Human-Centered AutoML

Marius Lindauer

Leibniz Universität Hannover
Germany

/ LindauerMarius
m.lindauer@ai.uni-hannover.de

Katharina Eggensperger

Eberhard Karls Universität Tübingen
Germany

/ KEggensperger
katharina.eggensperger@uni-tuebingen.de

Leibniz Universität Hannover

LUH|AI

AutoML.org

Eberhard Karls Universität Tübingen

machine learning
new perspectives
for science

Story Line Today

1. How to reach more people with AutoML?
    a. What's missing after 10+ years of research on AutoML?
2. What can we learn from running AutoML?
    a. Local explanations
    b. Global explanations
3. How can you augment AutoML with your own expert knowledge?
4. How will LLMs change the game?

DATA

AutoML

Optimization and automation of tedious design decisions of a complete ML pipeline in order to obtain a model with peak performance.

GAUSSIAN PROCESS

BNN

RANDOM FOREST

BAYESIAN OPTIMIZATION (BO)

Estimate the model's performance for unknown hyperparameters.

surro-gate model → acquisition function → TRAIN AND EVALUATE

EVOLUTIONARY ALGORITHM (EA)

Select, mutate and recombine configurations.

HYPERPARAMETER OPTIMIZATION (HPO)

Search for the best hyperparameter configuration given an algorithm.

PICK NEXT HYPER-PARAMETERS → hyperparameter configuration

CLASSIC ML: TRAIN → model → EVALUATE → model parameters

SINGLE-CRITERION OPTIMIZATION

MULTI-CRITERIA OPTIMIZATION

SPEED UP

Meta-learning across datasets

Grey-box optimization/ learning curve prediction

Multi-fidelity optimization

NEURAL ARCHITECTURE SEARCH (NAS)

Search for the best neural network architecture on different hierarchical levels given a task.

META-LEARNING

Learn across tasks.

LEARNING TO LEARN

POPULATION-BASED TRAINING

TRAIN → model → START CONFIGURATION

DYNAMIC ALGORITHM CONFIGURATION (DAC)

INTERPRETE

MODEL(S)

EVALUATE

CONFIGURATION SPACE

Generic ML Algorithms

Random Forest

Linear Regression

...

ML ALGORITHMS

Neural Networks

PRE-PROCESSING

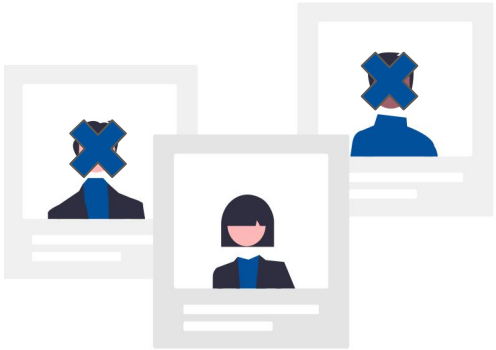
ALGORITHM SELECTION

Given a task choose the best algorithm based on performance prediction.

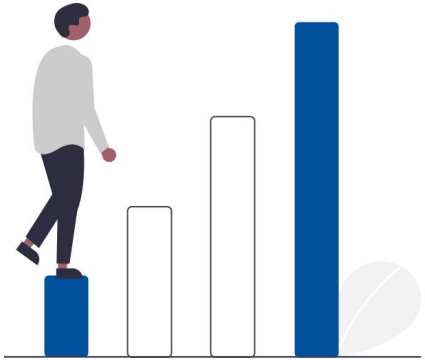
ALGORITHM PORTFOLIO

AutoML aims at

Automating workflows of ML development

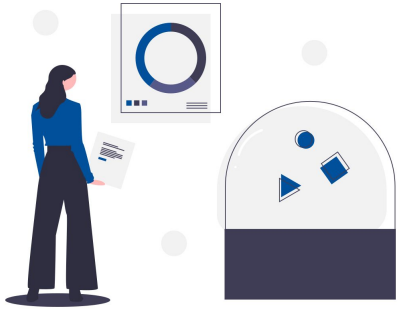


Reduce required expert knowledge

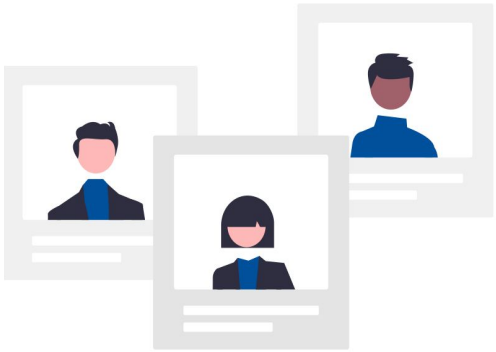


Scaling up

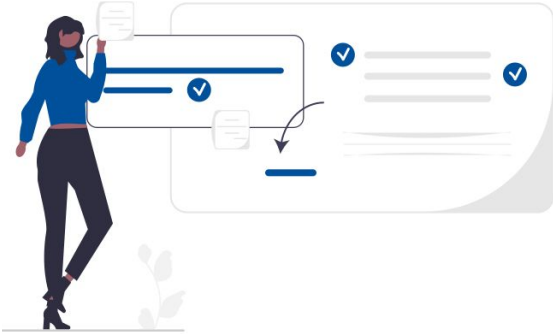
But what if ...



insights into the AutoML black box are crucial?



we have expert knowledge?



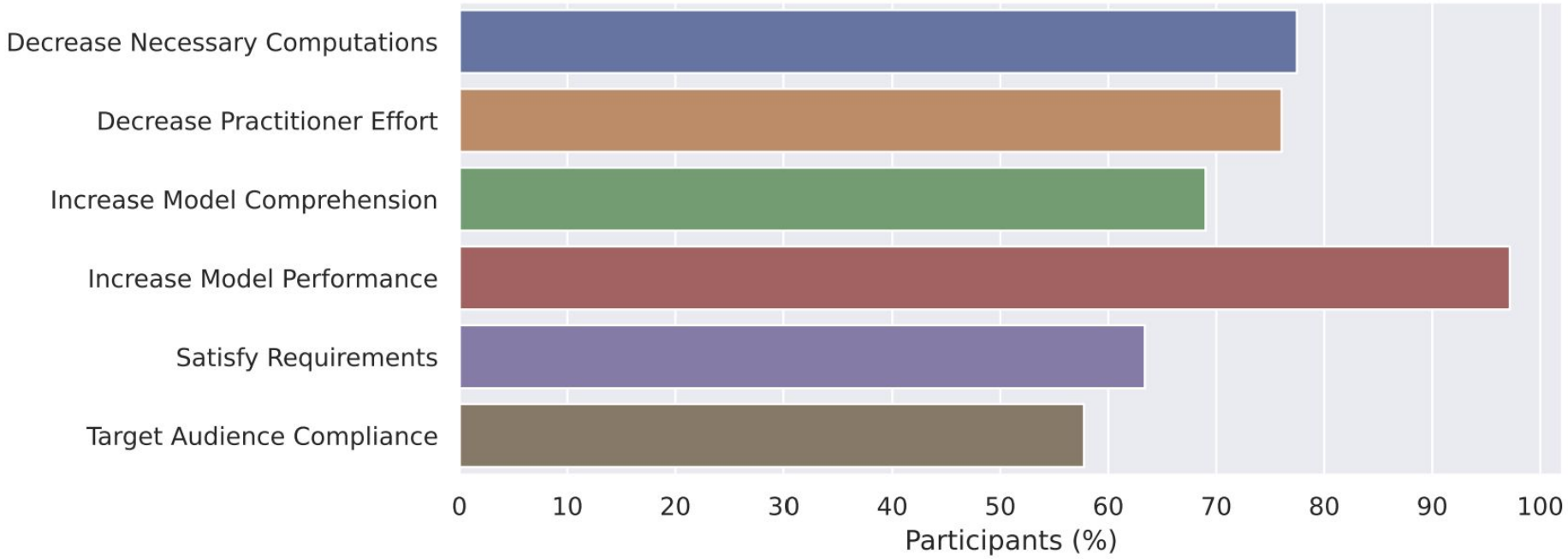
we want to learn from AutoML?

AutoML for Who?

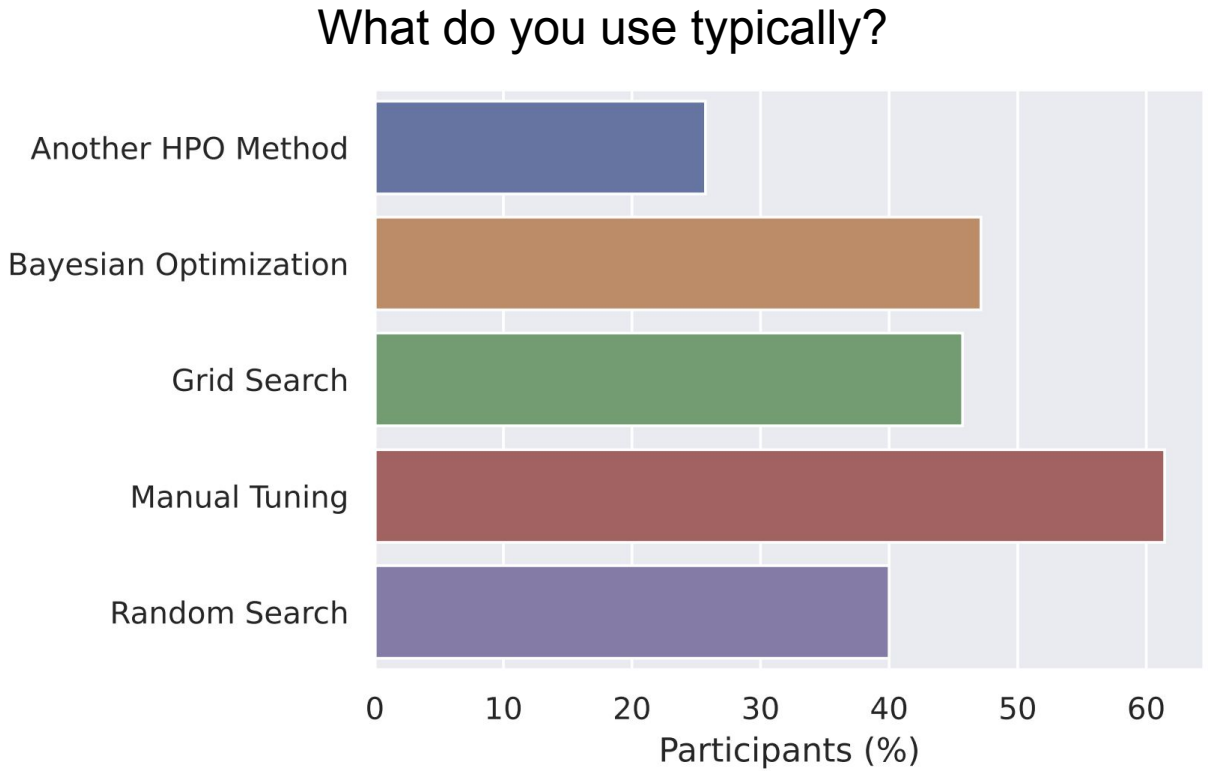
Domain experts with little to no ML expertise

ML practitioners and researchers

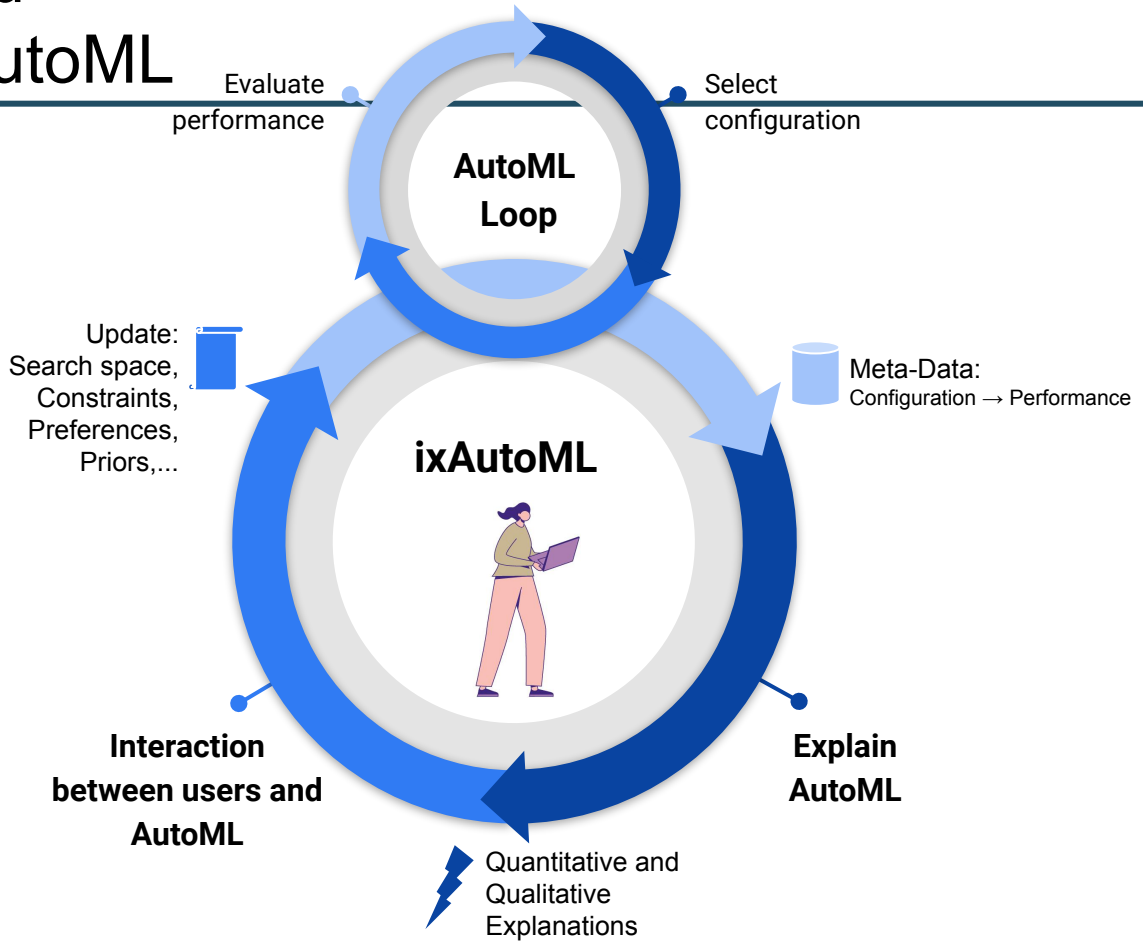
Goals in Using HPO [Hasebrook et al. 2023]



Survey on HPO Use [Hasebrook et al. 2023]



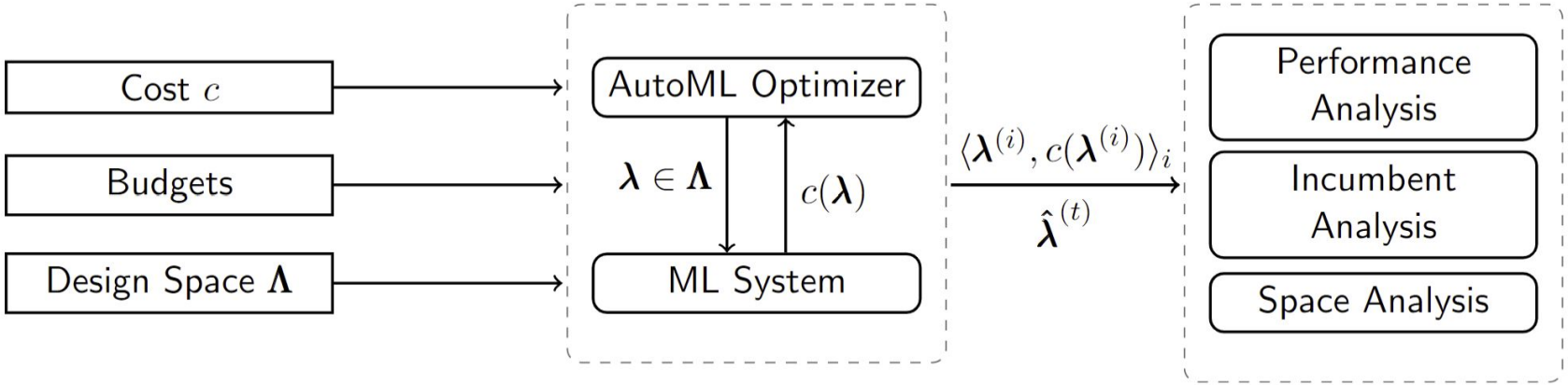
interactive and explainable AutoML



Explainable AutoML

1. **Local Explanations**
2. Global Explanations

Overall Workflow



Ablation Studies

- Ablation studies are quite common in ML papers
  - → shows which of your changes has the largest impact on performance
- Quite easy to implement
  - For your overall system S with (new) features F,
    turn off each f ∈ F and measure its performance

		10min		60min	
		∅	std	∅	std
With Portfolio	Policy selector	**3.58**	0.23	2.47	0.18
	Single best	3.69	0.14	**2.44**	0.12
Without Portfolio	Policy selector	5.63	0.89	3.42	0.32
	Single best	5.37	0.58	3.61	0.61

From Auto-Sklearn 2.0 [Feurer et al. 2022]

Automating Ablations in High-Dimensional Spaces

- Often, a new system adds more than one component
  (→ several binary hyperparameter)
- Even worse: hyperparameter settings might change
  (→ also binary hyperparameter decisions: change or not change!)

  ⇒ leads to a combinatorial space of changes

Example of changes:

- changing DNN optimizer, learning rate, learning rate schedule, depth of DNN, …

- **Problem:** We cannot enumerate all possible changes
- **Solution:** **Greedily** navigate from our starting system to your final system

Example for Automated Greedy Ablation [Fawcett & Hoos 2015]

- Start configuration:
  - a) your initial system that you modified for your research paper
  - b) the default configuration of your system (e.g., ML model)
- End configuration:
  - a) your final system (configuration)
  - b) the result of applying AutoML

$$\begin{aligned} \boldsymbol{\lambda}^{(\text{start})} &= [1, 1, 0, 100] & L_{\text{start}} &= 20\% \\ \boldsymbol{\lambda}^{(\text{end})} &= [0.98, 2.42, 1, 42] & L_{\text{end}} &= 4\% \end{aligned}$$

Example cont'd (i)

$$\boldsymbol{\lambda}^{(\text{start})} = [1, 1, 0, 100] \qquad L_{\text{start}} = 20\%$$
$$\boldsymbol{\lambda}^{(\text{end})} = [0.98, 2.42, 1, 42] \qquad L_{\text{end}} = 4\%$$

1st iteration – iterate through all possible changes from "start" to "end"

$$\boldsymbol{\lambda}^{(1)} = [0.98, 1, 0, 100] \qquad L_1 = 19\%$$
$$\boldsymbol{\lambda}^{(2)} = [1, 2.42, 0, 100] \qquad L_2 = 20\%$$
$$\boldsymbol{\lambda}^{(3)} = [1, 1, 1, 100] \qquad L_3 = 7\%$$
$$\boldsymbol{\lambda}^{(4)} = [1, 1, 0, 42] \qquad L_4 = 16\%$$

The 3rd configuration has the best loss.

→ Fix the 3rd hyperparameter to "1" instead of "0"

Example cont'd (ii)

$$
\begin{aligned}
\boldsymbol{\lambda}^{(\text{start})} &= [1, 1, 0, 100] & L_{\text{start}} &= 20\% \\
\boldsymbol{\lambda}^{(s1)} &= [1, 1, 1, 100] & L &= 7\% \\
\boldsymbol{\lambda}^{(\text{end})} &= [0.98, 2.42, 1, 42] & L_{\text{end}} &= 4\%
\end{aligned}
$$

2nd iteration – iterate through all (three) remaining hyperparameters

$$
\begin{aligned}
\boldsymbol{\lambda}^{(1)} &= [0.98, 1, 1, 100] & L_1 &= 6\% \\
\boldsymbol{\lambda}^{(2)} &= [1, 2.42, 1, 100] & L_2 &= 7\% \\
\boldsymbol{\lambda}^{(3)} &= [1, 1, 1, 42] & L_3 &= 5\%
\end{aligned}
$$

The 3rd configuration has the best loss.

→ Fix the **4th** hyperparameter to "42" instead of "100"

Example cont'd (iii)

$$
\begin{aligned}
\boldsymbol{\lambda}^{(\text{start})} &= [1, 1, 0, 100] & L_{\text{start}} &= 20\% \\
\boldsymbol{\lambda}^{(s1)} &= [1, 1, 1, 100] & L &= 7\% \\
\boldsymbol{\lambda}^{(s2)} &= [1, 1, 1, 42] & L &= 5\% \\
\boldsymbol{\lambda}^{(\text{end})} &= [0.98, 2.42, 1, 42] & L_{\text{end}} &= 4\%
\end{aligned}
$$

3rd iteration – iterate again over all remaining (two) hyperparameters

$$
\begin{aligned}
\boldsymbol{\lambda}^{(1)} &= [0.98, 1, 1, 42] & L_1 &= 4\% \\
\boldsymbol{\lambda}^{(2)} &= [1, 2.42, 1, 42] & L_2 &= 5\%
\end{aligned}
$$

The 1st configuration has the best loss.

→ Fix the **1st** hyperparameter to "0.98" instead of "1"

→ Fix the **2nd** hyperparameter to "2.42" instead of "1" (last remaining)

Example: Final (Greedy) Ablation Path

$$
\begin{aligned}
\boldsymbol{\lambda}^{(\text{start})} &= [1, 1, 0, 100] & L_{\text{start}} &= 20\% \\
\boldsymbol{\lambda}^{(s1)} &= [1, 1, 1, 100] & L &= 7\% \\
\boldsymbol{\lambda}^{(s2)} &= [1, 1, 1, 42] & L &= 5\% \\
\boldsymbol{\lambda}^{(s3)} &= [0.98, 1, 1, 42] & L &= 4\% \\
\boldsymbol{\lambda}^{(s4)} &= [0.98, 2.42, 1, 42] & L &= 4\% \\
\boldsymbol{\lambda}^{(\text{end})} &= [0.98, 2.42, 1, 42] & L_{\text{end}} &= 4\%
\end{aligned}
$$

Greedy Ablation Algorithm

Algorithm Greedy Ablation

Input : Algorithm $\mathcal{A}$ with configuration space $\mathbf{\Lambda}$, start configuration $\boldsymbol{\lambda}^{(\text{start})}$, end configuration $\boldsymbol{\lambda}^{(\text{end})}$, cost metric c

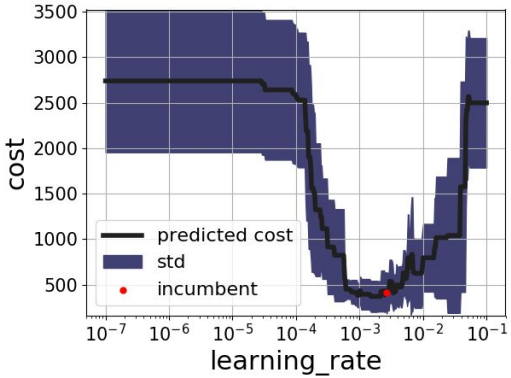
$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda}^{(\text{start})}$;
$P \leftarrow []$;
foreach $t \in \{1 \dots |\mathbf{\Lambda}|\}$ **do**
  foreach $\delta \in \Delta(\boldsymbol{\lambda}, \boldsymbol{\lambda}^{(end)})$ **do**
    $\boldsymbol{\lambda}'_\delta \leftarrow$ apply δ to $\boldsymbol{\lambda}$;
    evaluate $c(\boldsymbol{\lambda}'_\delta)$;
  Determine most important change $\delta^* \in \arg\min_{\delta \in \Delta(\boldsymbol{\lambda}, \boldsymbol{\lambda}^{(\text{end})})} c(\boldsymbol{\lambda}_\delta)$;
  $\boldsymbol{\lambda} \leftarrow$ apply δ^* to $\boldsymbol{\lambda}$;
  P.append(δ^*);
return *Ablation path* P

Final Remarks on Greedy Ablation

- Even this greedy ablation requires $O(n^2)$ steps
  - each step corresponds to training and evaluating a ML model
  - → still fairly expensive process
- We can also speedup that up by using surrogate models [Biedenkapp et al. 2017]
- Common observations:
  - Some hyperparameters might not matter
  - Often only a few of the hyperparameters have an big impact
  - You have plateaus in your ablation path because of interaction effects

LPI: Local Parameter Importance [Biedenkapp et al. 2018]

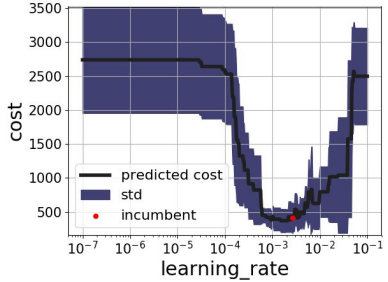
- Typical question of users:
  How would the performance change if we change **one** hyperparameter?
  - E.g., if we would change the learning rate, can we get even better results?
- Problem: Running full study is often too expensive
  - Each run of an ML-system is potential expensive
- Key Ideas:
  - Re-use probabilistic models as trained in BO
  - Plot performance change around incumbent configuration along each dimension
    - incumbent configuration: the configuration finally returned by AutoML



Computation of LPI

$$\mathrm{VAR}_{\boldsymbol{\lambda}}(i) = \sum_{v \in \boldsymbol{\Lambda}_i} \left(\mathbb{E}_{v \sim \boldsymbol{\Lambda}_i}[L(\boldsymbol{\lambda})] - L(\boldsymbol{\lambda}[\boldsymbol{\lambda}_i := v])\right)^2$$

$$\mathrm{LPI}(i \mid \boldsymbol{\lambda}) = \frac{VAR_{\boldsymbol{\lambda}}(i)}{\sum_j VAR_{\boldsymbol{\lambda}}(j)}$$



1. Compute ICE (Individual Conditional Expectation) curve at the incumbent
2. Compute the variance along Dimension i
3. Normalize variance for each Dimension i by variation along all dimensions

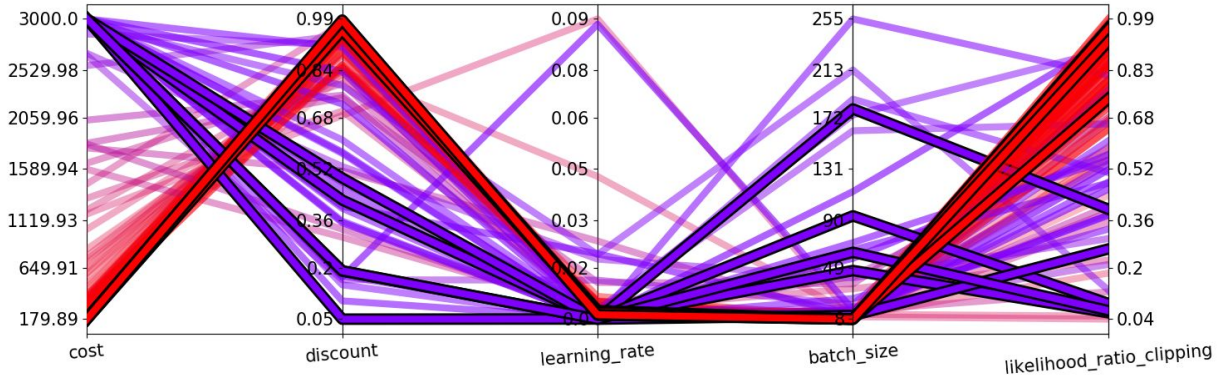
Note: Only one dimension at a time is changed; all others are fixed to the incumbent value

Kahoot Quiz I

Explainable AutoML

1. Local Explanations
2. **Global Explanations**

Parallel Coordinate Plots



PPO on cartpole; Source: [Lindauer et al. 2019]

- Visualization of sampling in high-dimensional hyperparameter spaces [Golovin et al. 2017]
- Allows to identify:
  - optimization focus of AutoML optimiziers
  - well-performing combination of settings
  - Interaction effects between settings (to some degree)
- Rather qualitative, less quantitative analysis
- Follow up: Conditional parallel coordinate plots [Weidele et al. 2019]

fANOVA

- Key idea: What is the importance of a hyperparameter by marginalizing over all other hyperparameter effects?
- Key Insight: We can use a surrogate model to compute these effects

fANOVA [Sobobl. 1993]

Write performance predictions as a sum of components:

$$\hat{y}(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_n) = \hat{f}_0 + \sum_{i=1}^{n} \hat{f}_i(\boldsymbol{\lambda}_i) + \sum_{i \neq j} \hat{f}_{ij}(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_j) + \ldots$$

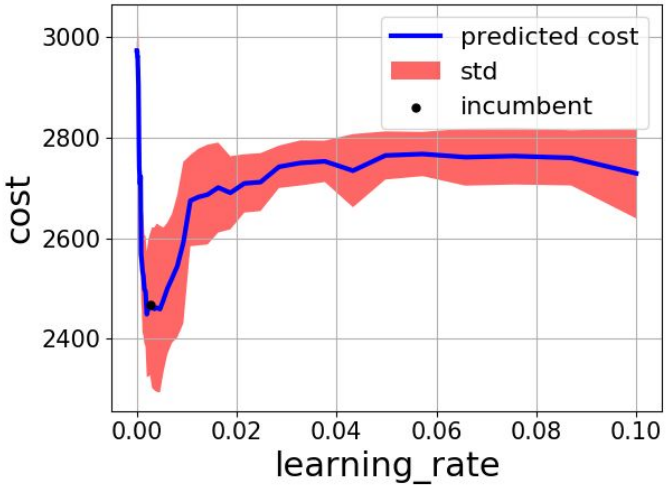
$$\hat{y}(\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_n) = \text{average response} + \text{main effects} + \text{2-D interaction effects} + \text{higher order effects}$$

Variance Decomposition

$$\mathrm{V} = \frac{1}{||\boldsymbol{\Lambda}||} \int_{\boldsymbol{\lambda}_1} \ldots \int_{\boldsymbol{\lambda}_n} [(\hat{y}(\boldsymbol{\lambda}) - \hat{f}_0)^2] d\boldsymbol{\lambda}_1 \ldots d\boldsymbol{\lambda}_n$$

fANOVA

- The fANOVA and variance decomposition can be done efficiently in linear time if the surrogate model is a random forest [Hutter et al. 2014]



- predicted cost is marginalized over all other hyperparameter effects
- Warning: The optimum on these curves does not have to be the global optimum across all hyperparameters

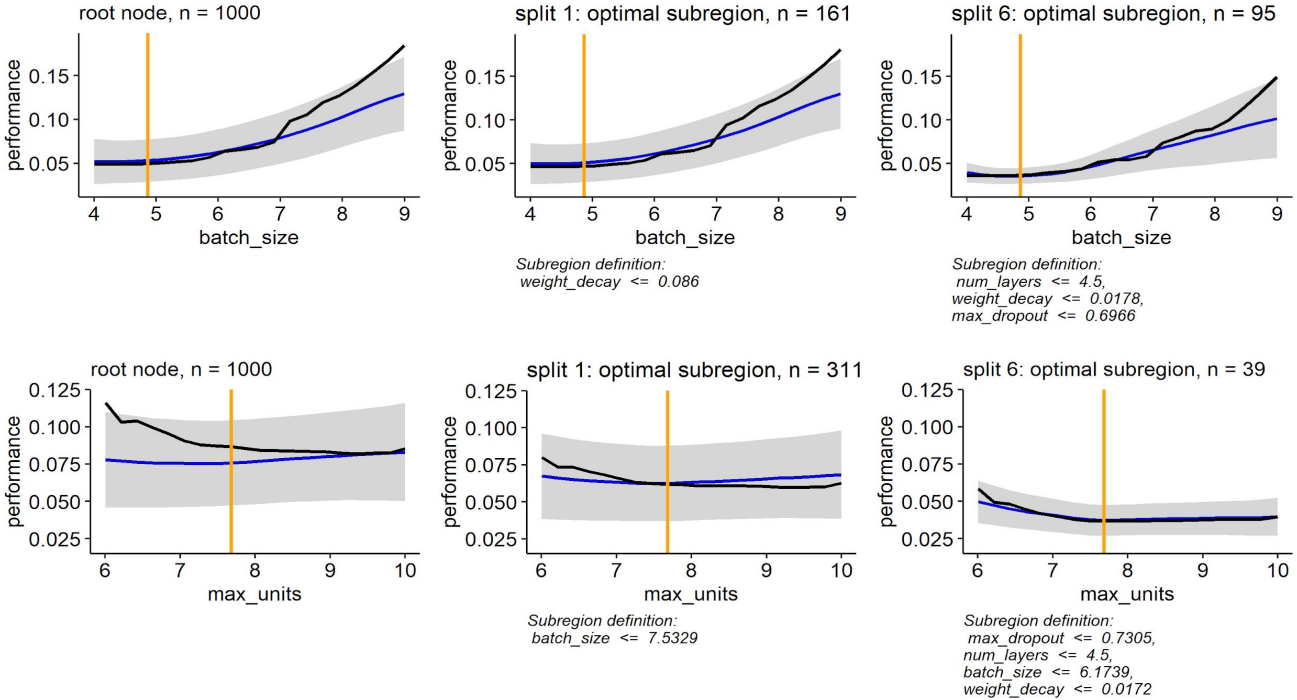
fANOVA Example

Table: Exemplary analysis of PPO on cartpole

Hyperparameter	Explained Variance
Discount rate	19.3 %
Batch size	15.7 %
Learning rate	3.7 %
Likelihood ration clipping	3.4%
...	
discount rate & batch size	10.4%
discount rate & likelihood ration clipping	4.4%
...	

Explaining HPO with PDPs [Moosbauer et al. NeurIPS'21]

Ground truth
PDP
incumbent



PDP: Partial Dependence Plots

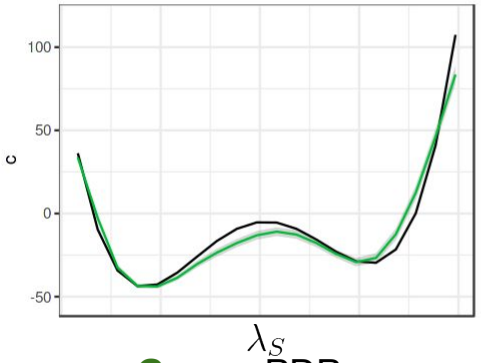
For, a subset S of the hyperparameters, the partial dependence function is:

$$c_S(\lambda_S) := \mathbb{E}_{\lambda_C}\left[c(\lambda)\right] = \int_{\Lambda_C} c(\lambda_S, \lambda_C) d\mathbb{P}(\lambda_C)$$

and can be approximated by Monte-Carlo integration on a surrogate model:

$$\hat{c}_S\left(\lambda_S\right) = \frac{1}{n} \sum_{i=1}^{n} \hat{m}\left(\lambda_S, \lambda_C^{(i)}\right)$$

where $\left(\lambda_C^{(i)}\right)_{i=1,\ldots,n} \sim \mathbb{P}(\lambda_C)$ and λ_S for a of grid points.



Green: PDP
Black: Ground truth

→ Average of ICE curves.

Quantifying Uncertainties in PDPs

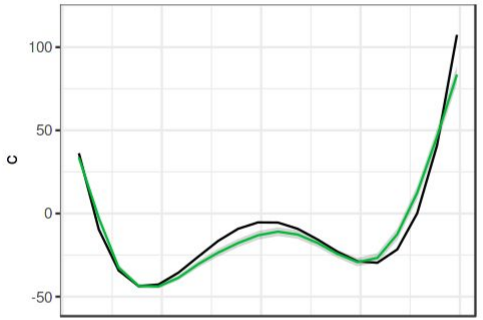
$$
\begin{aligned}
&\hat{s}_S^2(\lambda_S) \\
&= \mathbb{V}_{\hat{c}}\left[\hat{c}_S\left(\lambda_S\right)\right] \\
&= \mathbb{V}_{\hat{c}}\left[\frac{1}{n}\sum_{i=1}^{n}\hat{c}\left(\lambda_S, \lambda_C^{(i)}\right)\right] \\
&= \frac{1}{n^2} 1^\top \hat{K}\left(\lambda_S\right)\ 1.
\end{aligned}
$$

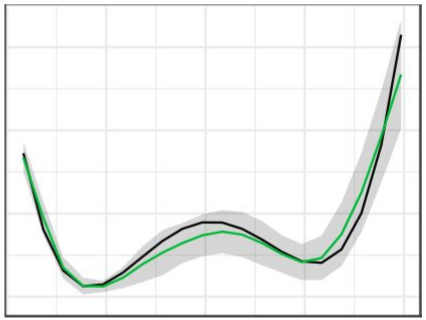
→ requires a kernel correctly specifying the covariance structure (e.g., GPs).

Approximation:

$$
\hat{s}_S^2\left(\lambda_S\right) \approx \frac{1}{n}\sum_{i=1}^{n}\hat{K}\left(\lambda_S\right)_{i,i}
$$

→ Model-agnostic (local) approximation





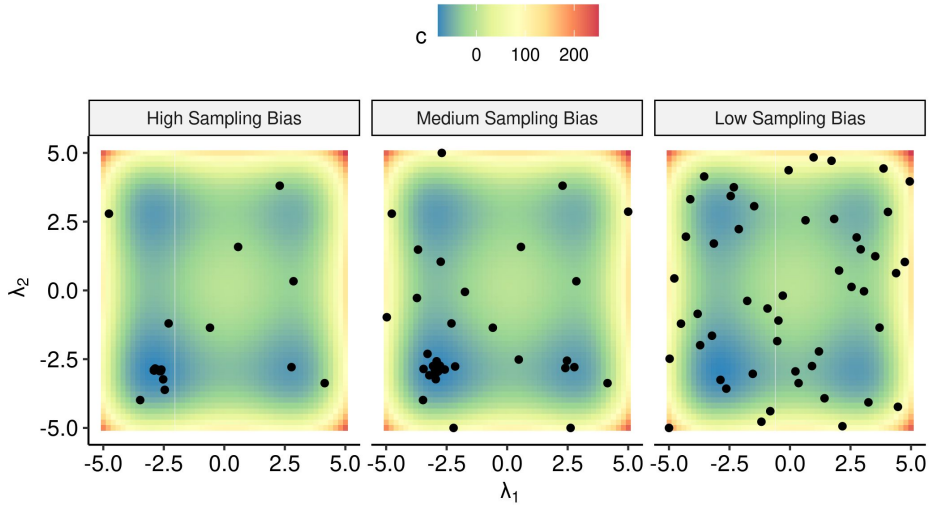
Ground truth
PDP
Uncertainty

Problem: Biased Sampling

- PDPs assume that the data is independently, identically distributed (iid)
- Obviously not the case for efficient AutoML tools with a focus on high-performance regions

- Example:
  - BO with GPs and LCB
  - Different exploration rate for LCB to show different sampling bias

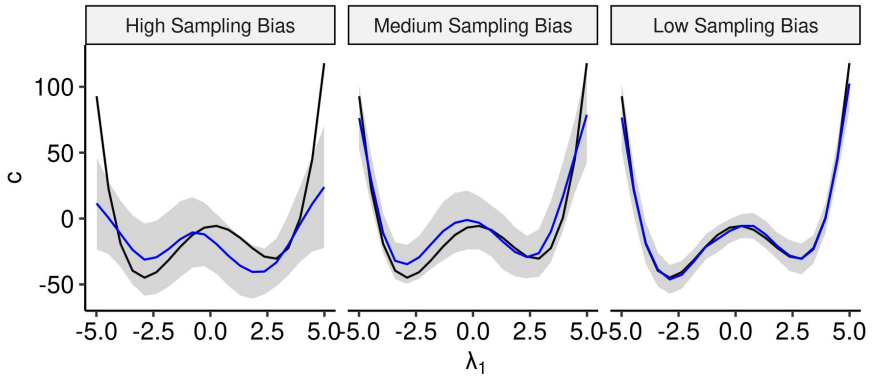
$$\mathrm{LCB}(\lambda) = \mu(\lambda) + \beta \cdot \sigma(\lambda)$$

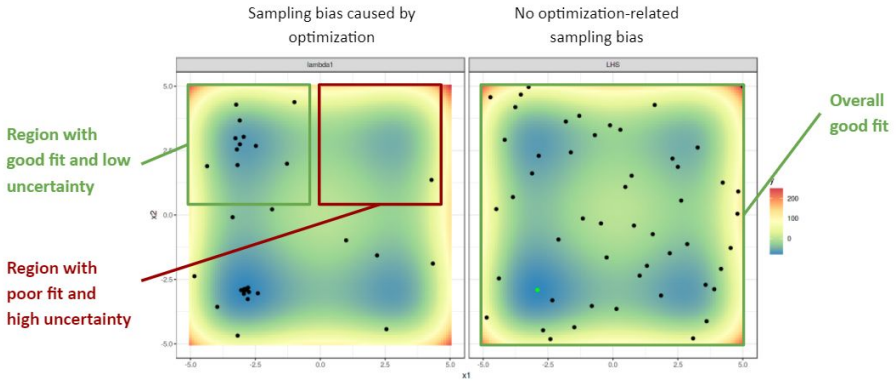


Impact of the Sampling Bias

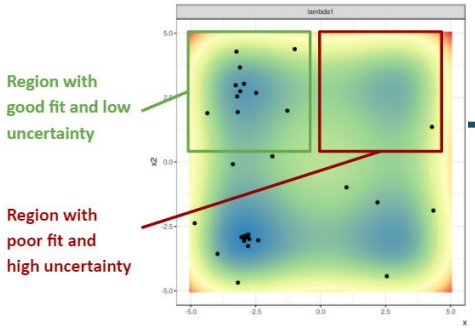
- Simply using all observations from AutoML tools might lead to misleading PDPs
- Uncertainty estimates help to quantify the poor fits

→ Sampling bias is wanted and a solution to this problem should not change the sampling behavior





Partitioning of Space



Partition space to obtain interpretable subspaces $\mathcal{N}'$.

Uncertainty variation across all ICE estimates:

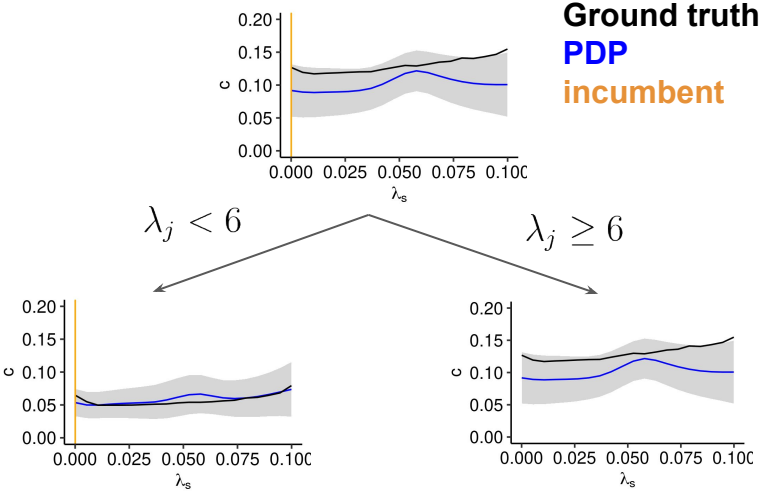
$$L\left(\lambda_S, \mathcal{N}'\right) = \sum\nolimits_{i \in \mathcal{N}} \left(\hat{s}^2\left(\lambda_S, \lambda_C^{(i)}\right) - \hat{s}^2_{S|\mathcal{N}'}\left(\lambda_S\right)\right)^2$$

$$\hat{s}^2_{S|\mathcal{N}'}\left(\lambda_S\right) := \frac{1}{|\mathcal{N}'|} \sum_{i \in \mathcal{N}'} \hat{s}^2\left(\lambda_S, \lambda_C^{(i)}\right)$$

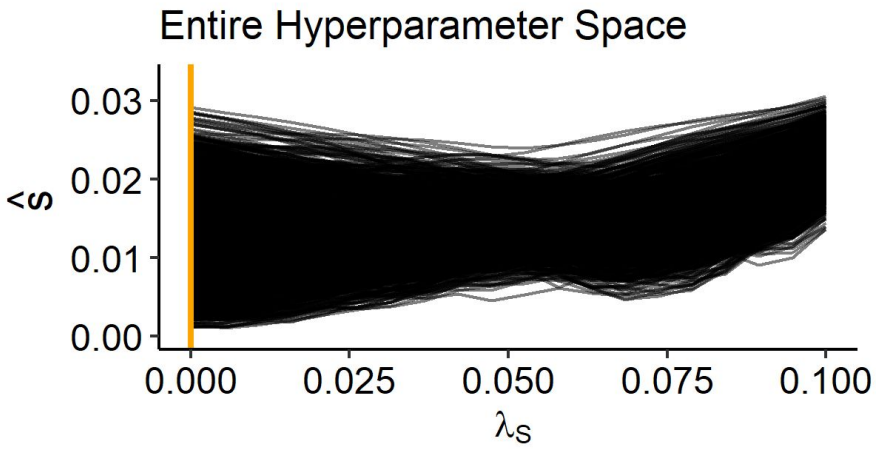
→ **Uncertainty structure of ICE curves should maximally agree**

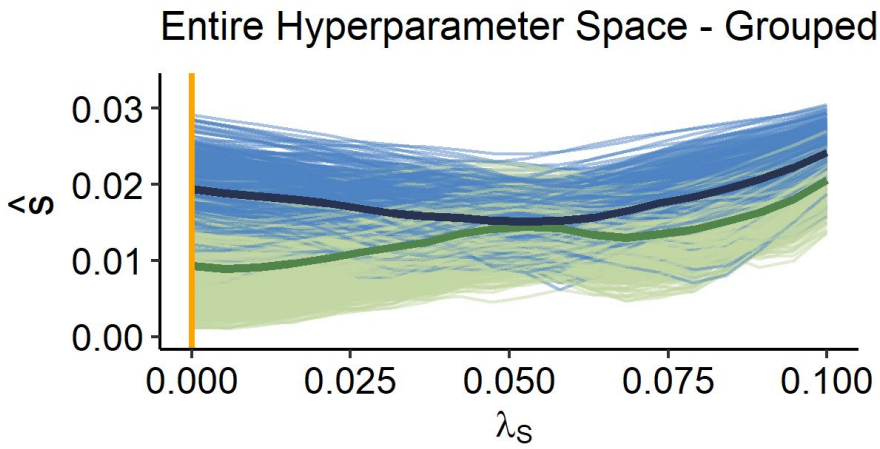
Split Loss = Aggregation over all grid points:

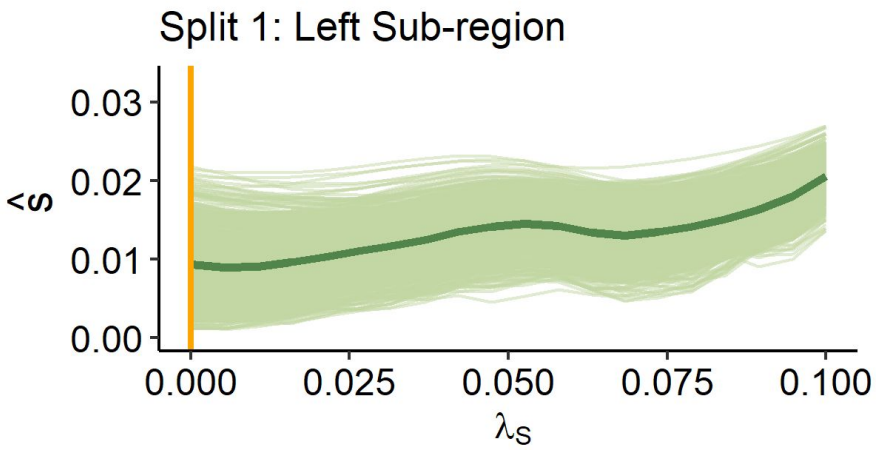
$$\mathcal{R}_{L2}(\mathcal{N}') = \sum\nolimits_{g=1}^{G} L(\lambda_S^{(g)}, \mathcal{N}')$$

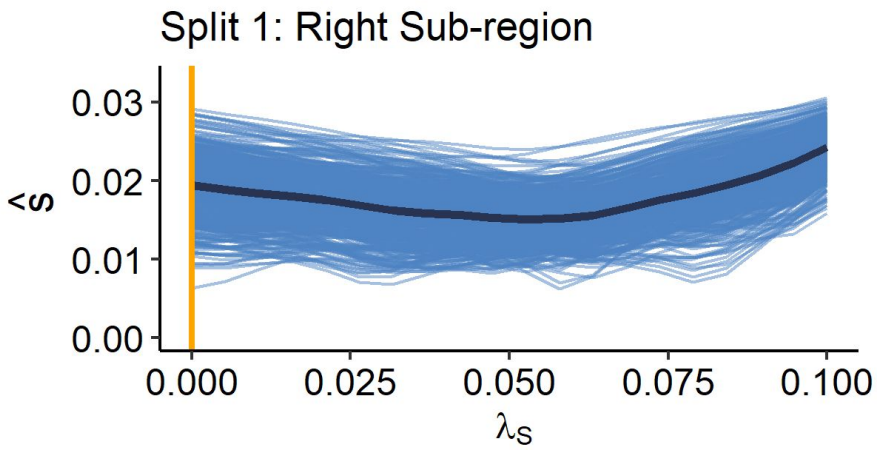


Note (i): Partition only along the marginalized dimensions

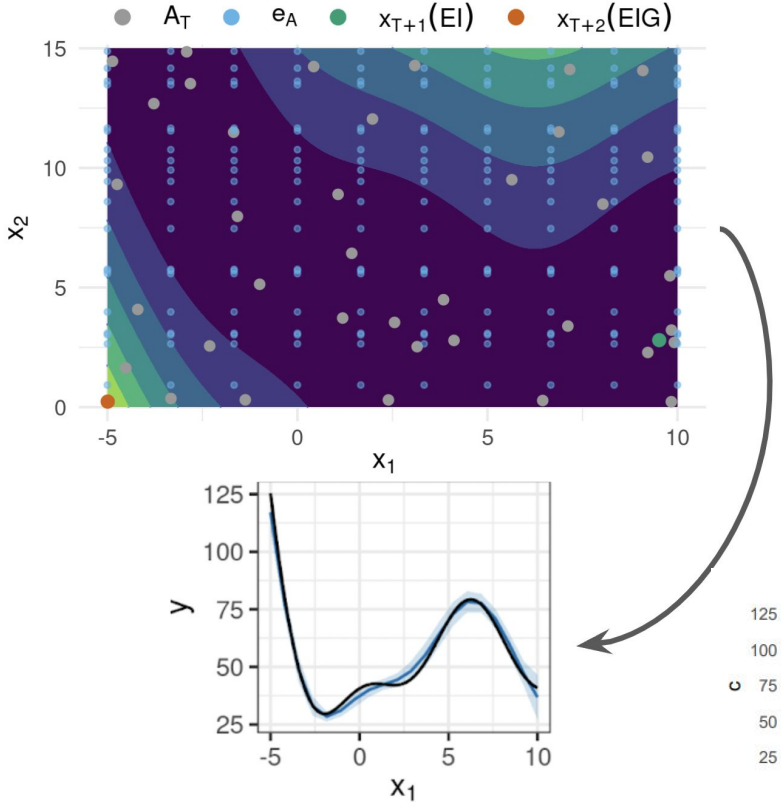

Entire Hyperparameter Space
0.03
0.02
0.01
0.00
0.000
0.025
0.050
0.075
0.100
λ_S


Entire Hyperparameter Space - Grouped
0.03
0.02
0.01
0.00
0.000
0.025
0.050
0.075
0.100
λ_S

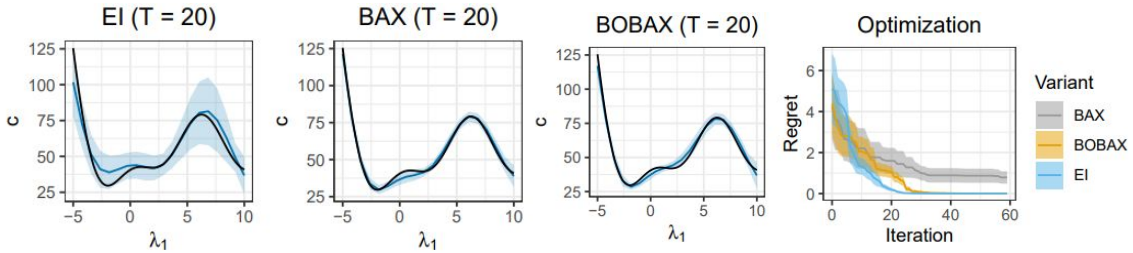

Split 1: Left Sub-region
0.03
0.02
0.01
0.00
0.000
0.025
0.050
0.075
0.100
λ_S


Split 1: Right Sub-region
0.03
0.02
0.01
0.00
0.000
0.025
0.050
0.075
0.100
λ_S

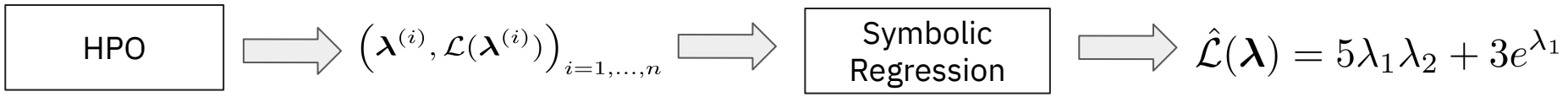
Better PDPs with BO-BAX [Moosbauer et al. 2022]



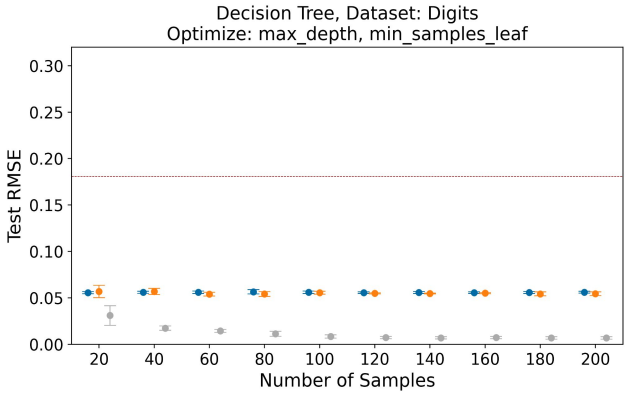
- Bayesian Algorithm Execution [Neiswanger et al. 2021] allows to compute the best next point to improve quality of PDPs
- Interleaving BO and BAX for PDPs leads to
  (i) nearly the same anytime performance than standard BO
  (ii) much better PDPs



Symbolic Regression for HPO [Segel et al. 2023]

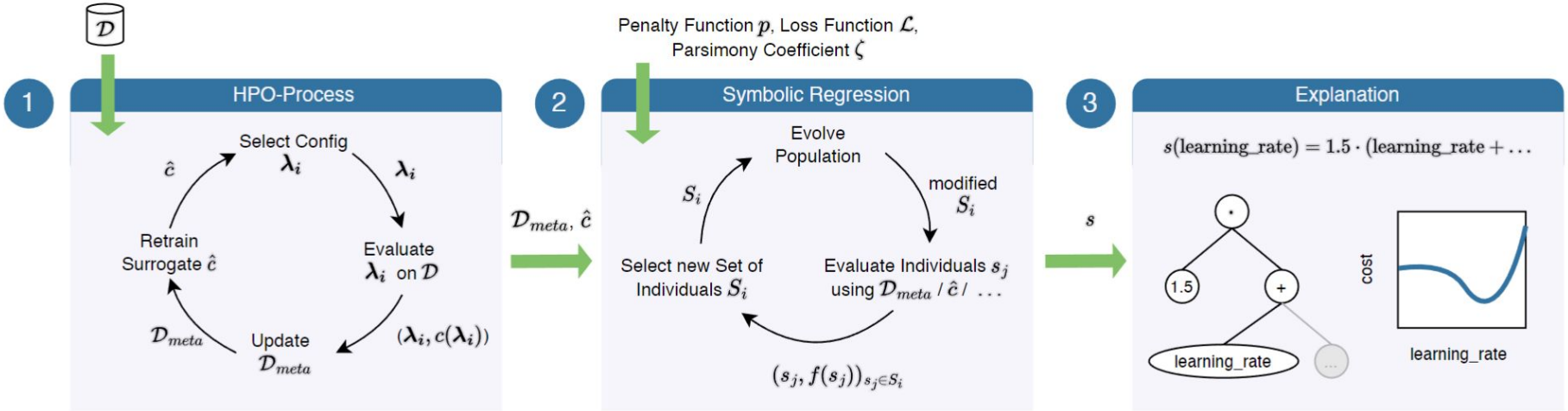


- Pushak and Hoos [2022] provided evidence for fairly benign HPO landscape
  - e.g. unimodal and little interactions
- ⇒ So, it should be feasible to learn interpretable symbolic regression models

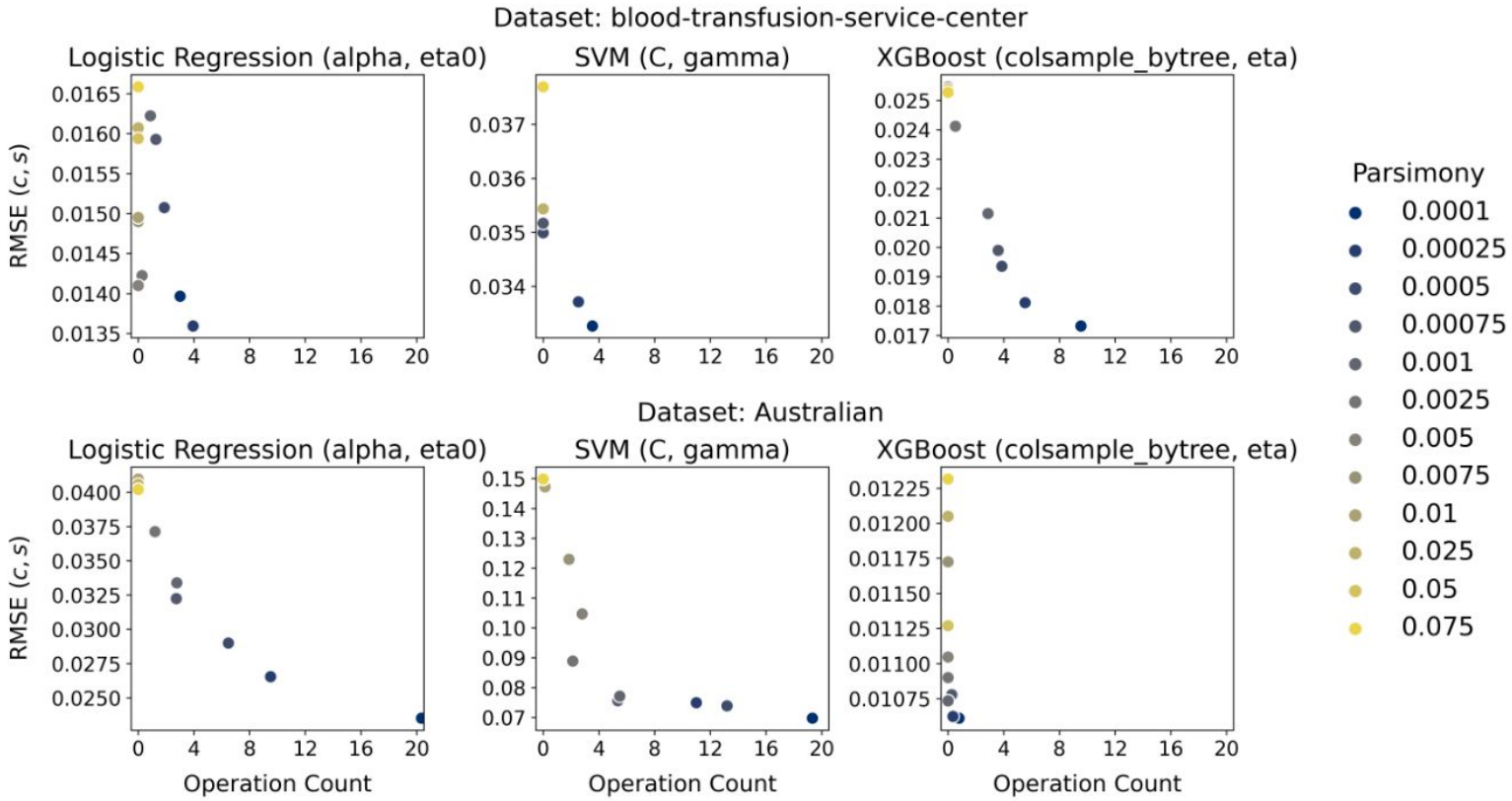


$$\mathcal{L}_{\text{Loss}} = \frac{0.845}{\sqrt{\text{max\_depth}}}$$

Symbolic Regression for HPO [Segel et al. 2023]



Complexity of Symbolic Regression for HPO [Segel et al. 2023]



Reducing Complexity of Symbolic Regression for HPO

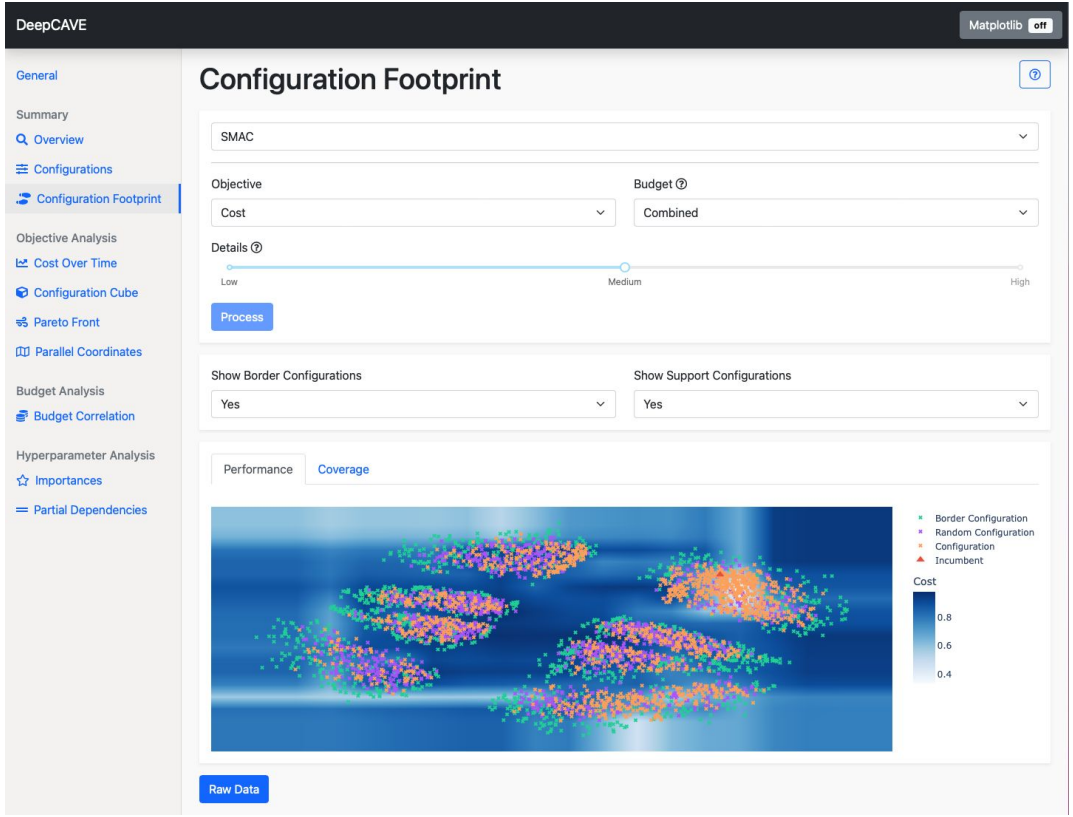
1. Apply fANOVA (or some other technique) to identify the *N* most important hyperparameters

2. Marginalize over the dropped hyperparamters
   a. E.g. similar to PDPs

3. Apply symbolic regression to it

4. Use elbow point if complexity of symbolic function is still too high

DeepCAVE: A Package to Analyse AutoML [Sass et al. 2022]



/automl/deepcave

Available:

- Summary of experimental setup
- Objective Analysis
- Budget analysis in multi-fidelity settings
- Hyperparameter analysis

⇒ All of our explanation techniques will be added to DeepCAVE!

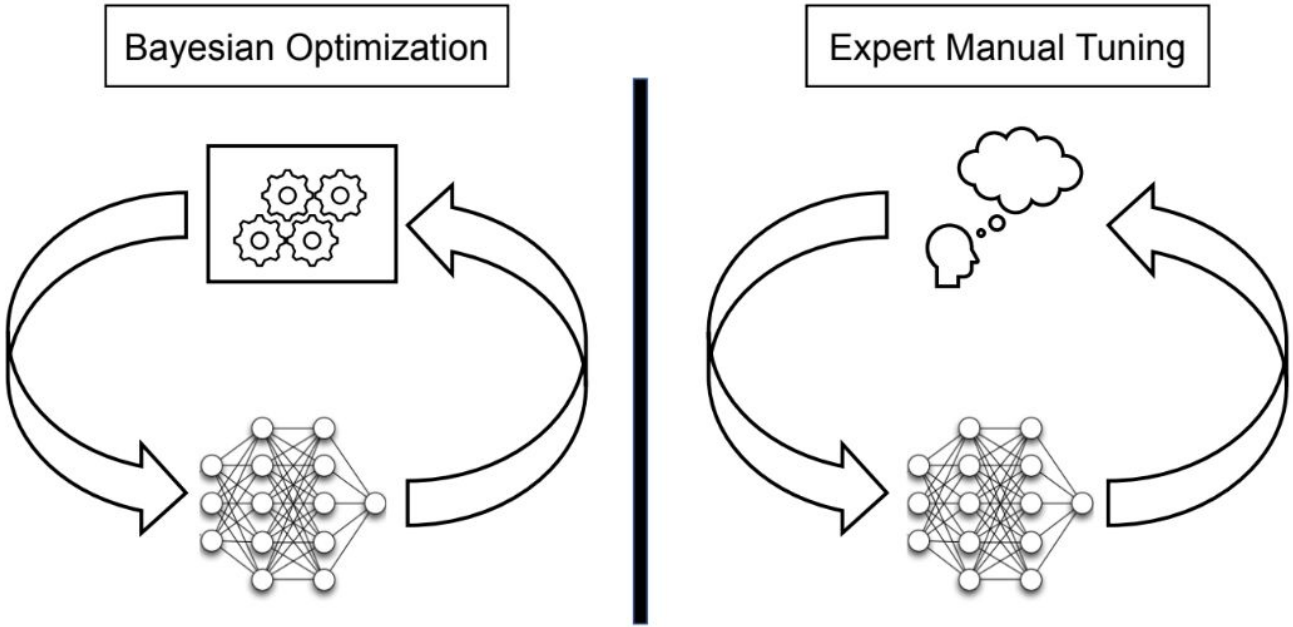
More Interpretability of AutoML?

- In principle, we could apply (nearly) all techniques from interpretable machine learning (iML) to surrogate models of AutoML optimizers
  - E.g., LIME, Anchors, SHAP, …
  - All of them face the same problem of underlying sampling bias induced by AutoML optimizers
- Challenges:
  - When to apply which interpretability method? What is interesting for users?
  - Assuming an AutoML practitioner with little to no expertise in ML, which explanations are they able to understand?
  - Assuming that AutoML landscapes changes dependent on the dataset at hand, what kind of universal take-aways can we learn from that?

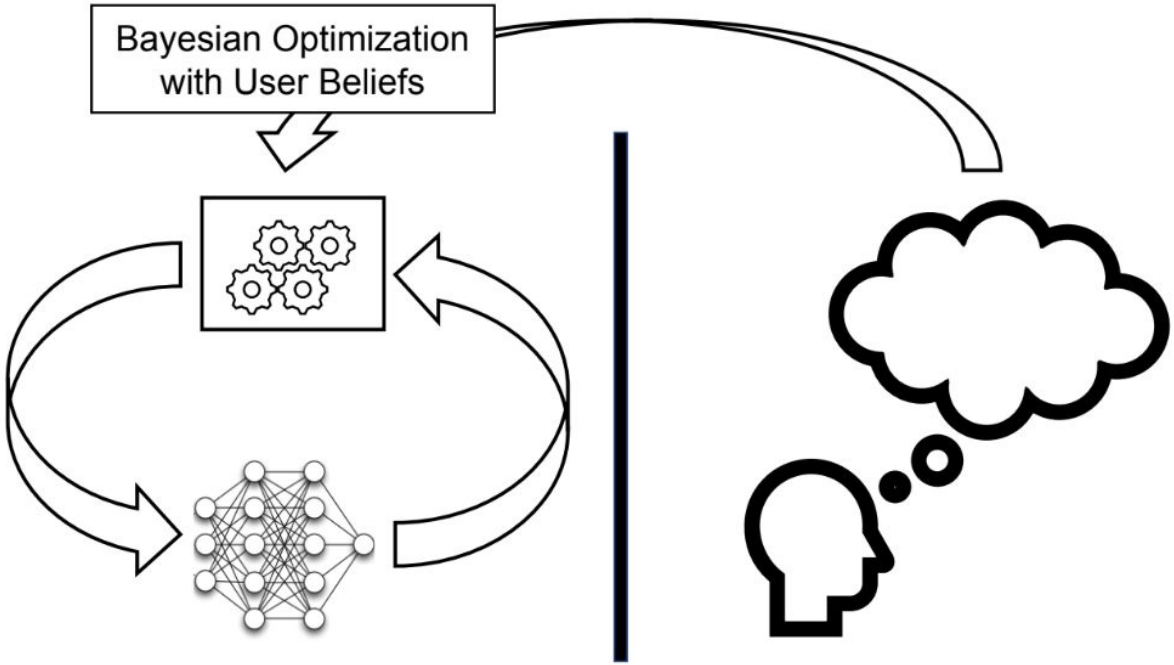
Kahoot Quiz II

Interactive AutoML

Considering Expert Knowledge in HPO [Souza et al. ECML'21, Hvarfner et al. ICLR'22]



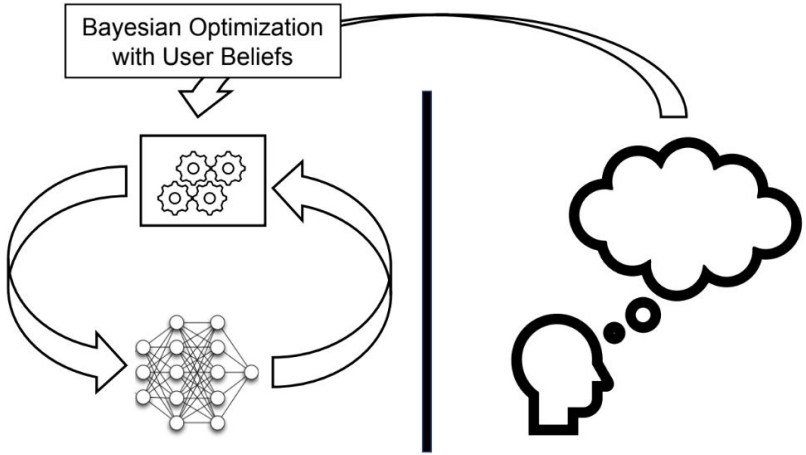
Considering Expert Knowledge in HPO[Souza et al. ECML'21, Hvarfner et al. ICLR'22]



Types of Expert Knowledge

1. Shape of the underlying function, e.g., smoothness
2. Interaction effects of hyperparameters
3. Importance of hyperparameters
4. **Areas of (presumable) well-performing hyperparameter configurations**

π-BO [Hvarfner et al. ICLR'22]



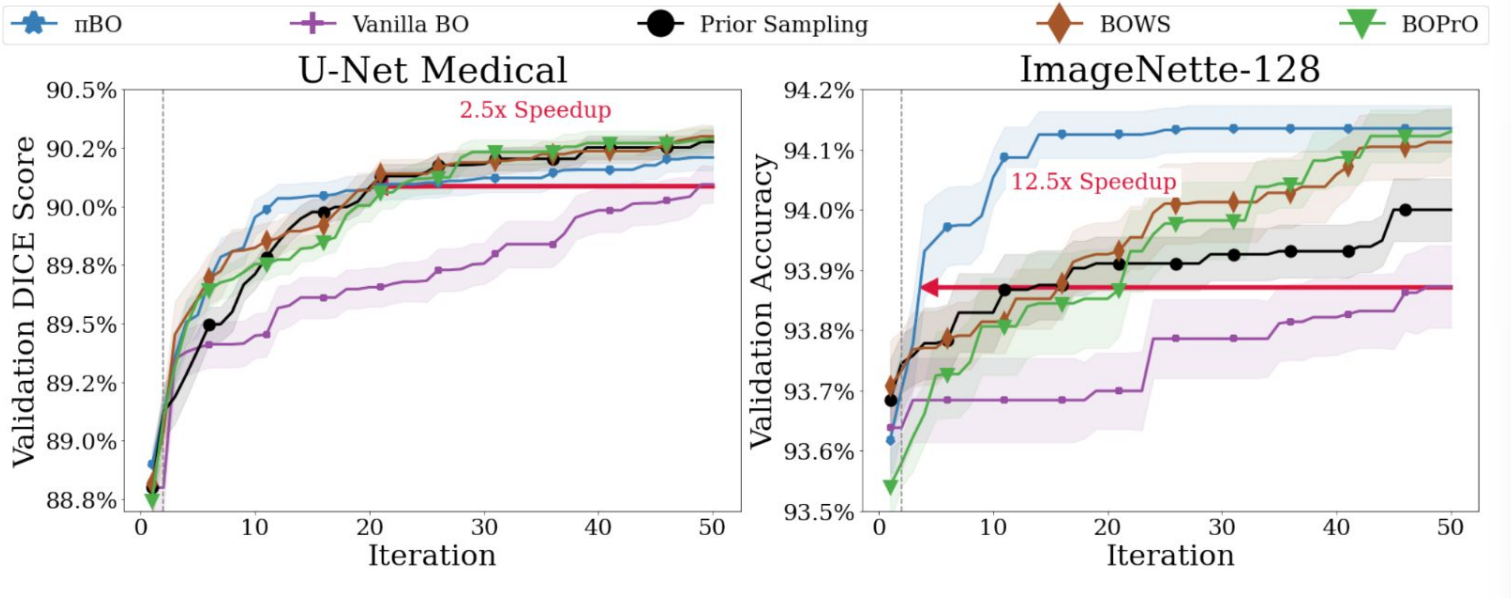
$$\lambda_n \in \text{argmax}_{\lambda \in \Lambda} \alpha(\lambda, \mathcal{D}_n) \pi(\lambda)^{\beta/n}$$

Acquisition Function

User Prior

Speed of forgetting user prior

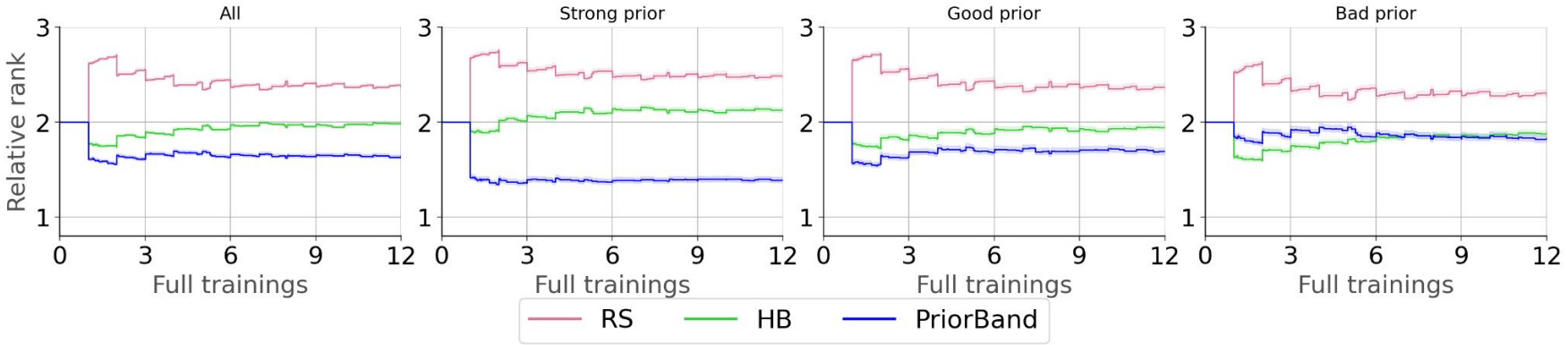
π-BO: Results [Hvarfner et al. ICLR'22]



- Uses expert knowledge to speed up Bayesian Optimization
- Robust also against wrong believes
- Substantially speeds up AutoML

PriorBand: Expert Priors + Hyperband [Malik et al. 2023]

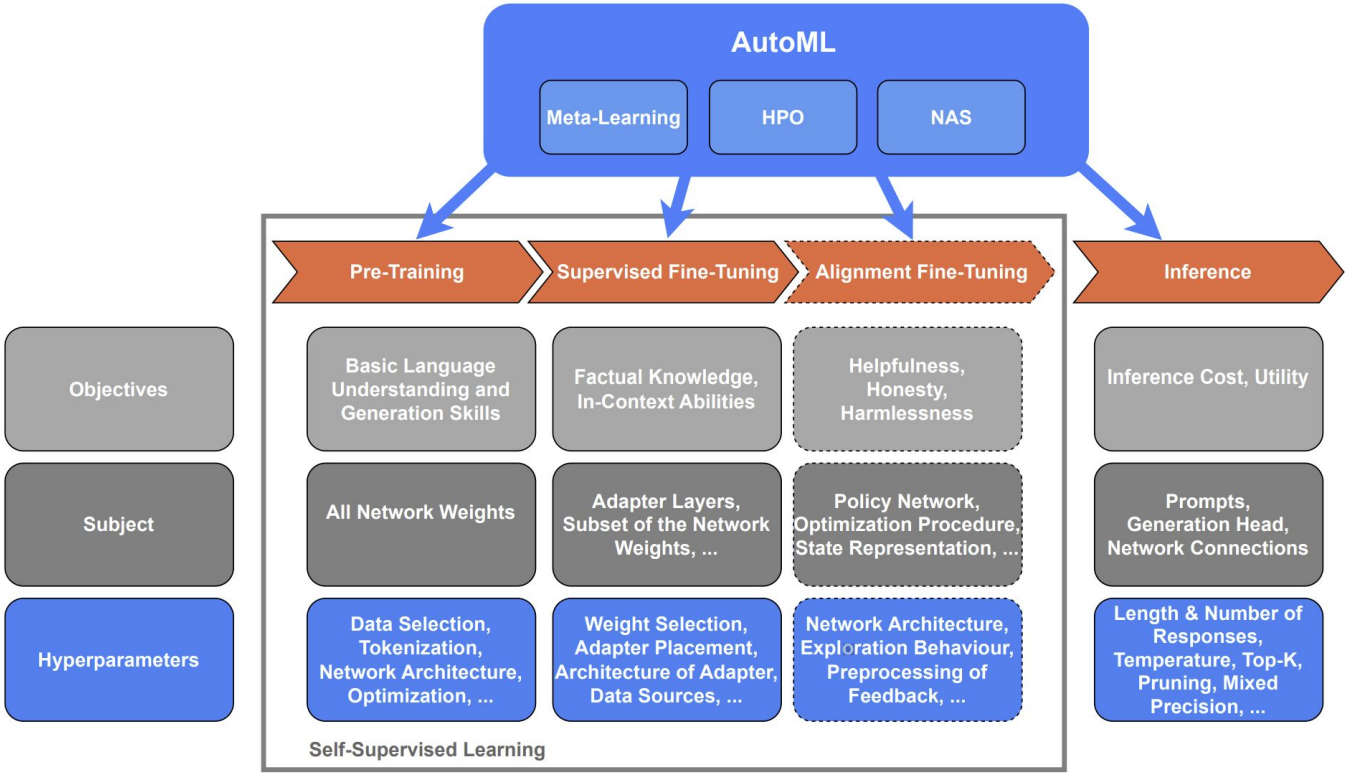
- How can we run HPO under 10 full model trainings?
- Four main insights
  - 0. Multi-fidelity and prior-guided optimization is key for efficiency
  - 1. Trusting priors more at higher fidelities
  - 2. Incumbent-based sampling of new configurations
  - 3. Ensembling of sampling policies



AutoML in the age of LLMs?

[Tornede et al. 2023]

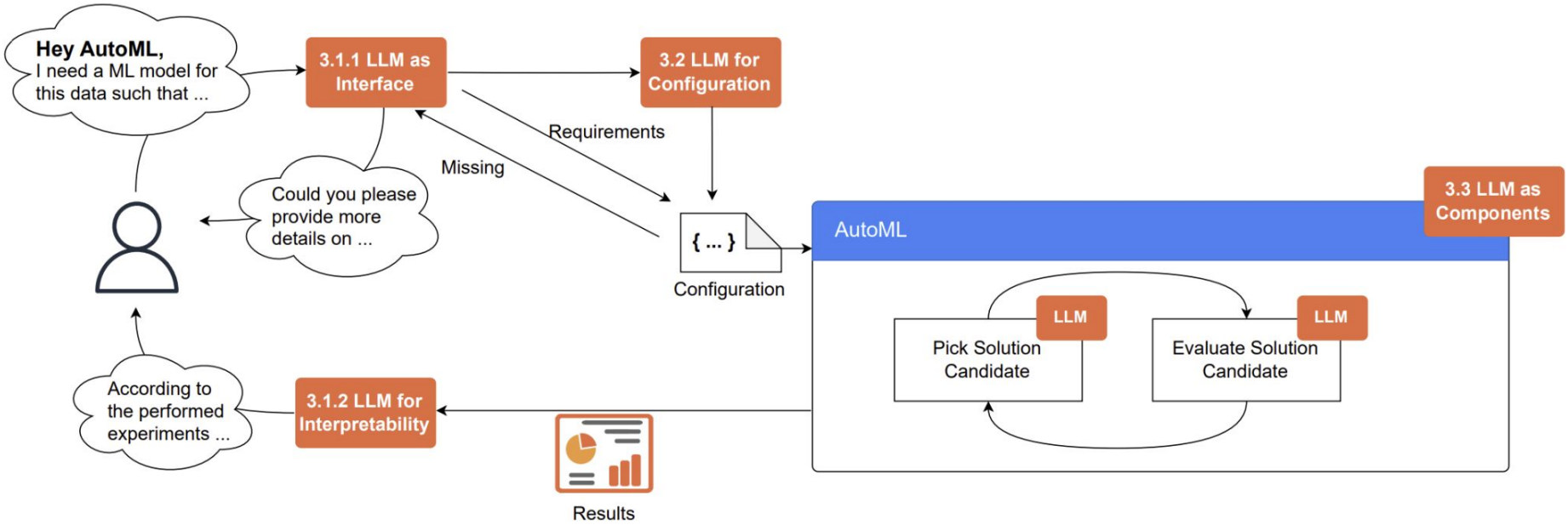
AutoML for LLMs is still a big challenge



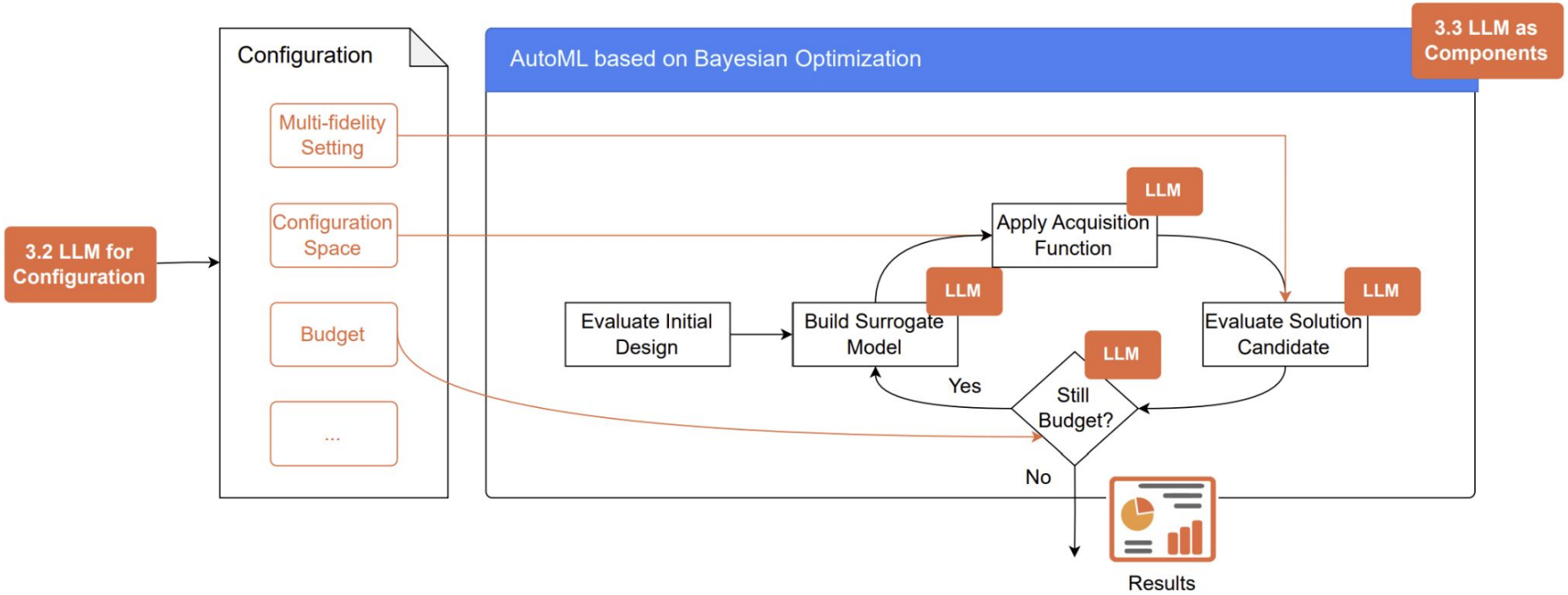
Challenges of Applying AutoML to LLMs

1. Pre-Training of LLMs is **very expensive**
   - scaling laws, multi-fidelity and human-priors might help
2. Joint optimization of the entire LLM pipeline is **not feasible** without knowing the downstream task (distribution)
3. Neural architecture search is still **not powerful** enough to find fully novel architectures
   - see lecture on NAS
4. There is **no single metric** to be optimized sinced different metrics are used for the different stages of training an LLM
5. Combination of several learning paradigms lead to **very complex search spaces**

Opportunities for interaction with AutoML via LLMs



Deeply Integrating LLMs into AutoML

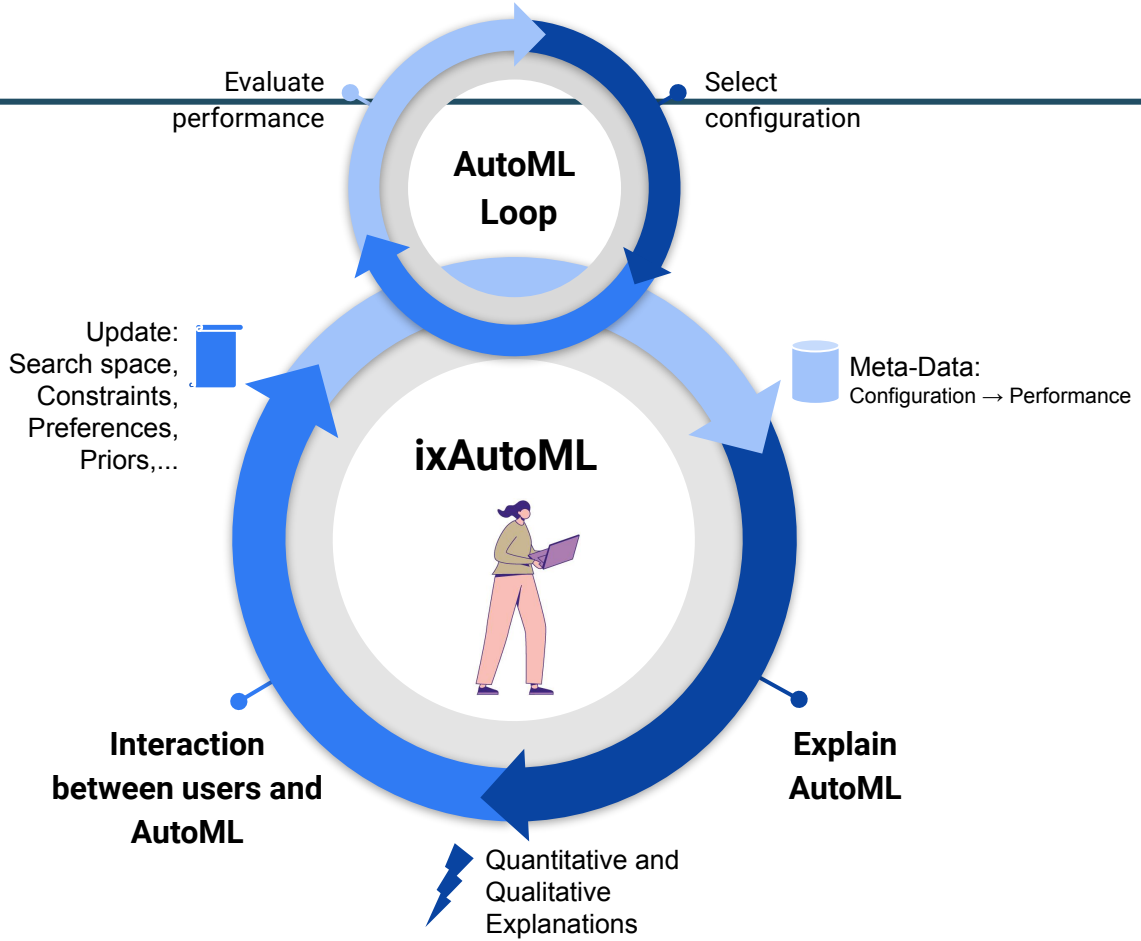


Conclusion

Take-Home Insights

- Who is going to run AutoML?
  - Different user groups have different expectations
  - How to increase trust into the results of AutoML?
- What would you like to learn from running AutoML?
  - importance of hyperparameters?
  - effects of hyperparameters?
- What kind of expert knowledge is available to you and AutoML users?
  - Do you have an expectation what should perform well?
- LLMs might change how we use (Auto)ML in the future

ixAutoML


Evaluate performance
Select configuration
AutoML Loop
Update: Search space, Constraints, Preferences, Priors,...
Meta-Data: Configuration → Performance
ixAutoML
Interaction between users and AutoML
Explain AutoML
Quantitative and Qualitative Explanations

Kahoot Quiz III

Thanks.
See you tomorrow!