FunBO:
Discovering new acquisition functions for Bayesian Optimization with FunSearch

Virginia Aglietti

Workshop on Bayesian Decision-making and Uncertainty - NeurIPS 2024

Bayesian Optimization

$$x^* = \operatorname*{argmin}_{x \in \mathcal{X}} f(x)$$

$$f : \mathcal{X} \to \mathbb{R}^P$$
$$\mathcal{X} \subseteq \mathbb{R}^D$$

Setting

- Objective function is explicitly unknown and multimodal.
- Gradients are not available.
- We can query the objective function but evaluations are expensive/limited.
- Objective function evaluations may be perturbed by noise.

Goal

Find the global optima x^* in the smallest number of function evaluations

Applications

- Efficiency: Hyper-parameters optimization e.g. LLMs data mixture optimization
- Robotics: Optimizing gait parameters
- Science: Molecules/drugs design
- Causal Decision Making: identification of optimal policies in causal environments

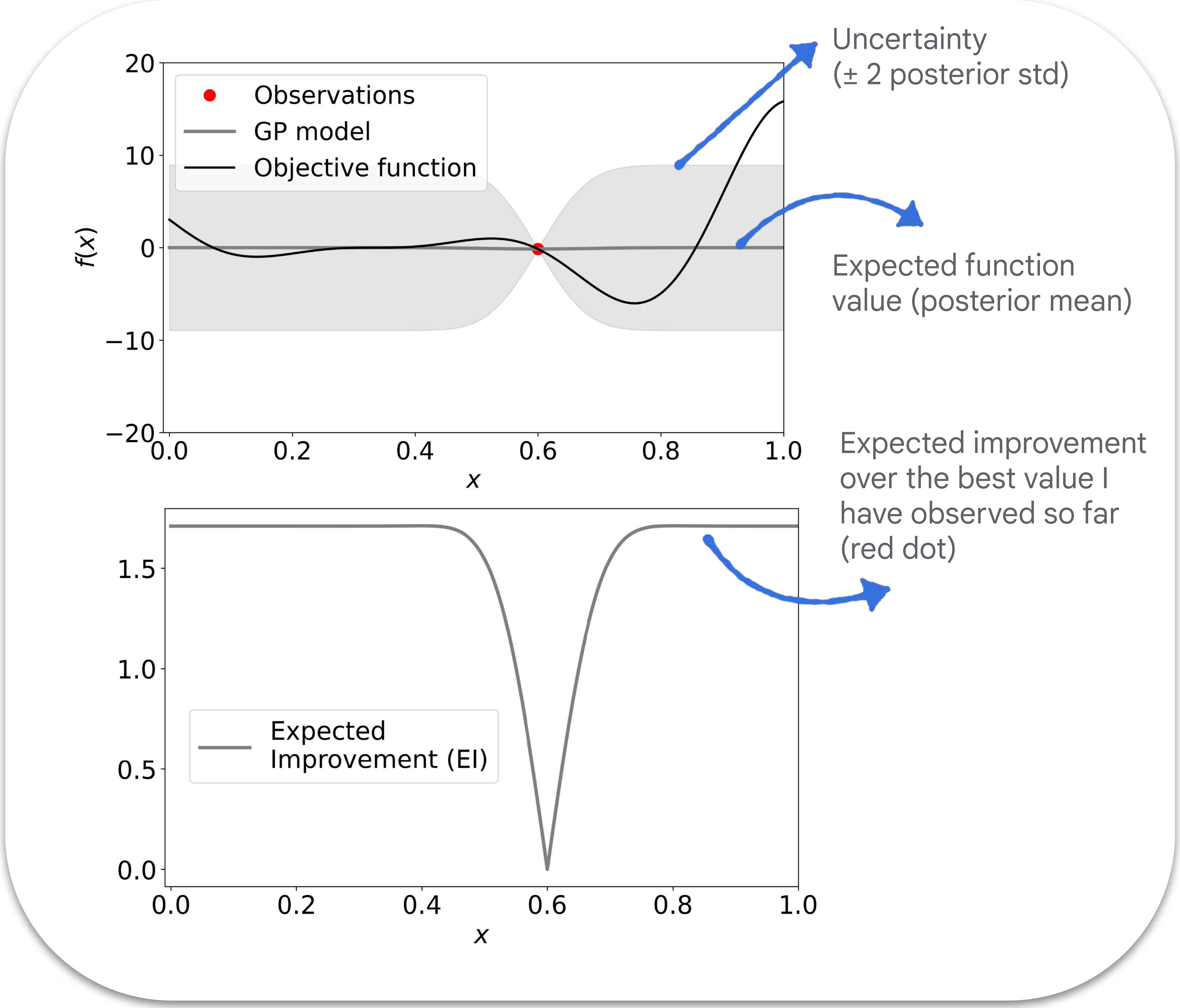
Bayesian Optimization

Surrogate model

- Model our belief about the objective function. This gets updated as we sequentially collect function evaluations.

Acquisition function (AF)

Determine the sequential acquisition of points thus balancing exploration and exploitation



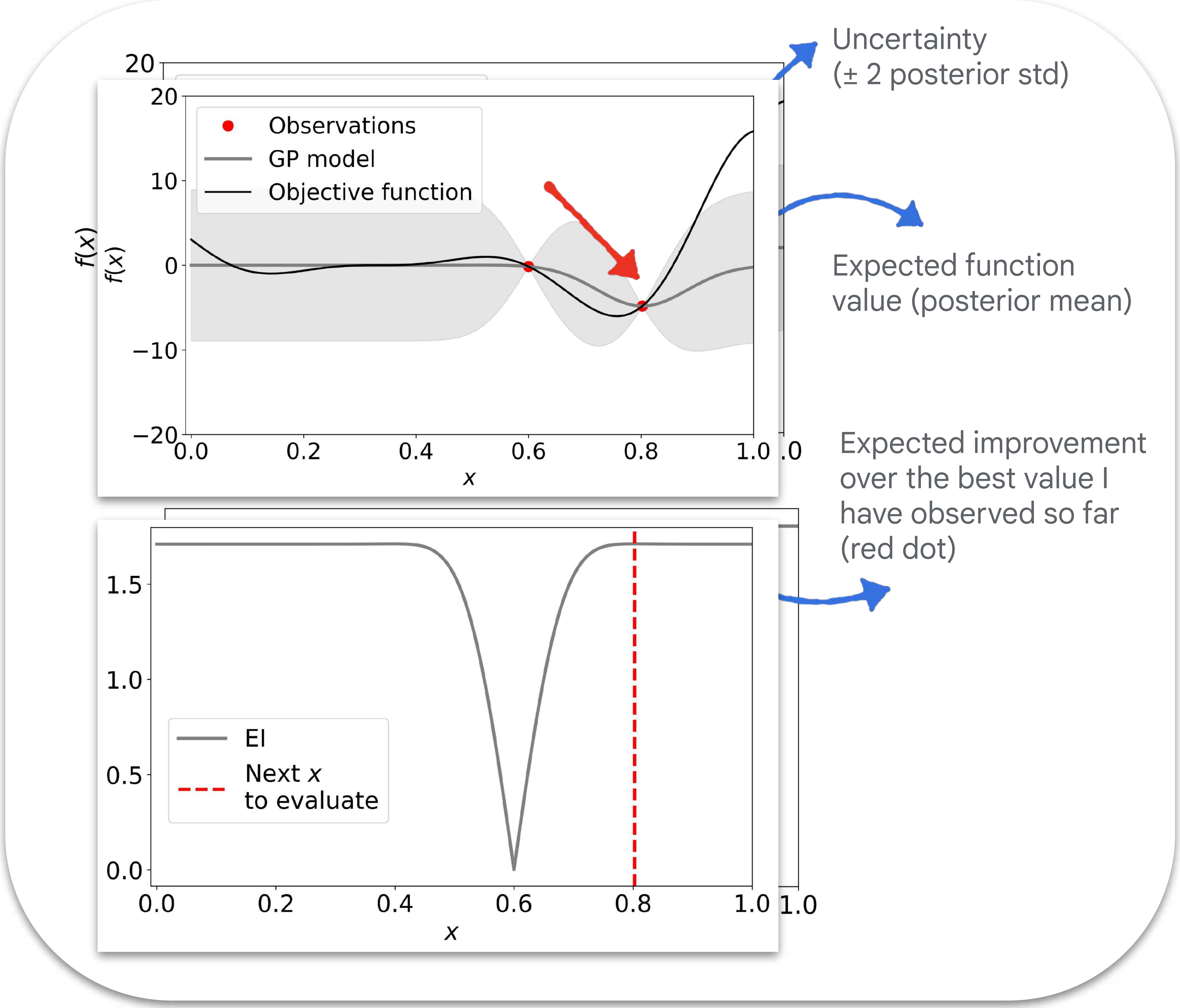
Bayesian Optimization

Surrogate model

- Model our belief about the objective function. This gets updated as we sequentially collect function evaluations.

Acquisition function (AF)

Determine the sequential acquisition of points thus balancing exploration and exploitation



General purpose or tailored AFs

General-purpose optimization strategies

Widely adopted general-purpose AFs that can be used out-of-the-box across BO algorithms and objective functions, e.g.:

Expected Improvement (EI)

Upper Confidence Bound (UCB)

Probability of Improvement (PofI)

...

Tailored optimization strategies

AFs tailored to specific objectives and are learned by transferring information from a set of related functions with a given training distribution. Generalization performance is not great.

MetaBO

Few-Shot BO (FSBO)

...

Can we develop a methodology that automatically identify new AFs capable of outperforming general-purpose and function-specific alternatives, both in and out of the training distribution?

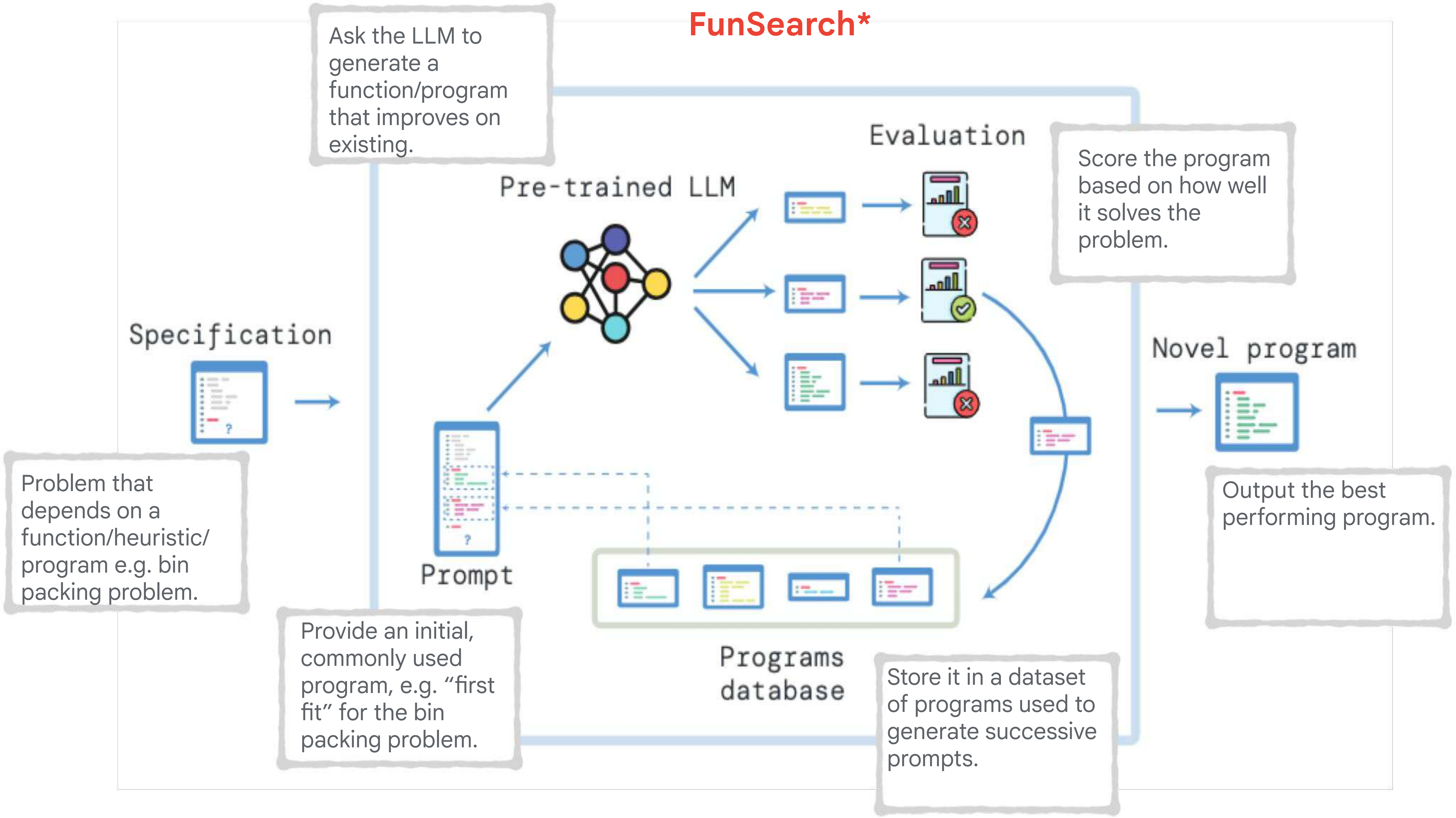
Research questions

- Can LLMs discover new *well performing acquisition functions (AFs)*?
- Can LLMs be used as a meta-learner for *hyperparameter optimization (HPO) problems*?
- Are discovered AFs *generalizing* with and across function classes?
- *Can* LLMs *be used in the context of few-shot adaptation?*

Research questions

*Romera-Paredes, B., Barekatain, M., Novikov, A. *et al.* [Mathematical discoveries from program search with large language models.](https://doi.org/10.1038/s41586-023-06924-6) *Nature* 625, 468–475 (2024). https://doi.org/10.1038/s41586-023-06924-6

- Can LLMs discover new *well performing acquisition functions (AFs)*?
- Can LLMs be used as a meta-learner for *hyperparameter optimization (HPO) problems*?
- Are discovered AFs *generalizing* with and across function classes?
- *Can LLMs be used in the context of few-shot adaptation?*



FunBO = FunSearch + BO

A Prompting the LLM

We focus on Python programs but FunBO can be readily applied to other languages supported by FunSearch.

```
"""Improve Bayesian Optimization by discovering a new acquisition function."""

def acquisition_function_v0(predictive_mean, predictive_var, incumbent, beta=1.0):
    """Returns the index of the point to collect ... (Full docstring in Fig. 8)"""
    # Code for lowest-scoring sampled AF.
    return ...

def acquisition_function_v1(predictive_mean, predictive_var, incumbent, beta=1.0):
    """Improved version of 'acquisition_function_v0'."""
    # Code for highest-scoring sampled AF.
    return ...

def acquisition_function_v2(predictive_mean, predictive_var, incumbent, beta=1.0):
    """Improved version of the previous 'acquisition_function'."""
```

A Prompting the LLM

We focus on Python programs but FunBO can be readily applied to other languages supported by FunSearch.

```
def acquisition_function(predictive_mean, predictive_var, incumbent, beta=1.0):
  """Returns the index of the point to collect ... (Full docstring in Fig. 8)."""
  z = (incumbent - predictive_mean) / np.sqrt(predictive_var)
  predictive_std = np.sqrt(predictive_var)
  vals = (incumbent - predictive_mean) * stats.norm.cdf(z) + predictive_std * stats.norm.pdf(z)
  return np.argmax(vals)
```

```
"""Improve Bayesian Optimization by discovering a new acquisition function."""

def acquisition_function_v0(predictive_mean, predictive_var, incumbent, beta=1.0):
    """Returns the index of the point to collect ... (Full docstring in Fig. 8)"""
    # Code for lowest-scoring sampled AF.
    return ...

def acquisition_function_v1(predictive_mean, predictive_var, incumbent, beta=1.0):
    """Improved version of ‘acquisition_function_v0‘."""
    # Code for highest-scoring sampled AF.
    return ...

def acquisition_function_v2(predictive_mean, predictive_var, incumbent, beta=1.0):
    """Improved version of the previous ‘acquisition_function‘."""
```

Docstring:

Returns the index of the point to collect in a vector of eval points.

Given the posterior mean and posterior variance of a GP model for the objective function, this function computes an heuristic and find its optimum. The next function evaluation will be placed at the point corresponding to the selected index in a vector of possible eval points.

Args:
predictive_mean: an array of shape [num_points , dim] containing the predicted mean values for the GP model on the objective function for ‘num_points ’ points of dimensionality ‘dim ’.

predictive_var: an array of shape [num_points , dim] containing the predicted variance values for the GP model on the objective function for ‘num_points ’ points of dimensionality ‘dim ’.

incumbent: current optimum value of objective function observed.

beta: a possible hyperparameter to construct the heuristic.

Returns:
An integer representing the index of the point in the array of shape [num_points , dim] that needs to be selected for function evaluation.

B Scoring the AFs

FunBO needs a scoring mechanism that captures small improvements in the proposed AF so as to steer the LLM toward promising regions of the function space. We scored generated functions based on:

- Distance to the true optimum. Value in [0, 1] where 0 corresponds to `found min=initial_min_y` and 1 corresponds to `found min=true_min`

```
score_distance = 1.0 - np.abs(found_min - true_min) / (initial_min_y - true_min)
```

Accurateness

- Percentage of total number of function evaluations needed to identify the optimum. Value in [0, 1] where 0 corresponds to `percentage_steps_needed = 100%` (no convergence) and 1 corresponds to `percentage_steps_needed = 0%` (convergence at first trial)

```
score_steps = 1.0 - percentage_steps_needed
```

Efficiency

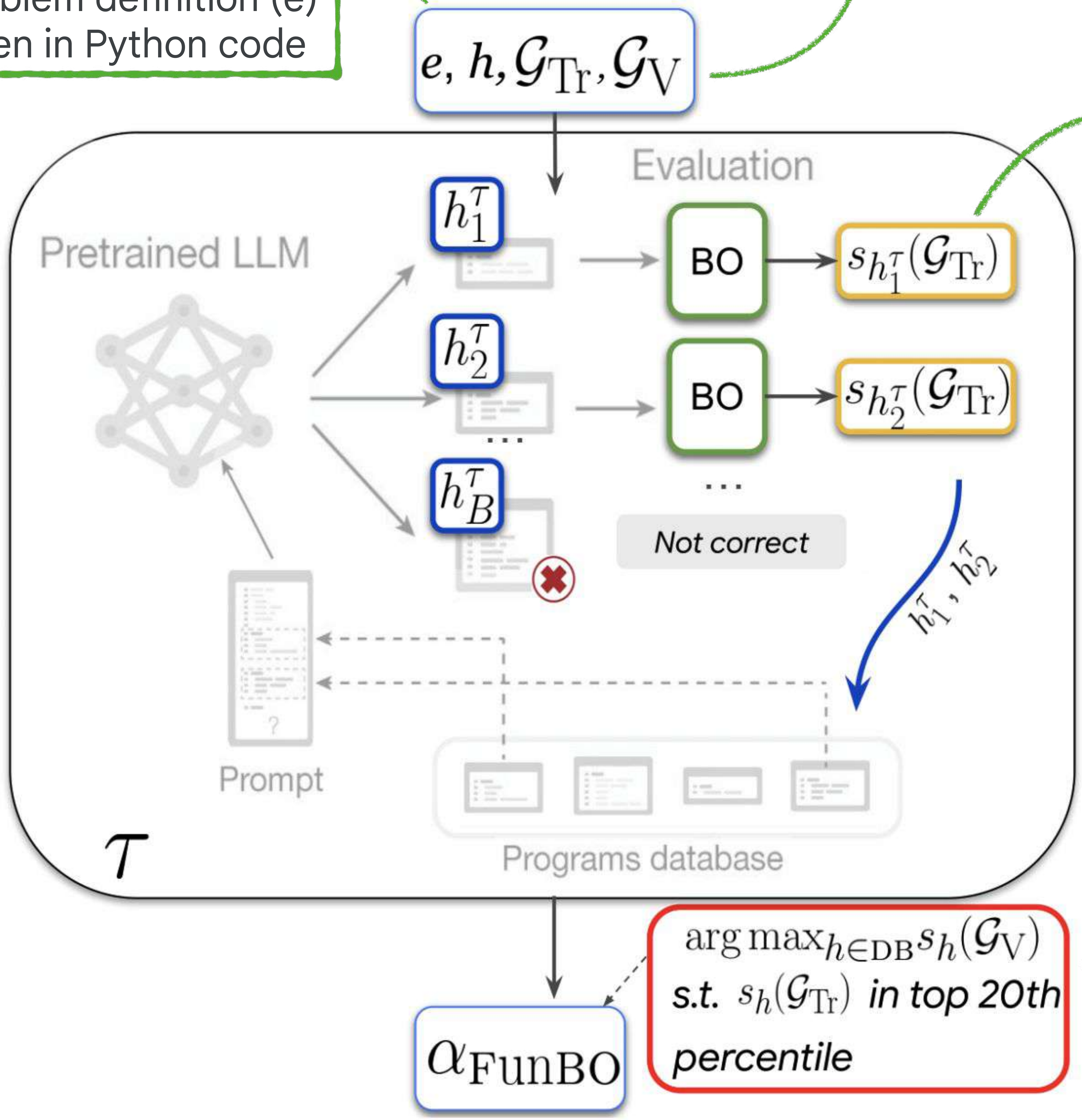
→ *Binary score* giving 1 or 0 based on weather algorithm converges to the global optimum does not provide enough signal during the exploration phase.

→ *Negative normalized cumulative regret* does not provide significant advantages over the currently adopted scoring mechanism

FunBO

Initial AF (h) and problem definition (e) given in Python code

Sets of functions used for training (Tr) and validation (V). Set differently depending on the experiment.



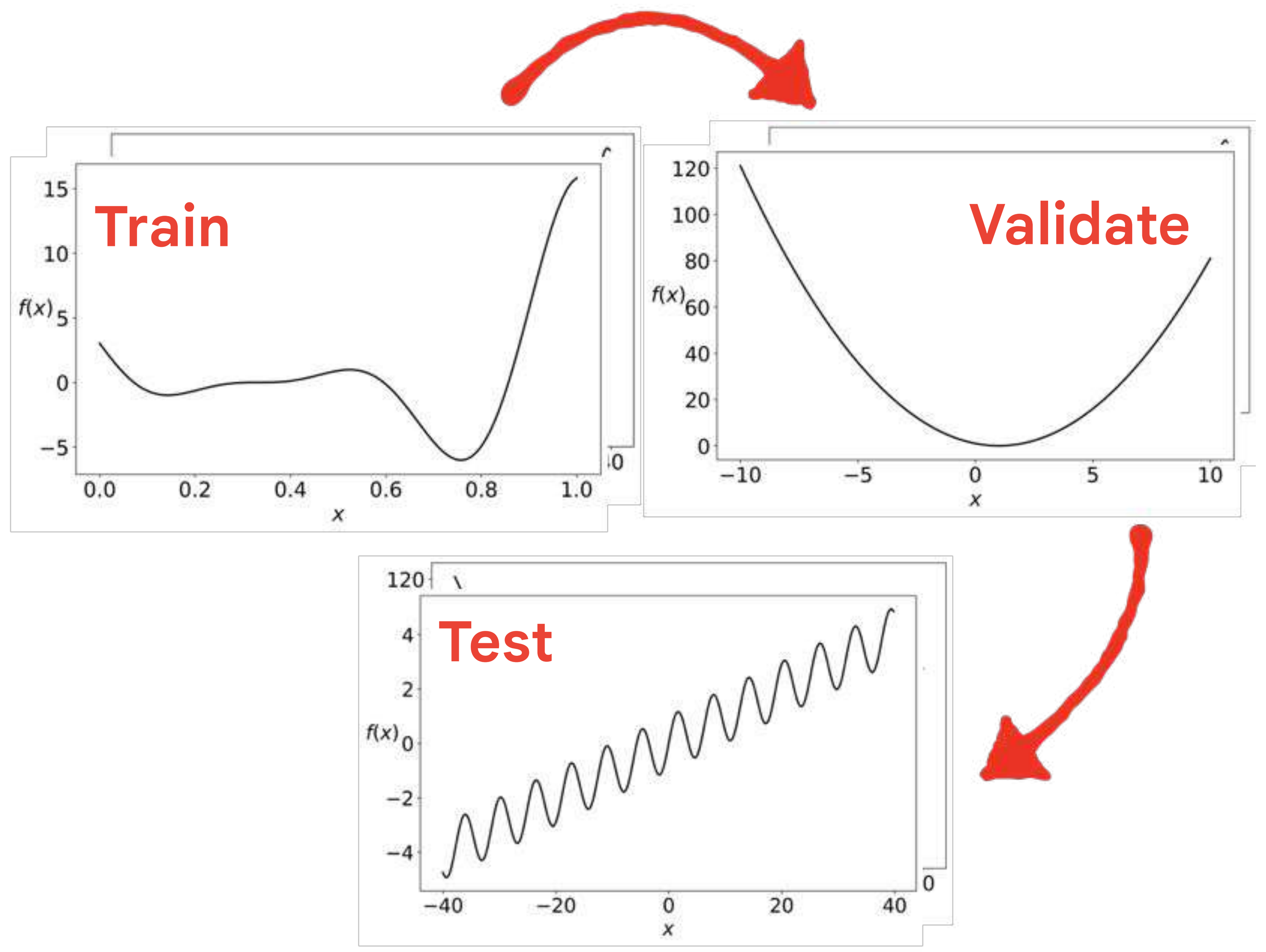
Score for each valid generated function given the performance of the corresponding BO run.

Scored AFs are added to the programs database which follows an island model. Prompts are created by sampling AFs from it.

The found AFs can be then inspected and tested on the optimization of a third set of functions (different from G_{Tr} and G_V)

Experiments*

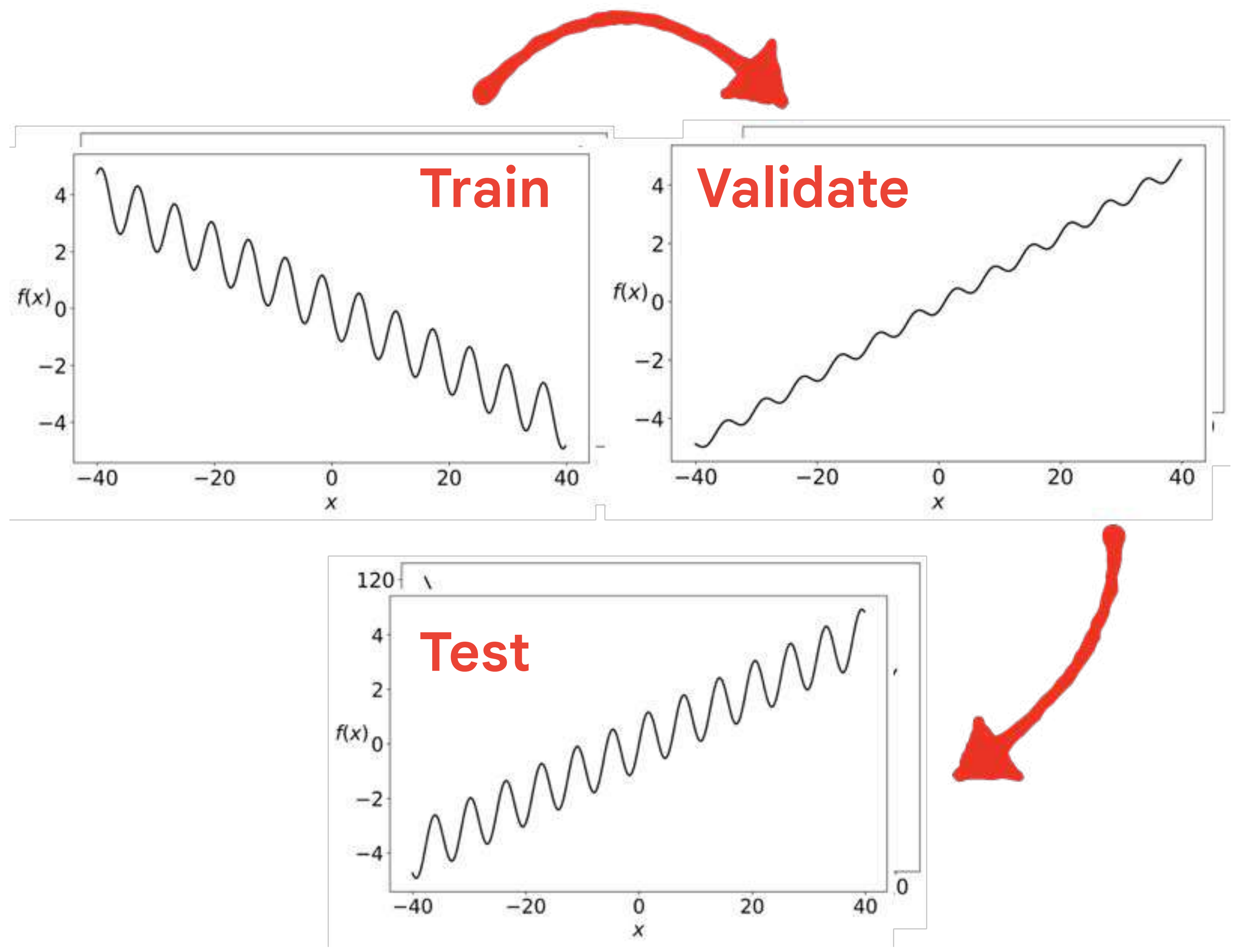
1 OOD-Bench



Generalization across function classes/out of the training distribution.

- Train and test on different standard global optimization benchmarks.
- Compare with standard AFs (EI, UCB, Probability of Improvement, Mean).

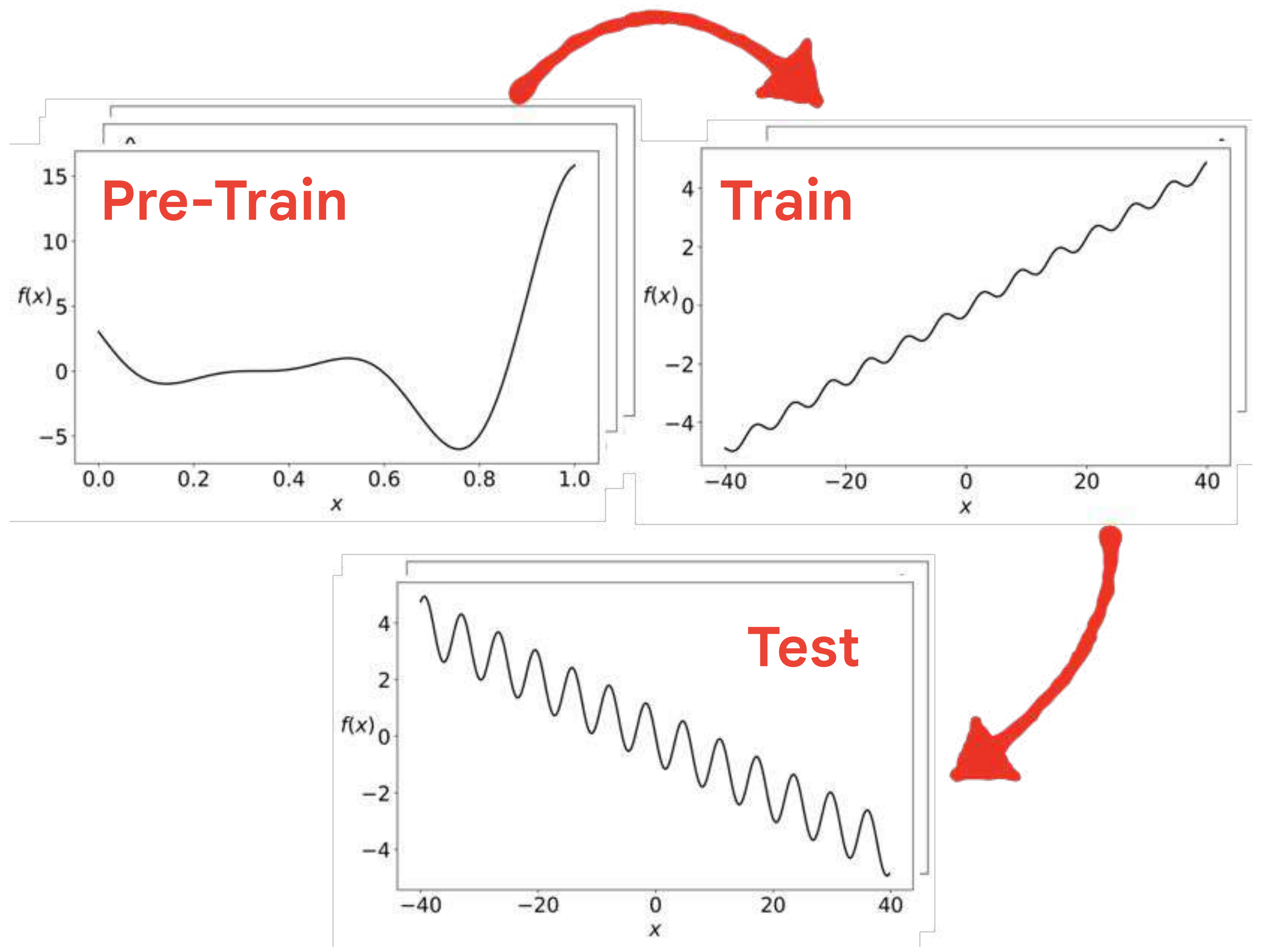
2 ID-Bench



Generalization within function classes/in the training distribution.

- Train and test on different instances of the target function.
- Compare with standard AFs and meta-learning approaches (MetaBO, learning a Neural AF via PPO).

3 Few-shots

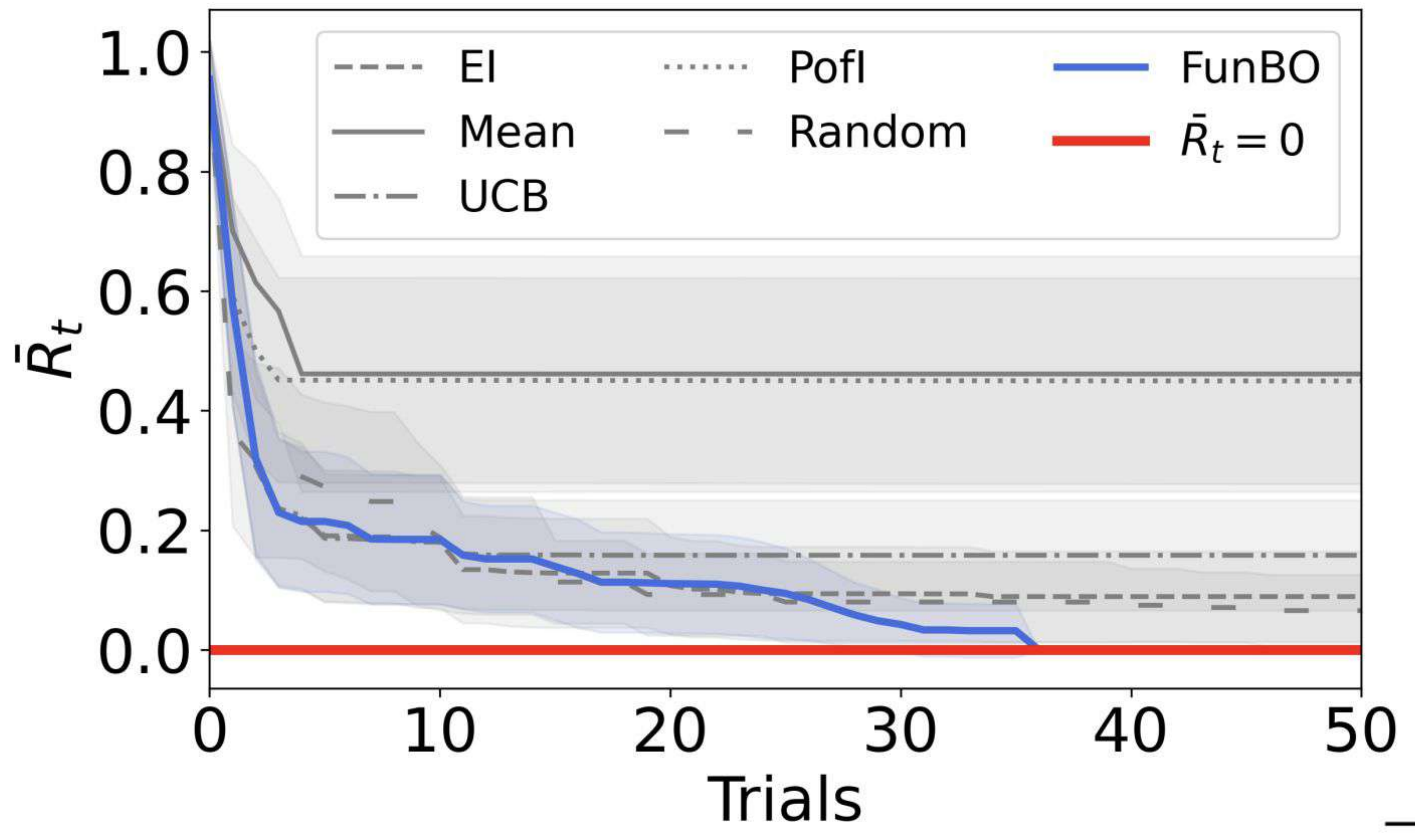


Quick adaptation of an AF using few instances of a target function.

- Train a general AF, adapt it with 5 instances of the target task and test on other instances from the same class.
- Compare against standard AFs and few-shot learning approaches (FSAF, AFs trained on GPs adapted via a DQN algorithm).

*All experiments are run with Codey - an LLM fine-tuned on a large code corpus and based on the PaLM model family

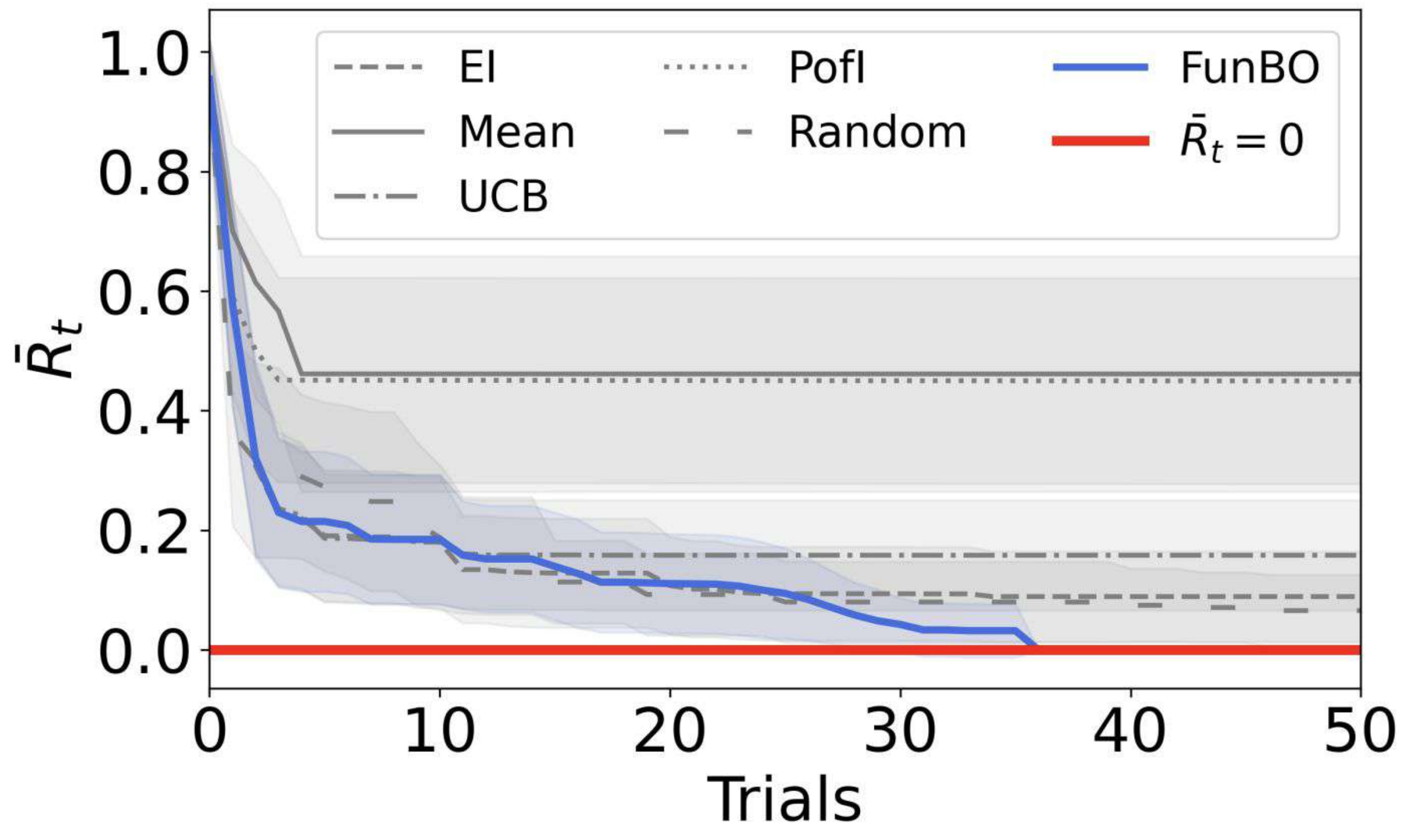
1 OOD-Bench



- **Training functions**: Ackley, Levy, and Schwefel (all 1D)
- **Validation function**: Rosenbrock (1D)
- **Test functions**: 50 scaled and translated instances of Sphere (d = 1), Styblinski-Tang (d = 1), Weierstrass (d = 1), Beale (d = 2), Branin (d = 2), Michalewicz (d = 2), Goldstein-Price (d = 2) and Hartmann (d = 3, d = 6)

	d	$\mathcal{X}$	N_{SG}	ℓ	σ_f^2	σ_f^2
Ackley	1	$[-4, 4]$	1000	0.21	28.19	$1e-5$
Levy	1	$[-10, 10]$	1000	1.05	83.32	$1e-5$
Schwefel	1	$[-500, 500]$	1000	18.46	76868.65	$1e-5$
Rosenbrock	1	$[-5, 10]$	1000	1.20	87328.20	$1e-5$
Sphere	1	$[-5, 5]$	1000	18.46	924202.43	$1e-5$
Styblinski-Tang	1	$[-5, 5]$	1000	7.34	119522207.86	$1e-5$
Weierstrass	1	$[-0.5, 0.5]$	1000	0.01	0.39	$1e-5$
Beale	2	$[-4, 5]^2$	10000	0.46	546837.32	$1e-5$
Branin	2	$[-5, 10] \times [0, 15]$	10000	4.65	155233.52	$1e-5$
Michalewicz	2	$[0, \pi]^2$	10000	0.22	0.10	$1e-5$
Goldstein-Price	2	$[-2, 2]^2$	10000	0.27	117903.96	$1e-5$
Hartmann-3	3	$[0, 1]^3$	1728	$[0.716, 0.298, 0.186]$	0.83	$1.688e-11$
Hartmann-6	6	$[0, 1]^6$	729	1.0	1.0	$1e-5$

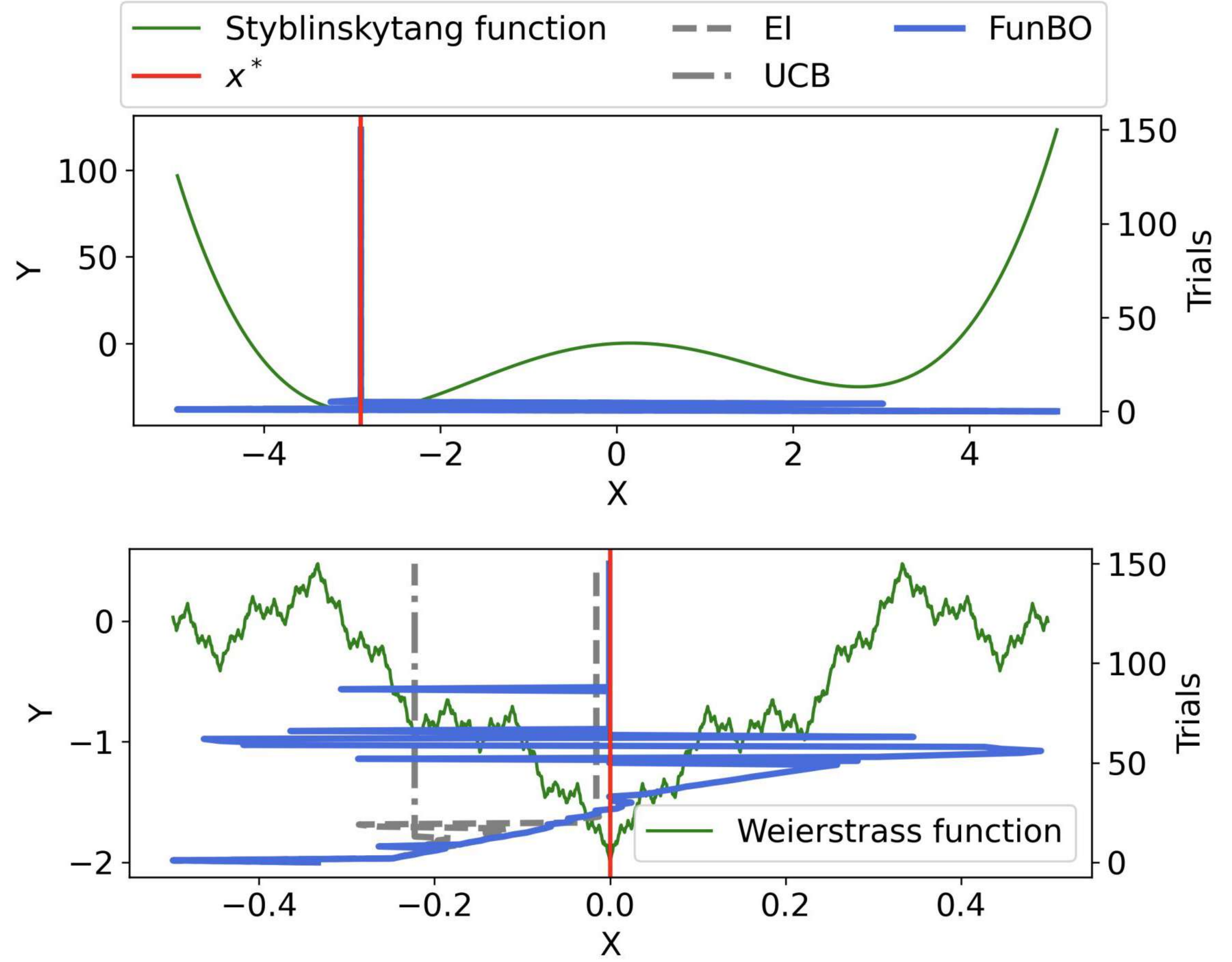
1 OOD-Bench



- **Training functions**: Ackley, Levy, and Schwefel (all 1D)
- **Validation function**: Rosenbrock (1D)
- **Test functions**: 50 scaled and translated instances of Sphere (d = 1), Styblinski-Tang (d = 1), Weierstrass (d = 1), Beale (d = 2), Branin (d = 2), Michalewicz (d = 2), Goldstein-Price (d = 2) and Hartmann (d = 3, d = 6)

```
def acquisition_function(predictive_mean
    predictive_var, incumbent, beta=1.0):
  """Returns the index of the point to collect..."""
  predictive_std = np.sqrt(predictive_var)
  diff_mean_std = (incumbent - predictive_mean + beta
  ↪  * predictive_std)
  z = diff_mean_std / predictive_std
  vals = diff_mean_std * stats.norm.cdf(z) +
  ↪  predictive_std * stats.norm.pdf(z)
  return np.argmax(vals)
```

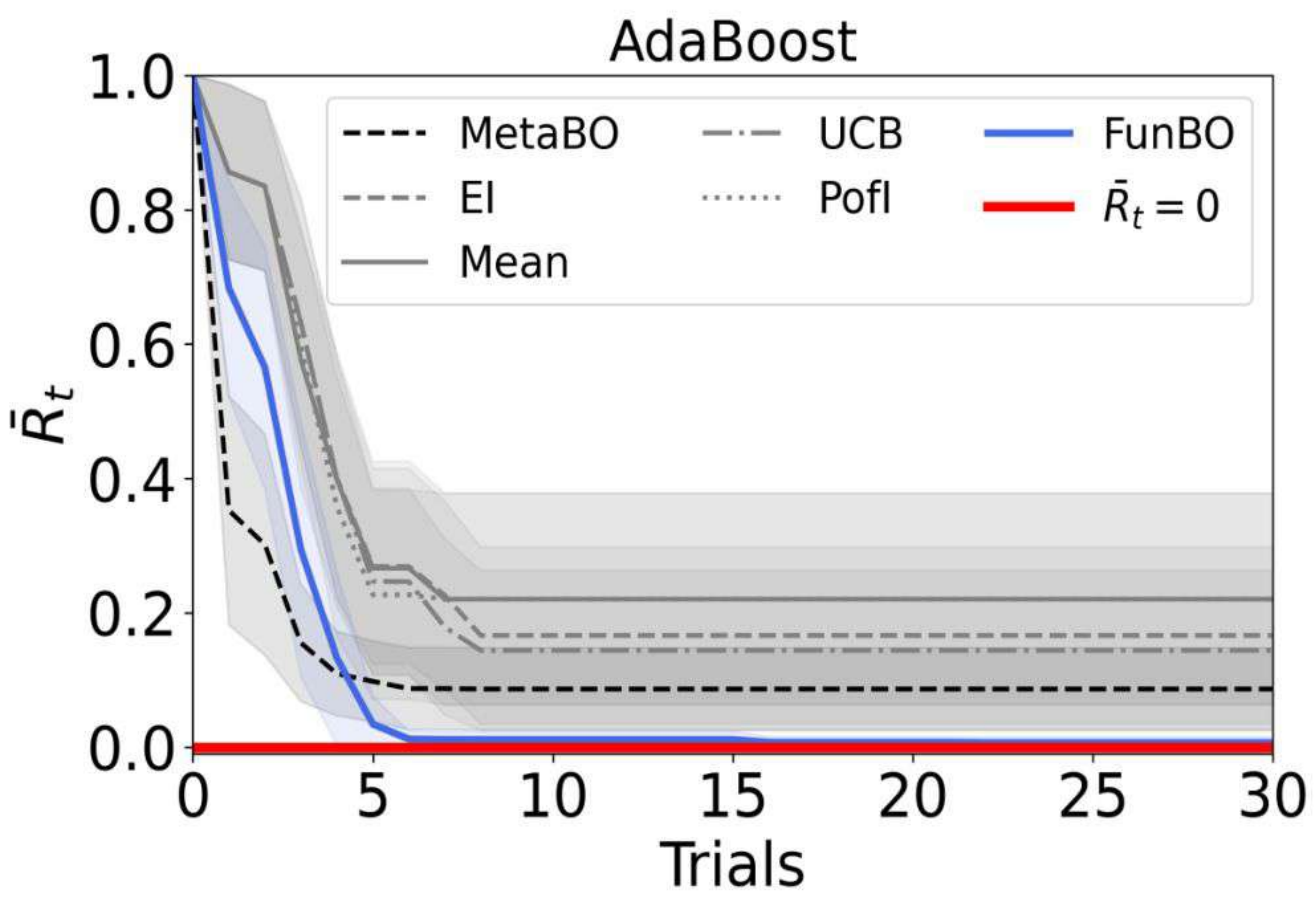
Better exploration of the input space for some functions with many local minima.

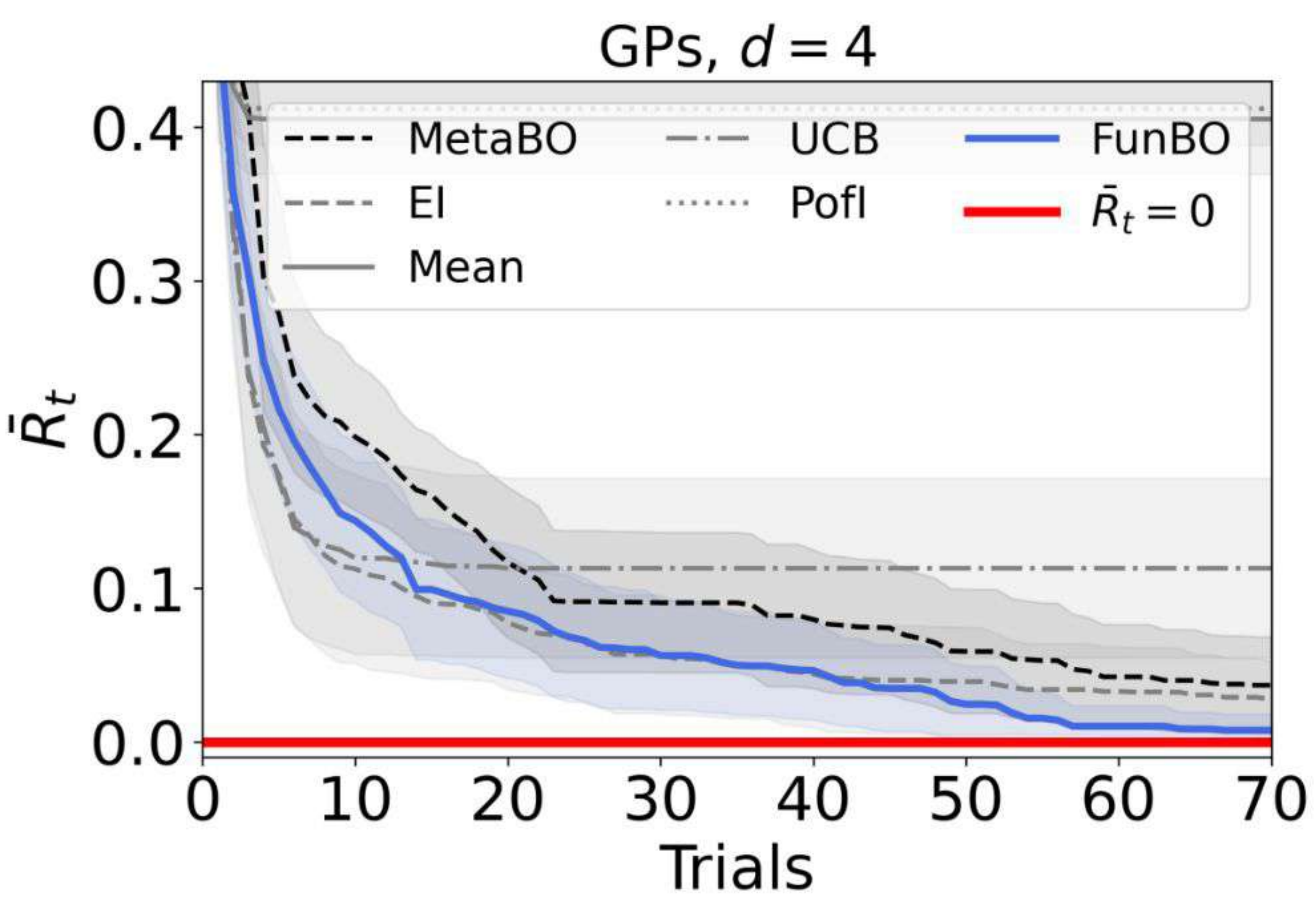


2 ID-Bench

- **Training functions**:
  - *Synthetic functions*: 20 scaled and translated instances
- **Validation function**:
  - *Synthetic functions*: 5 scaled and translated instances
- **Test functions**:
  - *Synthetic functions*: 100 scaled and translated instances

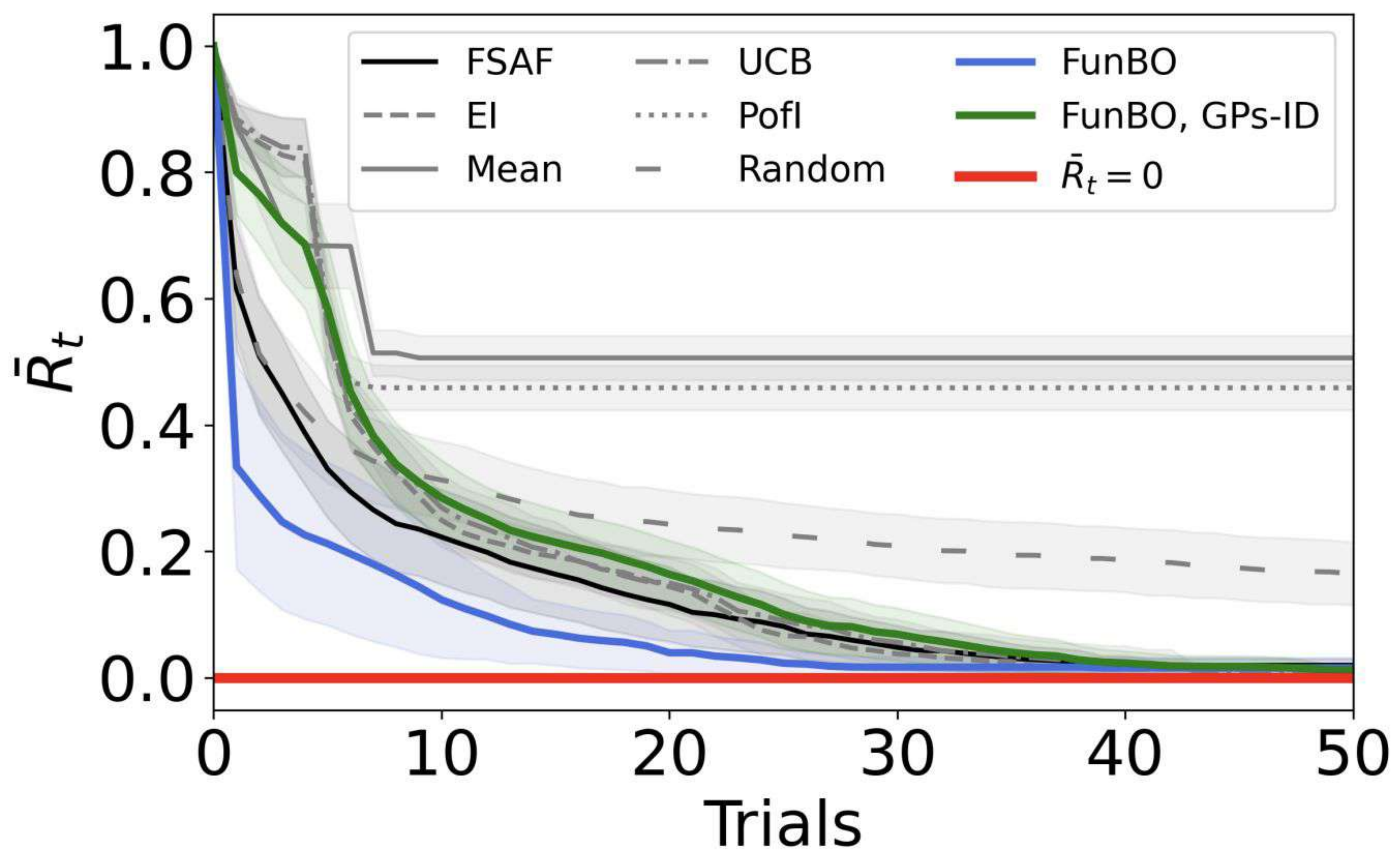
2 ID-Bench





- **Training functions**:
  - *HPO*: AdaBoost loss (2D) on 30 datasets
  - GPs: 20 functions from a GP prior (*3D*), RBF kernel and length-scale drawn uniformly from [0.05, 0.5].
- **Validation function**:
  - *HPO*: AdaBoost loss on 5 datasets
  - GPs: 5 functions sampled similarly from a GP prior (3D)
- **Test functions**:
  - *HPO:* AdaBoost loss on 15 datasets
  - GPs: 100 functions sampled similarly from a GP prior (*4D*).

3 Few-shots



We take the AFs found for GPs and adapt it with

- **Training functions**: 5 instances of scaled and translated Ackley (2D)
- **Validation function**: -
- **Test functions**: 100 instances of scaled and translated Ackley

Research questions

- Can LLMs discover new *well performing acquisition functions (AFs)*?
- Can LLMs be used as a meta-learner for *hyperparameter optimization (HPO) problems*?
- Are discovered AFs *generalizing* with and across function classes?
- *Can LLMs be used in the context of few-shot adaptation?*

Yes, using FunBO we were able to discover new well-performing AFs written in computer code across a variety of optimization problems. 1 2 3

Yes, our empirical investigation suggests that FunBO can identify AFs for the optimization of the hyper-parameters of AdaBoost and SVM. 2

FunBO achieves performances that are comparable to learned AFs as well as general purpose AFs. 1 2

FunBO can quickly adapts an AF based on 5 instances of the target function. 3

Looking ahead

Limitations

- Scalability to large number of functions in training/validation sets.
- Simple scoring mechanism.
- Experimental cost.
- Variance of the results.

Possible extensions

- Discover new AFs for various adaptations of this problem e.g. constrained or *non myopic optimization*, noisy evaluations, or alternative surrogate models etc.
- How and what assumptions can be encoded within AFs, based on the desired program characteristics and prior knowledge about the objective function.
- Identify AFs that can be added to the standard suite of AFs available in BO packages
- Extend this approach to select actions in a causal environment e.g. encode prior knowledge about cause-effect relationships or previously observed experimental outcomes, leverage LLMs' prior causal knowledge

Thank you.

Read more about FunBO

Get in touch:
causal-intelligence@google.com

FunBO Team

Ira Ktena

Jessica Schrouff

Eleni Sgouritsa

Virginia Aglietti

Francisco Ruiz

Alan Malek

Alexis Bellot

Silvia Chiappa

Google DeepMind