. Reaction conditions for each aryl halide are ranked with respect to their associated yield measurement, *i.e.* rank 1 corresponds to the reaction conditions which produce the highest yield for that aryl halide.

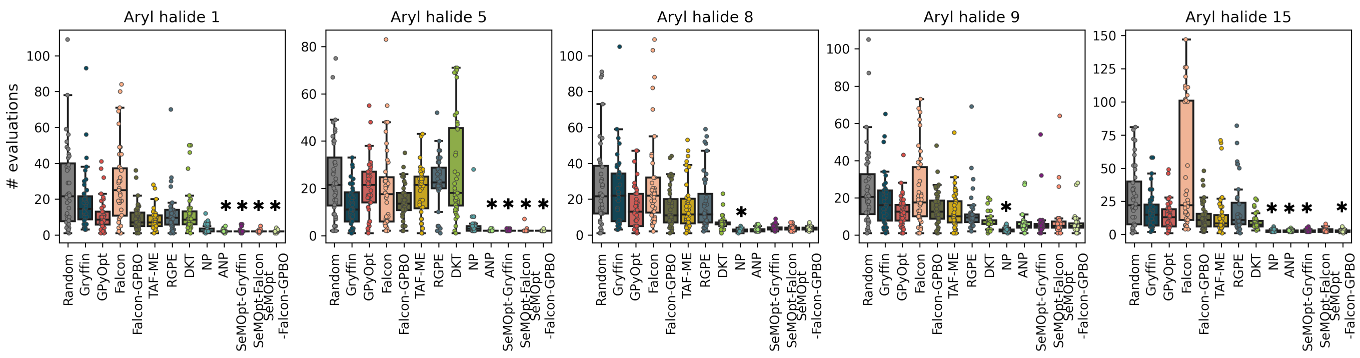

FIG. S9: Overlayed box-and-whisker and strip plots show the distributions of the number of evaluations needed for optimization strategies to evaluate reaction conditions leading to a top-10 yield measurement for each aryl halide. Each strategy is executed independently 40 times. Best performing strategies for each case are indicated with an asterisk.

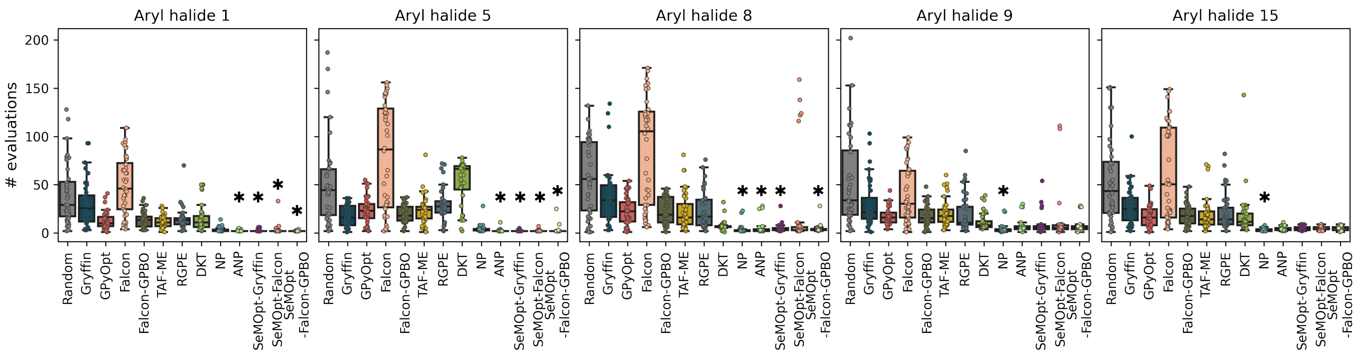

FIG. S10: Overlayed box-and-whisker and strip plots show the distributions of the number of evaluations needed for optimization strategies to evaluate reaction conditions leading to a top-5 yield measurement for each aryl halide. Each strategy is executed independently 40 times. Best performing strategies for each case are indicated with an asterisk.

[1] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. A. Eslami, and Y. W. Teh, “Neural Processes,” *arXiv:1807.01622 [cs, stat]*, July 2018. arXiv: 1807.01622.

[2] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *arXiv:1412.6980 [cs]*, Jan. 2017. arXiv: 1412.6980.

[3] H. Kim, A. Mnih, J. Schwarz, M. Garnelo, A. Eslami, D. Rosenbaum, O. Vinyals, and Y. W. Teh, “Attentive Neural Processes,” *arXiv:1901.05761 [cs, stat]*, Jan. 2019. arXiv: 1901.05761.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin, “Attention is All you Need,” in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[5] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy, “Botorch: A framework for efficient monte-carlo bayesian optimization,” 2020.

[6] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, “Efficient and Robust Automated Machine Learning,” p. 9.

[7] M. Feurer, J. T. Springenberg, and F. Hutter, “Using Meta-Learning to Initialize Bayesian Optimization of Hyperparameters,” p. 8.

[8] K. Swersky, J. Snoek, and R. P. Adams, “Multi-Task Bayesian Optimization,” in *Advances in Neural Information Processing Systems 26* (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), pp. 2004–2012, Curran Associates, Inc., 2013.

[9] G. E. Hinton and R. R. Salakhutdinov, “Using deep belief nets to learn covariance kernels for gaussian processes,” in *Advances in Neural Information Processing Systems* (J. Platt, D. Koller, Y. Singer, and S. Roweis, eds.), vol. 20, Curran Associates, Inc., 2007.

[10] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, “Deep kernel learning,” in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics* (A. Gretton and C. C. Robert, eds.), vol. 51 of *Proceedings of Machine Learning Research*, (Cadiz, Spain), pp. 370–378, PMLR, 09–11 May 2016.

[11] M. Wistuba and J. Grabocka, “Few-Shot Bayesian Optimization with Deep Kernel Surrogates,” *arXiv:2101.07667 [cs]*, Jan. 2021. arXiv: 2101.07667.

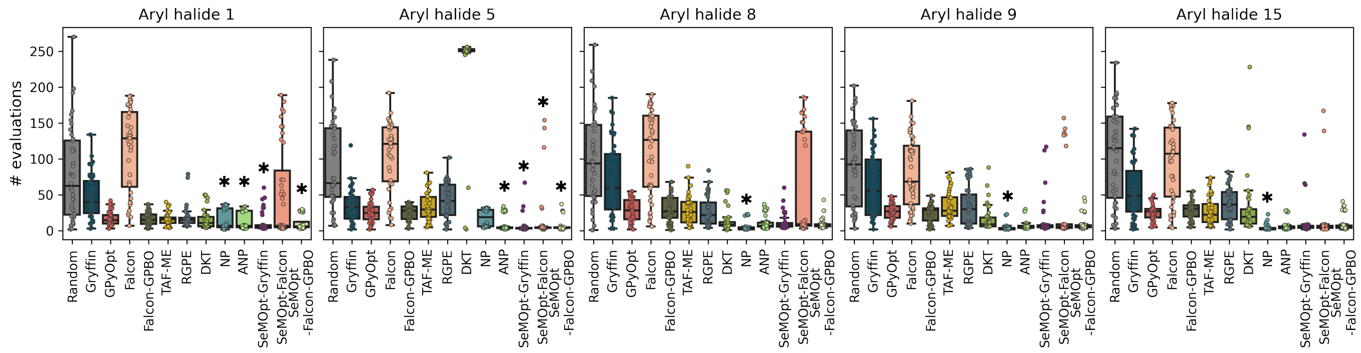

FIG. S11: Overlayed box-and-whisker and strip plots show the distributions of the number of evaluations needed for optimization strategies to evaluate reaction conditions leading to a top-2 yield measurement for each aryl halide. Each strategy is executed independently 40 times. Best performing strategies for each case are indicated with an asterisk.

[12] F. Häse, M. Aldeghi, R. J. Hickman, L. M. Roch, and A. Aspuru-Guzik, “Gryffin: An algorithm for Bayesian optimization of categorical variables informed by expert knowledge,” *Applied Physics Reviews*, vol. 8, p. 031406, Sept. 2021. Publisher: American Institute of Physics.

[13] F. Häse, L. M. Roch, C. Kreisbeck, and A. Aspuru-Guzik, “Phoenics: A Bayesian Optimizer for Chemistry,” *ACS Central Science*, vol. 4, pp. 1134–1145, Sept. 2018.

[14] M. Volpp, L. P. Fröhlich, K. Fischer, A. Doerr, S. Falkner, F. Hutter, and C. Daniel, “Meta-Learning Acquisition Functions for Transfer Learning in Bayesian Optimization,” *arXiv:1904.02642 [cs, stat]*, Feb. 2020. arXiv: 1904.02642.

[15] D. T. Ahneman, J. G. Estrada, S. Lin, S. D. Dreher, and A. G. Doyle, “Predicting reaction performance in C–N cross-coupling using machine learning,” *Science*, vol. 360, pp. 186–190, Apr. 2018. Publisher: American Association for the Advancement of Science Section: Report.

	top-m	Aryl halide 1	Aryl halide 5	Aryl halide 8	Aryl halide 9	Aryl halide 15
Random Search	10	26.35 ± 3.67	24.55 ± 2.70	28.18 ± 3.77	25.30 ± 3.41	28.27 ± 3.60
	5	40.10 ± 4.94	50.30 ± 6.80	57.73 ± 5.87	52.60 ± 7.35	54.05 ± 6.78
	2	79.47 ± 10.14	91.53 ± 9.70	99.92 ± 10.20	91.25 ± 9.03	104.95 ± 10.12
Gryffin	10	18.85 ± 2.64	12.80 ± 1.36	24.43 ± 3.17	18.20 ± 2.21	18.07 ± 2.20
	5	29.80 ± 3.67	17.60 ± 1.69	37.42 ± 4.69	29.27 ± 3.71	27.68 ± 3.09
	2	47.08 ± 4.75	34.58 ± 3.71	70.97 ± 7.92	63.90 ± 6.89	55.70 ± 6.13
GPyOpt	10	10.93 ± 1.45	20.70 ± 1.86	15.75 ± 1.86	13.00 ± 1.24	14.15 ± 1.70
	5	12.65 ± 1.42	23.45 ± 2.11	23.05 ± 2.24	17.30 ± 1.37	17.67 ± 1.76
	2	17.07 ± 1.54	25.73 ± 2.12	28.15 ± 2.43	26.57 ± 1.64	25.08 ± 1.78
Falcon	10	28.48 ± 3.43	19.77 ± 2.76	29.25 ± 4.15	24.18 ± 3.10	48.17 ± 7.29
	5	49.58 ± 4.70	79.00 ± 8.70	86.33 ± 8.63	40.83 ± 4.71	62.15 ± 7.66
	2	113.55 ± 8.93	105.33 ± 7.43	109.35 ± 9.30	81.65 ± 7.54	96.20 ± 8.46
Falcon-GPBO	10	9.80 ± 1.17	14.28 ± 1.20	15.38 ± 1.78	14.47 ± 1.50	13.43 ± 1.68
	5	13.70 ± 1.28	19.60 ± 1.45	23.35 ± 2.19	17.73 ± 1.74	18.52 ± 1.69
	2	16.27 ± 1.40	25.15 ± 1.58	29.90 ± 2.63	24.50 ± 1.92	27.80 ± 1.88
TAF-ME	10	8.55 ± 0.89	18.73 ± 1.67	16.32 ± 2.17	12.97 ± 1.62	14.90 ± 2.63
	5	11.47 ± 0.93	23.40 ± 2.20	22.20 ± 2.98	20.62 ± 2.14	19.07 ± 2.54
	2	16.35 ± 1.31	33.52 ± 2.99	29.60 ± 3.32	34.23 ± 2.94	28.05 ± 3.12
RGPE	10	11.93 ± 1.85	22.38 ± 1.79	17.95 ± 2.50	12.05 ± 1.79	18.98 ± 3.05
	5	14.53 ± 1.72	29.43 ± 2.73	23.40 ± 3.06	19.15 ± 2.90	21.32 ± 2.95
	2	18.48 ± 2.32	42.65 ± 3.83	28.50 ± 3.16	35.05 ± 4.21	38.60 ± 3.49
DKT	10	12.38 ± 1.92	27.77 ± 3.44	6.83 ± 0.57	7.25 ± 0.65	9.57 ± 0.89
	5	15.32 ± 2.17	53.38 ± 3.62	7.45 ± 0.80	11.85 ± 1.40	17.27 ± 3.52
	2	16.68 ± 2.23	234.75 ± 9.65	13.55 ± 2.19	17.05 ± 2.75	32.52 ± 6.95
NP	10	3.17 ± 0.37	4.20 ± 0.66	**2.50 ± 0.13**	**2.75 ± 0.19**	**2.62 ± 0.16**
	5	3.27 ± 0.39	4.28 ± 0.66	**3.60 ± 0.69**	**3.98 ± 0.71**	**2.95 ± 0.22**
	2	**14.62 ± 2.17**	18.00 ± 1.86	**4.95 ± 0.93**	**4.00 ± 0.71**	**3.90 ± 0.65**
ANP	10	**2.23 ± 0.12**	**2.08 ± 0.04**	3.17 ± 0.18	6.22 ± 0.84	**2.83 ± 0.11**
	5	**2.27 ± 0.11**	**2.33 ± 0.23**	**4.95 ± 0.98**	6.97 ± 0.99	4.70 ± 0.29
	2	**14.10 ± 1.82**	**8.25 ± 1.52**	11.40 ± 1.55	7.17 ± 0.98	6.83 ± 1.09
Gryffin *w*/SeMOpt	10	**2.33 ± 0.17**	**2.10 ± 0.05**	3.95 ± 0.22	6.78 ± 1.40	**2.88 ± 0.14**
	5	**2.38 ± 0.17**	**2.17 ± 0.09**	**4.92 ± 0.65**	8.20 ± 1.54	4.80 ± 0.30
	2	**11.65 ± 2.19**	**6.70 ± 1.80**	12.10 ± 1.82	14.62 ± 4.06	11.43 ± 3.74
Falcon *w*/SeMOpt	10	**2.15 ± 0.11**	**2.15 ± 0.13**	3.62 ± 0.22	7.88 ± 1.67	3.23 ± 0.22
	5	**3.15 ± 0.78**	**2.23 ± 0.13**	19.90 ± 6.75	11.43 ± 3.63	4.95 ± 0.30
	2	52.42 ± 9.93	**15.57 ± 5.63**	54.75 ± 10.95	25.52 ± 7.42	12.60 ± 5.13
Falcon-GPBO *w*/SeMOpt	10	**2.02 ± 0.07**	**2.02 ± 0.02**	3.67 ± 0.17	6.17 ± 0.83	**2.95 ± 0.15**
	5	**2.08 ± 0.05**	**2.77 ± 0.59**	**4.47 ± 0.63**	7.33 ± 1.10	4.78 ± 0.29
	2	**11.00 ± 1.54**	**6.60 ± 1.24**	9.88 ± 1.18	9.18 ± 1.56	9.72 ± 1.67

TABLE S6: Number of evaluations needed for strategies to identify the additive, catalyst and base setting corresponding to one of the top-m ($m = \{10, 5, 2\}$) yields for the respective aryl halide in Ahneman *et al.* [15]. We report the number of evaluations needed to identify a top-10, top-5 and top-2 yield value. Numerical values are averages over 40 independently seeded runs and are reported with standard errors on the mean. The best performing strategy for each of the target aryl halides and each of the top-m metrics are bolded. Statistical hypothesis tests are conducted using a Wilcoxon signed-rank test.