

# Speedup Techniques for Hyperparameter Optimization

## Overview

Bernd Bischl    Frank Hutter    Lars Kotthoff  
Marius Lindauer    Joaquin Vanschoren

# Beyond Black-box Optimization

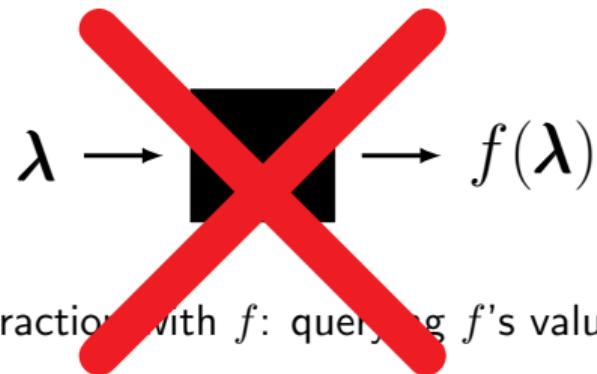
Recall general blackbox optimization:

$$\lambda \rightarrow \boxed{\quad} \rightarrow f(\lambda)$$

Only mode of interaction with  $f$ : querying  $f$ 's value at a given  $\lambda$

# Beyond Black-box Optimization

Recall general blackbox optimization:



Only mode of interaction with  $f$ : querying  $f$ 's value at a given  $\lambda$

Too slow for tuning expensive models

# Methods for Going Beyond Blackbox Bayesian Optimization

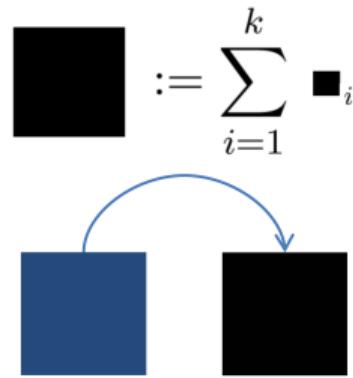
- Sum of little black boxes

- ▶ Each little black box is fast but only yields a noisy estimate
- ▶ SMAC [Hutter et al. 2011] directly solves  $\arg \min_{\lambda \in \Lambda} \sum_{i=1}^N c(\lambda, i)$
- ▶ Auto-WEKA [Thornton et al, 2013] used this to optimize 10-fold cross-validation performance

$$\blacksquare := \sum_{i=1}^k \blacksquare_i$$

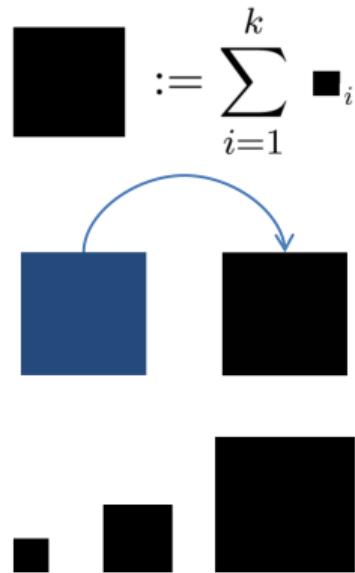
# Methods for Going Beyond Blackbox Bayesian Optimization

- Sum of little black boxes
  - ▶ Each little black box is fast but only yields a noisy estimate
  - ▶ SMAC [Hutter et al. 2011] directly solves  $\arg \min_{\lambda \in \Lambda} \sum_{i=1}^N c(\lambda, i)$
  - ▶ Auto-WEKA [Thornton et al, 2013] used this to optimize 10-fold cross-validation performance
- Meta-learning across problems / datasets



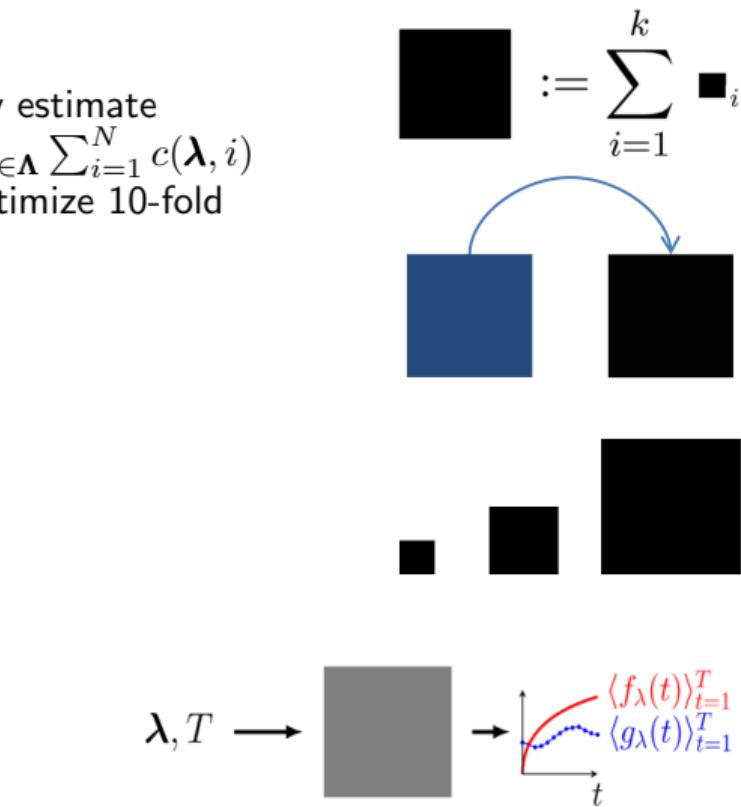
# Methods for Going Beyond Blackbox Bayesian Optimization

- Sum of little black boxes
  - ▶ Each little black box is fast but only yields a noisy estimate
  - ▶ SMAC [Hutter et al. 2011] directly solves  $\arg \min_{\lambda \in \Lambda} \sum_{i=1}^N c(\lambda, i)$
  - ▶ Auto-WEKA [Thornton et al, 2013] used this to optimize 10-fold cross-validation performance
- Meta-learning across problems / datasets
- Multi-fidelity optimization



# Methods for Going Beyond Blackbox Bayesian Optimization

- Sum of little black boxes
  - ▶ Each little black box is fast but only yields a noisy estimate
  - ▶ SMAC [Hutter et al. 2011] directly solves  $\arg \min_{\lambda \in \Lambda} \sum_{i=1}^N c(\lambda, i)$
  - ▶ Auto-WEKA [Thornton et al, 2013] used this to optimize 10-fold cross-validation performance
- Meta-learning across problems / datasets
- Multi-fidelity optimization
- Graybox optimization / learning curve prediction



# Learning Goals of this Lecture

After this lecture, students can ...

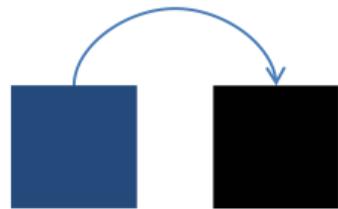
- Describe many different ways of using [meta-learning](#) to speed up HPO
- Explain the concept of [multi-fidelity](#) optimization to speed up HPO
- Explain the [Successive Halving](#) and [Hyperband](#) algorithms
- Explain how to combine Bayesian optimization and Hyperband in BOHB
- Describe how to [exploit multiple fidelities](#) in Bayesian optimization
- Discuss several ways of predicting [learning curves](#)
- Discuss [success stories](#) of speeding up Bayesian optimization

# Speedup Techniques for Hyperparameter Optimization

## Meta-Learning

Bernd Bischl    Frank Hutter    Lars Kotthoff  
Marius Lindauer    Joaquin Vanschoren

# Introduction



- Learning essentially never stops:
  - ▶ Many models are periodically re-fit to track changes in the data
  - ▶ Many models are re-fit to perform well on new tasks
- The best hyperparameter configuration tends to remain quite stable across tasks

# Introduction



- Learning essentially never stops:
  - ▶ Many models are periodically re-fit to track changes in the data
  - ▶ Many models are re-fit to perform well on new tasks
- The best hyperparameter configuration tends to remain quite stable across tasks

For a good introduction to meta-learning in general, see [AutoML Book: Chapter 2]

## Problem Statement

Given:

- a set of prior tasks:  $t_j \in \mathcal{T}_{\text{meta}} \subset \mathcal{T}$ ,
- a set of new tasks:  $t_{\text{new}} \in \mathcal{T}$ ,

## Problem Statement

Given:

- a set of prior tasks:  $t_j \in \mathcal{T}_{\text{meta}} \subset \mathcal{T}$ ,
- a set of new tasks:  $t_{\text{new}} \in \mathcal{T}$ ,
- a set of learning algorithms, fully defined by  $\theta_i \in \Theta$

## Problem Statement

Given:

- a set of prior tasks:  $t_j \in \mathcal{T}_{\text{meta}} \subset \mathcal{T}$ ,
- a set of new tasks:  $t_{\text{new}} \in \mathcal{T}$ ,
- a set of learning algorithms, fully defined by  $\theta_i \in \Theta$
- a set of prior evaluations  $\mathcal{D}_{\text{meta}}$  on  $t_j \in \mathcal{T}_{\text{meta}}$
- a set of evaluations  $\mathcal{D}_{\text{new}}$  on new task  $t_{\text{new}}$

# Problem Statement

Given:

- a set of prior tasks:  $t_j \in \mathcal{T}_{\text{meta}} \subset \mathcal{T}$ ,
- a set of new tasks:  $t_{\text{new}} \in \mathcal{T}$ ,
- a set of learning algorithms, fully defined by  $\theta_i \in \Theta$
- a set of prior evaluations  $\mathcal{D}_{\text{meta}}$  on  $t_j \in \mathcal{T}_{\text{meta}}$
- a set of evaluations  $\mathcal{D}_{\text{new}}$  on new task  $t_{\text{new}}$

Goal of meta-learning:

- use meta-data  $\mathcal{D}_{\text{meta}}$  to choose  $\theta_i \in \Theta$  for  $t_{\text{new}}$  better than only based on  $\mathcal{D}_{\text{new}}$ .

[adapted from AutoML Book: Chapter 2]

# The Role of Meta-Features

- We can often extract additional characteristics for each task, called **meta-features**
- Each task  $t_j$  can be described by a vector of  $K$  meta-features:

$$m(t_j) = (m_{j,1}, \dots, m_{j,K})$$

# The Role of Meta-Features

- We can often extract additional characteristics for each task, called **meta-features**
- Each task  $t_j$  can be described by a vector of  $K$  meta-features:

$$m(t_j) = (m_{j,1}, \dots, m_{j,K})$$

- This vector can be used to define a **similarity measure** between two tasks
  - ▶ e.g., calculating the Euclidean distance between  $m(t_i)$  and  $m(t_j)$
  - ▶ Based on similarity, we can transfer information from the most similar tasks to new task  $t_{\text{new}}$

# Overview of Meta-Features in Machine Learning

- **Simple** - easily extracted from the data, describe the basic dataset structure
  - ▶ e.g., number of features, data points or classes

# Overview of Meta-Features in Machine Learning

- **Simple** - easily extracted from the data, describe the basic dataset structure
  - ▶ e.g., number of features, data points or classes
- **Statistical** - characterize the data via descriptive statistics:
  - ▶ e.g., average or standard deviation of features, or their correlation with the labels

# Overview of Meta-Features in Machine Learning

- **Simple** - easily extracted from the data, describe the basic dataset structure
  - ▶ e.g., number of features, data points or classes
- **Statistical** - characterize the data via descriptive statistics:
  - ▶ e.g., average or standard deviation of features, or their correlation with the labels
- **Information-theoretic** - measure the class entropy in the data
  - ▶ capture the amount of information in the data

# Overview of Meta-Features in Machine Learning

- **Simple** - easily extracted from the data, describe the basic dataset structure
  - ▶ e.g., number of features, data points or classes
- **Statistical** - characterize the data via descriptive statistics:
  - ▶ e.g., average or standard deviation of features, or their correlation with the labels
- **Information-theoretic** - measure the class entropy in the data
  - ▶ capture the amount of information in the data
- **Model-based** - extracted from a model induced using the training data
  - ▶ these are often based on properties of decision tree models
  - ▶ e.g., number of leaves, number of nodes, shape of the tree

# Overview of Meta-Features in Machine Learning

- **Simple** - easily extracted from the data, describe the basic dataset structure
  - ▶ e.g., number of features, data points or classes
- **Statistical** - characterize the data via descriptive statistics:
  - ▶ e.g., average or standard deviation of features, or their correlation with the labels
- **Information-theoretic** - measure the class entropy in the data
  - ▶ capture the amount of information in the data
- **Model-based** - extracted from a model induced using the training data
  - ▶ these are often based on properties of decision tree models
  - ▶ e.g., number of leaves, number of nodes, shape of the tree
- **Landmarking** - computed by running several fast ML algorithms on the dataset
  - ▶ e.g., is fast algorithm A better than fast algorithm B on this dataset?
  - ▶ this can capture different properties of the dataset, e.g., linear separability

# Overview of Meta-Features in Machine Learning

- **Simple** - easily extracted from the data, describe the basic dataset structure
  - ▶ e.g., number of features, data points or classes
- **Statistical** - characterize the data via descriptive statistics:
  - ▶ e.g., average or standard deviation of features, or their correlation with the labels
- **Information-theoretic** - measure the class entropy in the data
  - ▶ capture the amount of information in the data
- **Model-based** - extracted from a model induced using the training data
  - ▶ these are often based on properties of decision tree models
  - ▶ e.g., number of leaves, number of nodes, shape of the tree
- **Landmarking** - computed by running several fast ML algorithms on the dataset
  - ▶ e.g., is fast algorithm A better than fast algorithm B on this dataset?
  - ▶ this can capture different properties of the dataset, e.g., linear separability
- **Others** - not included in the previous groups
  - ▶ e.g., time related measures, clustering and distance-based measures

## Meta-Learning for HPO Approach 1: Warmstarting

- Experts often start HPO from a strong default (rather than random configurations)

## Meta-Learning for HPO Approach 1: Warmstarting

- Experts often start HPO from a strong default (rather than random configurations)
- Can we learn from meta-data  $\mathcal{D}_{\text{meta}}$  how to [initialize](#) HPO?

## Meta-Learning for HPO Approach 1: Warmstarting

- Experts often start HPO from a strong default (rather than random configurations)
- Can we learn from meta-data  $\mathcal{D}_{\text{meta}}$  how to [initialize](#) HPO?
- Note: just a single default configuration often does not perform great on a new dataset
  - ▶ Otherwise there would be no point in HPO

## Meta-Learning for HPO Approach 2: Model-Warmstarting

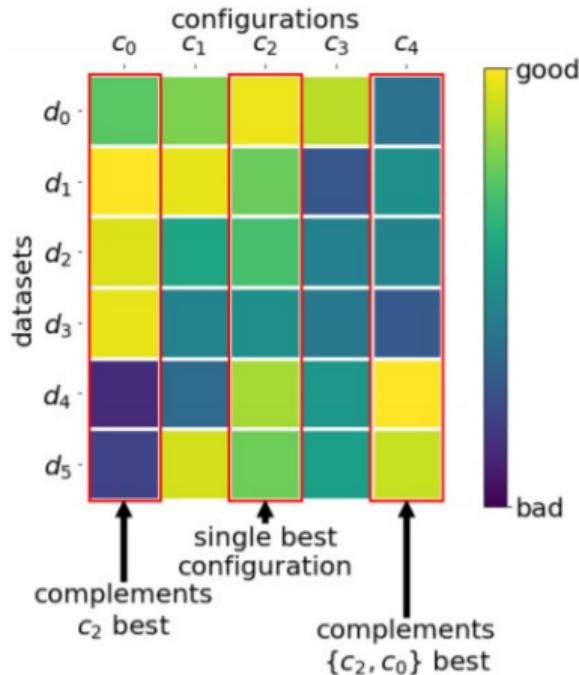
- Many HPO methods use a predictive model (e.g., Bayesian optimization)
- By running HPO on different datasets, we learn something about the search landscape
  - ▶ E.g., what are bad regions of the configuration space in general

## Meta-Learning for HPO Approach 2: Model-Warmstarting

- Many HPO methods use a predictive model (e.g., Bayesian optimization)
- By running HPO on different datasets, we learn something about the search landscape
  - ▶ E.g., what are bad regions of the configuration space in general
- Given:  $n$  predictive models  $\hat{c}_{\mathcal{D}_i} : \Lambda \rightarrow \mathbb{R}$  from HPO on  $\mathcal{T}_{\text{meta}}$
- How can we use these  $\hat{c}_{\mathcal{D}_i}$  to speed up HPO?

# Meta-Learning for HPO Approach 3: Task-independent Recommendations

- *Idea:* learn a sorted list of defaults
- *Method:* mostly greedy on  $\mathcal{T}_{\text{meta}}$
- *Results:* surprisingly strong,  
better than Bayesian Optimization



[Wistuba et al. 2017; Feurer et al. 2018; Pfisterer et al. 2018]

# Meta-Learning for HPO Approach 3: Task-independent Recommendations

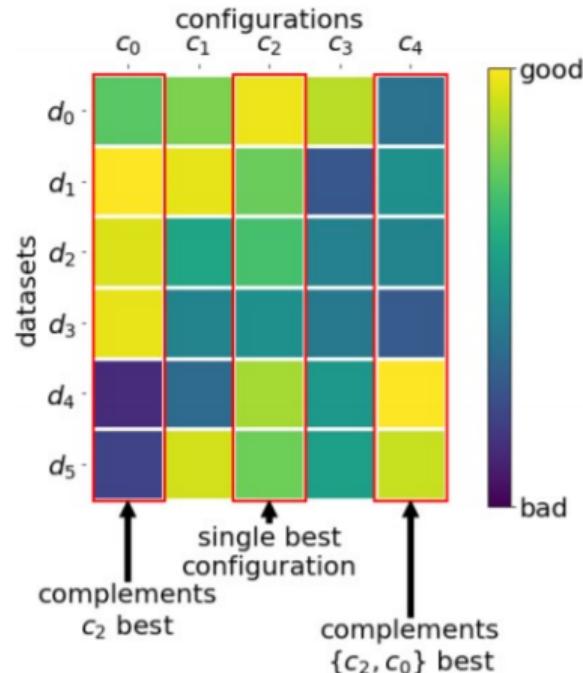
- Idea: learn a sorted list of defaults
- Method: mostly greedy on  $\mathcal{T}_{\text{meta}}$
- Results: surprisingly strong,  
better than Bayesian Optimization

## Advantages

- Easy to share and use
- Strong anytime performance
- Embarrassingly parallel

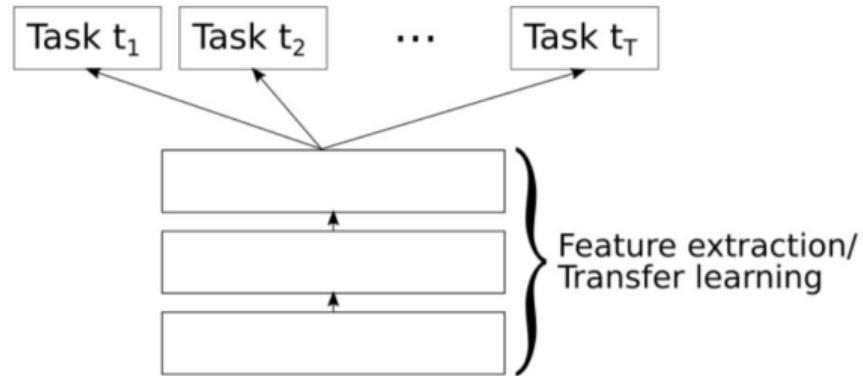
## Disadvantages

- Not adaptive



## Meta-Learning for HPO Approach 4: Joint model for Bayesian optimization

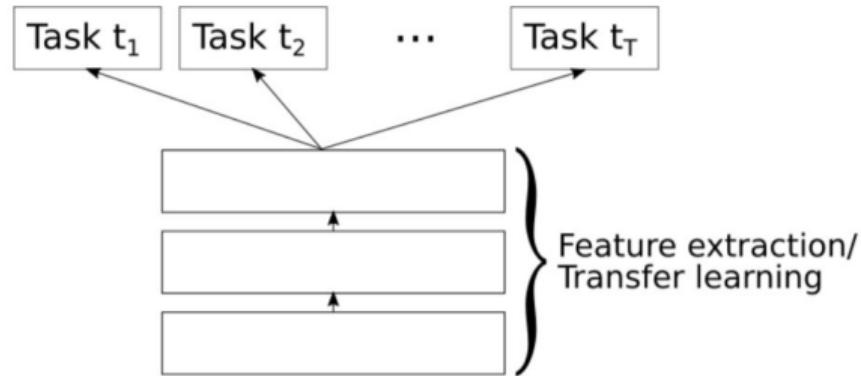
- Jointly train a “deep” neural network **on all tasks**



[Perrone et al. 2018]

# Meta-Learning for HPO Approach 4: Joint model for Bayesian optimization

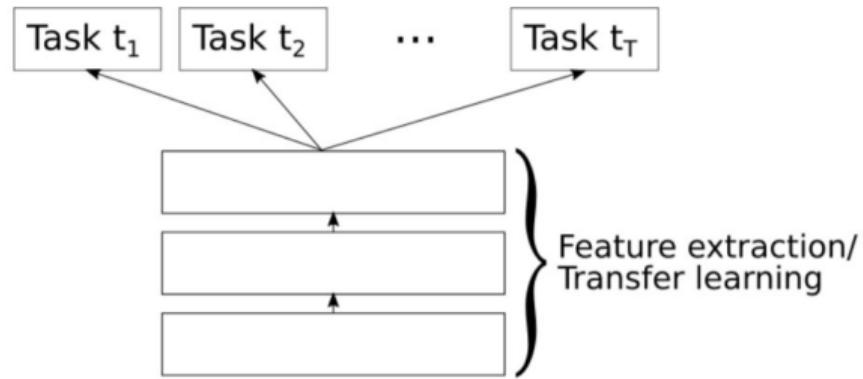
- Jointly train a “deep” neural network **on all tasks**
  - ▶ Have a separate output layer (head) for each task
  - ▶ Each head is a Bayesian linear regression (recall DNGO)



[Perrone et al. 2018]

## Meta-Learning for HPO Approach 4: Joint model for Bayesian optimization

- Jointly train a “deep” neural network **on all tasks**
  - ▶ Have a separate output layer (head) for each task
  - ▶ Each head is a Bayesian linear regression (recall DNGO)
- This uses meta-learning for feature extraction on the hyperparameter configurations



[Perrone et al. 2018]

## Meta-Learning for HPO Approach 5: Learning a Black-box Optimization Algorithm from Data

- Learning a blackbox optimization algorithm
  - ▶ Use  $\mathcal{D}_{\text{meta}}$  to learn a mapping from  $\mathcal{D}_{\text{new}}$  to the next configuration  $\lambda$  to evaluate
  - ▶ This mapping can be a (recurrent) neural net  $\text{NN}_\phi : \mathcal{D}_{\text{new}} \mapsto \lambda$  parameterized by weights  $\phi$

## Meta-Learning for HPO Approach 5: Learning a Black-box Optimization Algorithm from Data

- Learning a blackbox optimization algorithm
  - ▶ Use  $\mathcal{D}_{\text{meta}}$  to learn a mapping from  $\mathcal{D}_{\text{new}}$  to the next configuration  $\lambda$  to evaluate
  - ▶ This mapping can be a (recurrent) neural net  $\text{NN}_\phi : \mathcal{D}_{\text{new}} \mapsto \lambda$  parameterized by weights  $\phi$
  - ▶ This mapping  $\text{NN}_\phi$  constitutes a blackbox optimization algorithm

## Meta-Learning for HPO Approach 5: Learning a Black-box Optimization Algorithm from Data

- Learning a blackbox optimization algorithm
  - ▶ Use  $\mathcal{D}_{\text{meta}}$  to learn a mapping from  $\mathcal{D}_{\text{new}}$  to the next configuration  $\lambda$  to evaluate
  - ▶ This mapping can be a (recurrent) neural net  $\text{NN}_\phi : \mathcal{D}_{\text{new}} \mapsto \lambda$  parameterized by weights  $\phi$
  - ▶ This mapping  $\text{NN}_\phi$  constitutes a blackbox optimization algorithm
- Existing approaches for learning a blackbox optimizer
  - ▶ Gradient descent on  $\phi$  [Chen et al. 2016]
    - ★ Simplest technique, but requires backpropagation through the optimization trace
    - ★ This also requires the blackbox functions  $f$  used for training to be differentiable

## Meta-Learning for HPO Approach 5: Learning a Black-box Optimization Algorithm from Data

- Learning a blackbox optimization algorithm
  - ▶ Use  $\mathcal{D}_{\text{meta}}$  to learn a mapping from  $\mathcal{D}_{\text{new}}$  to the next configuration  $\lambda$  to evaluate
  - ▶ This mapping can be a (recurrent) neural net  $\text{NN}_\phi : \mathcal{D}_{\text{new}} \mapsto \lambda$  parameterized by weights  $\phi$
  - ▶ This mapping  $\text{NN}_\phi$  constitutes a blackbox optimization algorithm
- Existing approaches for learning a blackbox optimizer
  - ▶ Gradient descent on  $\phi$  [Chen et al. 2016]
    - ★ Simplest technique, but requires backpropagation through the optimization trace
    - ★ This also requires the blackbox functions  $f$  used for training to be differentiable
  - ▶ Reinforcement learning [Li and Malik. 2016]
    - ★ Can be harder to get to work, but does not require differentiable  $f$

## Meta-Learning for HPO Approach 6: Learning Algorithm Parts

- Learning a complete optimization algorithm requires a lot of data
- It would be more sample-efficient to only replace hand-designed parts of an algorithm

## Meta-Learning for HPO Approach 6: Learning Algorithm Parts

- Learning a complete optimization algorithm requires a lot of data
- It would be more sample-efficient to only replace hand-designed parts of an algorithm
- In Bayesian optimization, a critical hand-designed heuristic is the acquisition function
  - ▶ Trade-off between exploitation and exploration, e.g., via PI, EI, UCB, ES, KG, ...
  - ▶ Depending on the problem at hand, you might need a different acquisition function

## Meta-Learning for HPO Approach 6: Learning Algorithm Parts

- Learning a complete optimization algorithm requires a lot of data
- It would be more sample-efficient to only replace hand-designed parts of an algorithm
- In Bayesian optimization, a critical hand-designed heuristic is the acquisition function
  - ▶ Trade-off between exploitation and exploration, e.g., via PI, EI, UCB, ES, KG, ...
  - ▶ Depending on the problem at hand, you might need a different acquisition function
- Idea: Learn a neural acquisition function from data, but still make use of the sample efficiency of Gaussian processes [Volpp et al. 2019]

## Meta-Learning for HPO Approach 6: Learning Algorithm Parts

- Learning a complete optimization algorithm requires a lot of data
- It would be more sample-efficient to only replace hand-designed parts of an algorithm
- In Bayesian optimization, a critical hand-designed heuristic is the acquisition function
  - ▶ Trade-off between exploitation and exploration, e.g., via PI, EI, UCB, ES, KG, ...
  - ▶ Depending on the problem at hand, you might need a different acquisition function
- Idea: Learn a neural acquisition function from data, but still make use of the sample efficiency of Gaussian processes [Volpp et al. 2019]
- Two options:
  - ▶ Only depend on predicted mean and variance:  $u_\phi(\lambda) = u_\phi(\mu_t(\lambda), \sigma_t(\lambda))$ 
    - ★ This allows to learn a general acquisition function

## Meta-Learning for HPO Approach 6: Learning Algorithm Parts

- Learning a complete optimization algorithm requires a lot of data
- It would be more sample-efficient to only replace hand-designed parts of an algorithm
- In Bayesian optimization, a critical hand-designed heuristic is the acquisition function
  - ▶ Trade-off between exploitation and exploration, e.g., via PI, EI, UCB, ES, KG, ...
  - ▶ Depending on the problem at hand, you might need a different acquisition function
- Idea: Learn a neural acquisition function from data, but still make use of the sample efficiency of Gaussian processes [Volpp et al. 2019]
- Two options:
  - ▶ Only depend on predicted mean and variance:  $u_\phi(\lambda) = u_\phi(\mu_t(\lambda), \sigma_t(\lambda))$ 
    - ★ This allows to learn a general acquisition function
  - ▶ Also depend on the  $\lambda$  value:  $u_\phi(\lambda) = u_\phi(\mu_t(\lambda), \sigma_t(\lambda), \lambda)$ 
    - ★ This allows to fine-tune to the characteristics of  $\mathcal{D}_{\text{meta}}$  (e.g., avoid poor parts of the space)

## Questions to Answer for Yourself / Discuss with Friends

- **Repetition.** What are the different kinds of meta-features which can be used to describe machine learning datasets?
- **Repetition.** List all the different ways of using the meta data for HPO you recall
- **Discussion.** In the various meta-learning approaches, what will happen if all prior tasks are dissimilar to the target task?

# Speedup Techniques for Hyperparameter Optimization

## Overview of Multi-Fidelity Optimization

Bernd Bischl    Frank Hutter    Lars Kotthoff  
Marius Lindauer    Joaquin Vanschoren

## Motivating Example

- One possible cheap approximation of an expensive function: use a data subset
  - ▶ Many cheap evaluations on small subsets
  - ▶ Few expensive evaluations on the full data



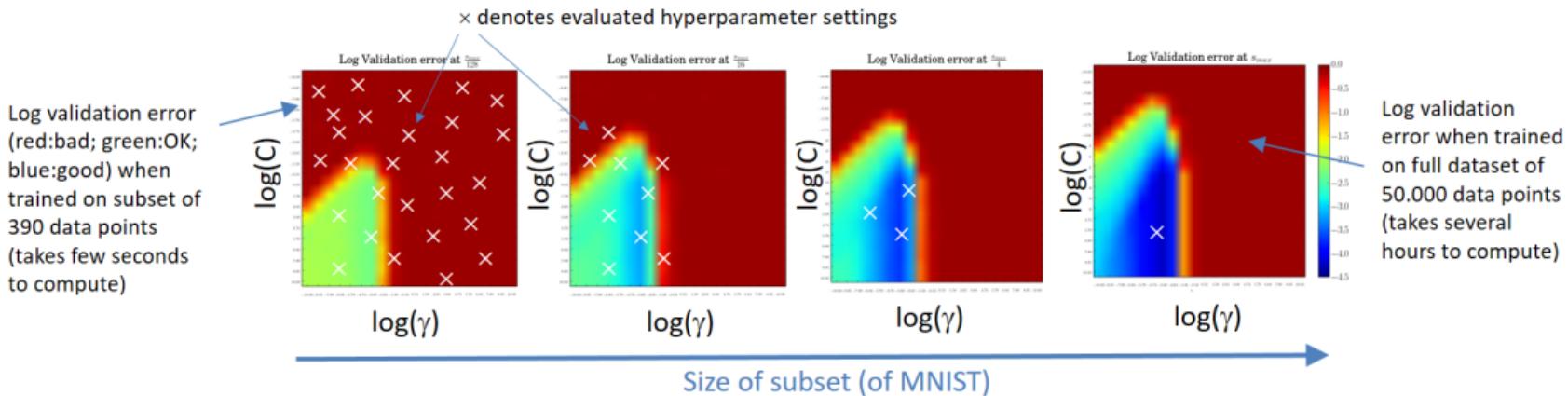
# Motivating Example

- One possible cheap approximation of an expensive function: use a data subset

- ▶ Many cheap evaluations on small subsets
  - ▶ Few expensive evaluations on the full data



- E.g.: Support Vector Machines (SVM) on MNIST dataset (hyperparameters:  $C$ ,  $\gamma$ )



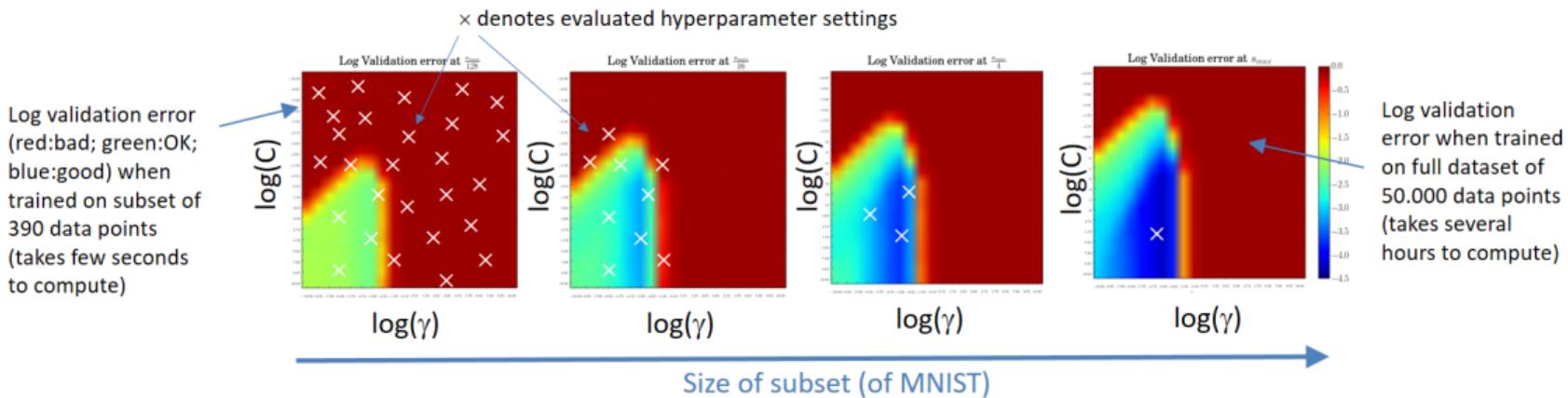
# Motivating Example

- One possible cheap approximation of an expensive function: use a data subset

- ▶ Many cheap evaluations on small subsets
  - ▶ Few expensive evaluations on the full data



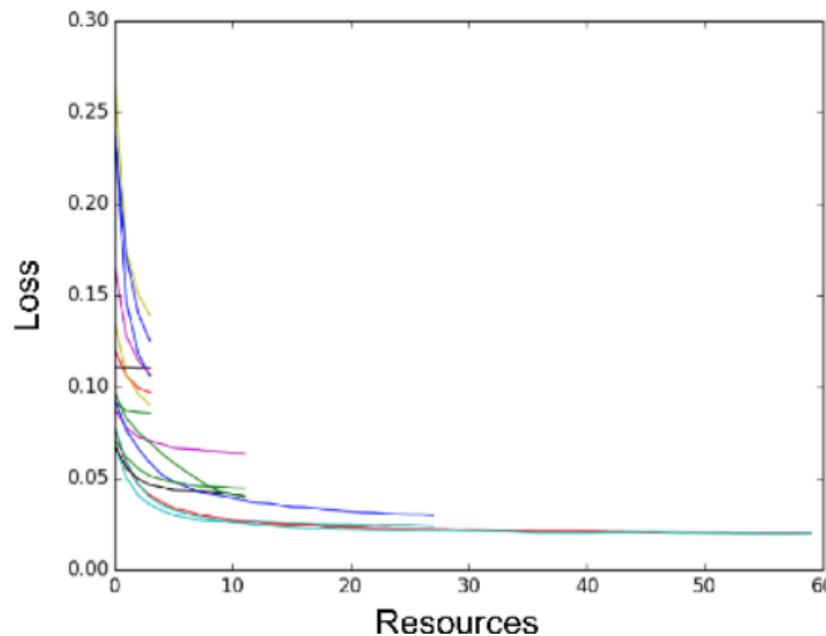
- E.g.: Support Vector Machines (SVM) on MNIST dataset (hyperparameters:  $C$ ,  $\gamma$ )



→ up to 1000x speedups over blackbox optimization on full data [Klein et al, AISTATS 2017]

## Motivating Example 2: Shorter Runs of Anytime Algorithms

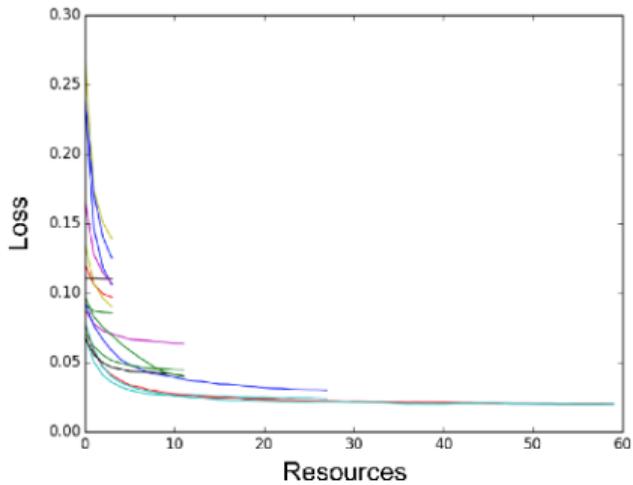
- Performance with shorter runs of an anytime algorithm (such as SGD):



# Multi-Fidelity Optimization In General

Exploit cheap approximations of an expensive blackbox function → afford more configurations

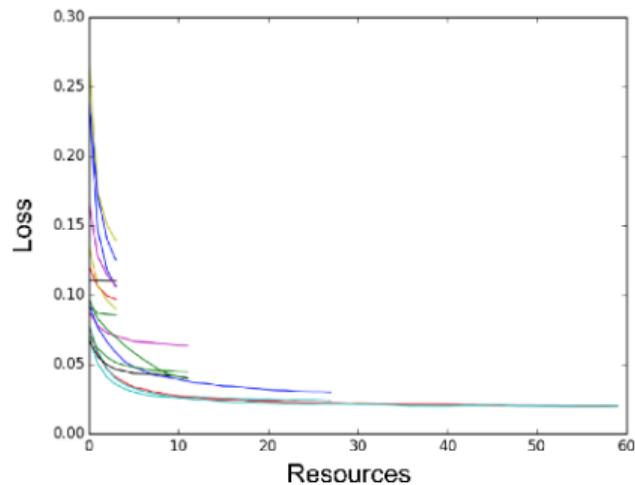
- Idea: eliminate poor configurations early, allocate more resources to promising ones.



# Multi-Fidelity Optimization In General

Exploit cheap approximations of an expensive blackbox function → afford more configurations

- Idea: eliminate poor configurations early, allocate more resources to promising ones.
  - Possible Resources:
    - ▶ Data subset size
    - ▶ Runtime / # epochs / # iterations

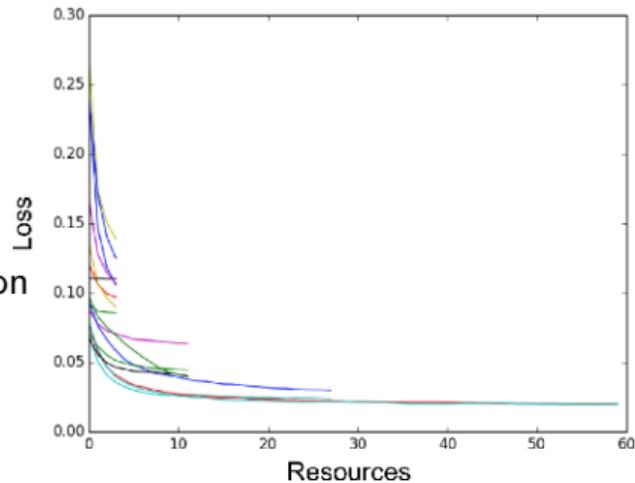


# Multi-Fidelity Optimization In General

Exploit cheap approximations of an expensive blackbox function → afford more configurations

- Idea: eliminate poor configurations early, allocate more resources to promising ones.

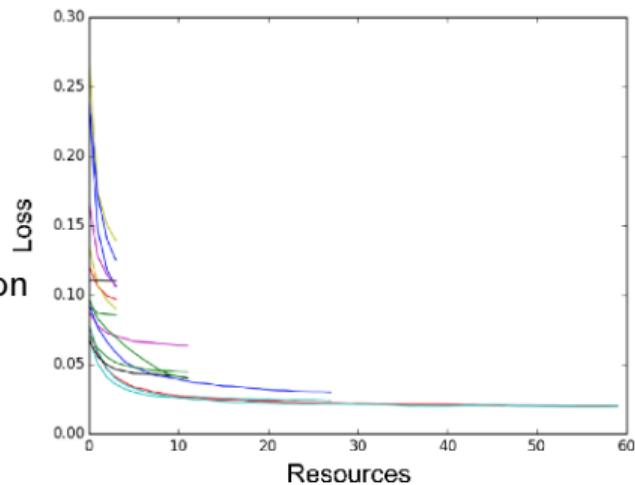
- Possible Resources:
  - ▶ Data subset size
  - ▶ Runtime / # epochs / # iterations
  - ▶ Downsampled size of images in object recognition
  - ▶ Depth / width of neural networks



# Multi-Fidelity Optimization In General

Exploit cheap approximations of an expensive blackbox function → afford more configurations

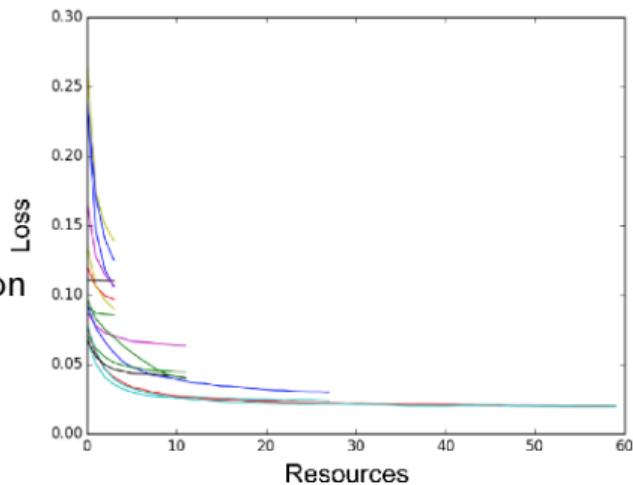
- Idea: eliminate poor configurations early, allocate more resources to promising ones.
- Possible Resources:
  - ▶ Data subset size
  - ▶ Runtime / # epochs / # iterations
  - ▶ Downsampled size of images in object recognition
  - ▶ Depth / width of neural networks
  - ▶ Number of trees
  - ▶ Number of features
  - ▶ Number of cross validation folds



# Multi-Fidelity Optimization In General

Exploit cheap approximations of an expensive blackbox function → afford more configurations

- Idea: eliminate poor configurations early, allocate more resources to promising ones.
  - Possible Resources:
    - Data subset size
    - Runtime / # epochs / # iterations
    - Downsampled size of images in object recognition
    - Depth / width of neural networks
    - Number of trees
    - Number of features
    - Number of cross validation folds
  - General concept, applicable even in fields outside ML, e.g., fluid simulation:
    - Number of particles
    - Time scale of simulation



## General Remarks on Multi-Fidelity Optimization

- Often, we have a choice which resources we use as budget

## General Remarks on Multi-Fidelity Optimization

- Often, we have a choice which resources we use as budget
- For multi-fidelity optimization to be helpful, performance with low budgets should be informative about performance with high budgets

## General Remarks on Multi-Fidelity Optimization

- Often, we have a choice which resources we use as budget
- For multi-fidelity optimization to be helpful, performance with low budgets should be informative about performance with high budgets
- In the simplest case: good with low resources  $\leftrightarrow$  good with high resources.
  - ▶ In practice, this is of course not always true

## How Useful is the Cheap Approximation? The Rank Correlation

Given:

- A set of configurations  $\Lambda = \{\lambda_1, \dots, \lambda_n\}$
- The performances  $f(\lambda_1), \dots, f(\lambda_n)$  on the expensive black box
- The performances  $g(\lambda_1), \dots, g(\lambda_n)$  on a cheap approximation of the black box

## How Useful is the Cheap Approximation? The Rank Correlation

Given:

- A set of configurations  $\Lambda = \{\lambda_1, \dots, \lambda_n\}$
- The performances  $f(\lambda_1), \dots, f(\lambda_n)$  on the expensive black box
- The performances  $g(\lambda_1), \dots, g(\lambda_n)$  on a cheap approximation of the black box

We compute the **Spearman rank correlation** between  $[f(\lambda_1), \dots, f(\lambda_n)]$  and  $[g(\lambda_1), \dots, g(\lambda_n)]$

- If this is high (in the extreme: 1), the relative ranking of the configurations is the same on  $f$  and  $g$ 
  - ▶ In that case, we can optimize cheaply on  $g$  and also obtain an optimum for  $f$
- If it is low ( $\approx 0$ ), optimizing  $g$  does not tell us anything about  $f$

## How Useful is the Cheap Approximation? The Rank Correlation

Given:

- A set of configurations  $\Lambda = \{\lambda_1, \dots, \lambda_n\}$
- The performances  $f(\lambda_1), \dots, f(\lambda_n)$  on the expensive black box
- The performances  $g(\lambda_1), \dots, g(\lambda_n)$  on a cheap approximation of the black box

We compute the **Spearman rank correlation** between  $[f(\lambda_1), \dots, f(\lambda_n)]$  and  $[g(\lambda_1), \dots, g(\lambda_n)]$

- If this is high (in the extreme: 1), the relative ranking of the configurations is the same on  $f$  and  $g$ 
  - ▶ In that case, we can optimize cheaply on  $g$  and also obtain an optimum for  $f$
- If it is low ( $\approx 0$ ), optimizing  $g$  does not tell us anything about  $f$

Goal: find approximations  $g$  that are very cheap but have high rank correlations with  $f$

## Questions to Answer for Yourself / Discuss with Friends

- **Repetition.** Which cheap approximation is better in this hypothetical case?
  - ▶ Downscaling images (5x cheaper, rank correlation of 0.8)
  - ▶ Less epoch of SGD (4x cheaper, rank correlation of 0.75)
- **Discussion.** Can you think of an application of your interest where you would likely have a good multi-fidelity approximation?

# Speedup Techniques for Hyperparameter Optimization

## Hyperband

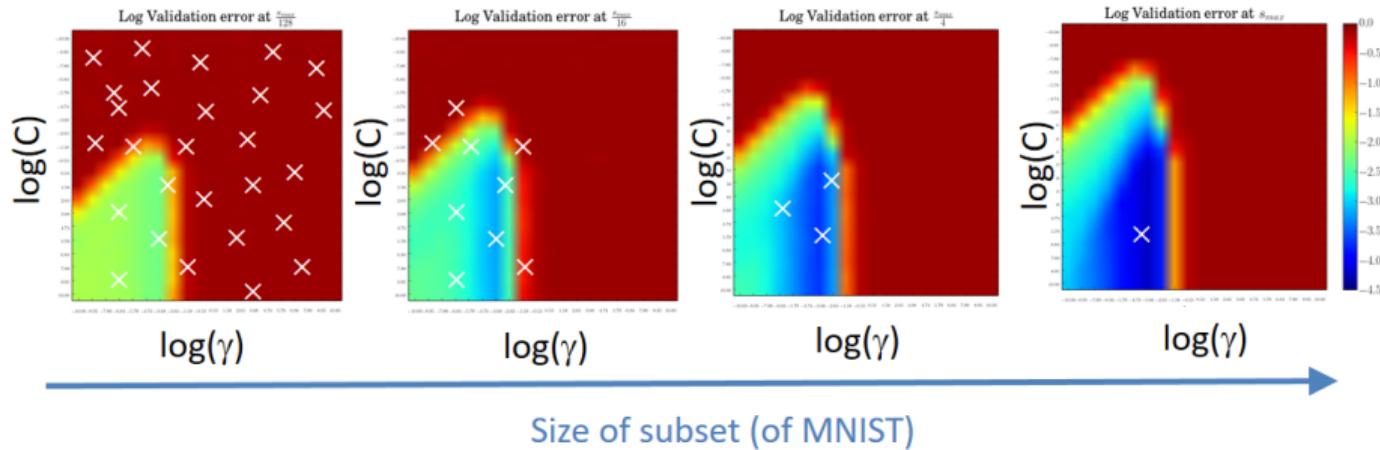
Bernd Bischl    Frank Hutter    Lars Kotthoff  
Marius Lindauer    Joaquin Vanschoren

# A Simple Multi-Fidelity Algorithms: Successive Halving (SH)

[Jamieson and Talwalkar, AISTATS 2016]

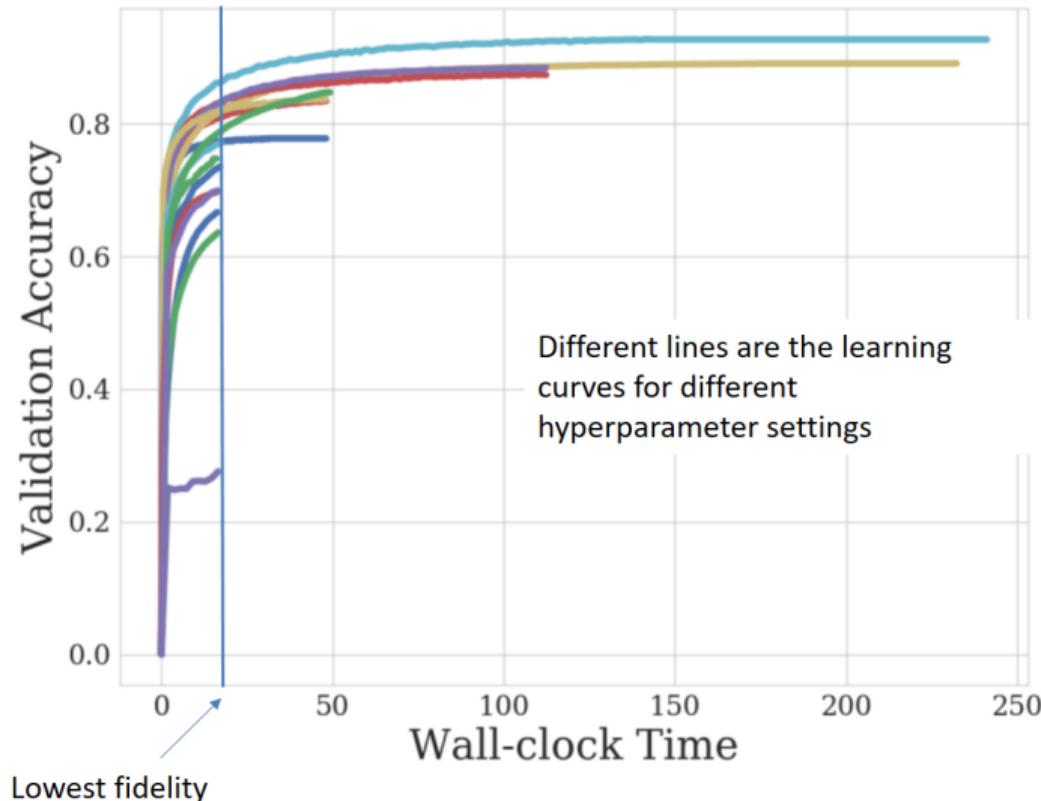
- A very simple algorithm:

- ▶ Sample N configurations uniformly at random & evaluate them on the cheapest fidelity
- ▶ Keep the best half (or third), move them to the next fidelity
- ▶ Iterate until the most expensive fidelity (= original expensive black box)



# The Same SH Algorithm When the Fidelity is Runtime

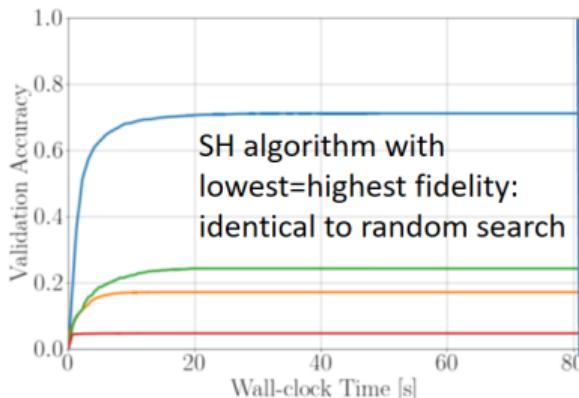
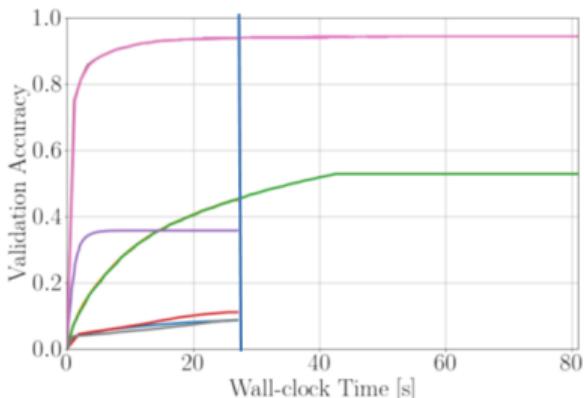
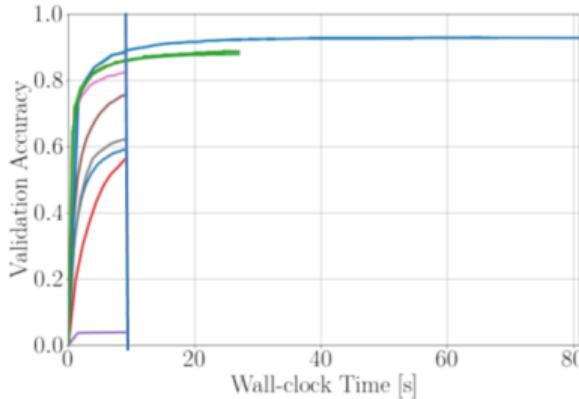
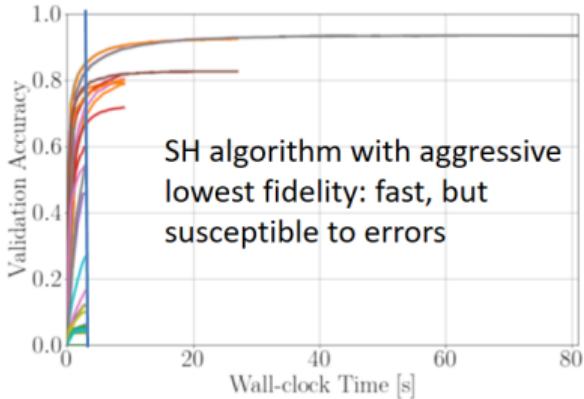
[Jamieson and Talwalkar, AISTATS 2016]



# An Extension of SH with Theoretical Guarantees: Hyperband

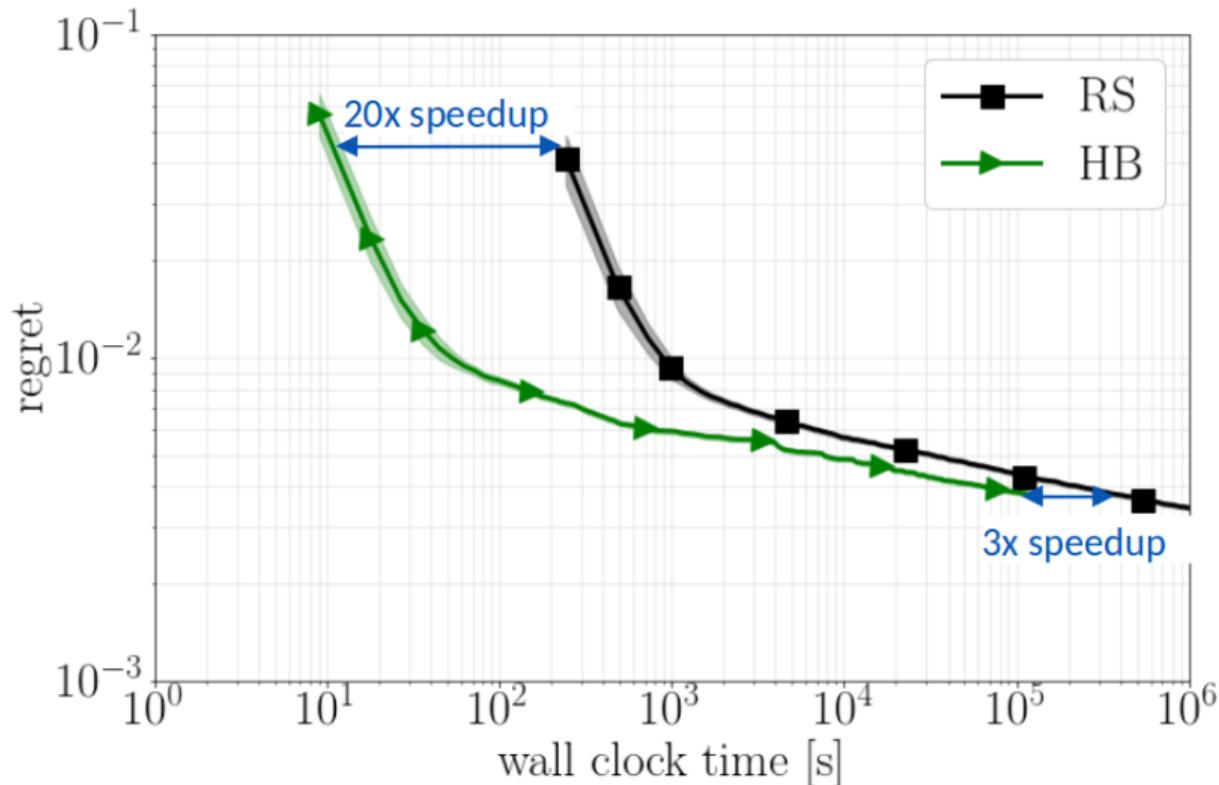
[Li et al., JMLR 2018]

- Main Idea:  
hedge against  
errors in cheap  
approximations
- Algorithm:  
run multiple copies  
of SH in parallel,  
starting at  
different cheapest  
fidelities



# Empirical Evaluation: Hyperband vs. Random Search

[Falkner, Klein & Hutter, ICML 2018]



## Questions to Answer for Yourself / Discuss with Friends

- Discussion. How do you think Hyperband would compare to successive halving using the most aggressive fidelity?
- Discussion. How slow is Hyperband in the worst case?

# Speedup Techniques for Hyperparameter Optimization

## BOHB

Bernd Bischl    Frank Hutter    Lars Kotthoff  
Marius Lindauer    Joaquin Vanschoren

# Robust and Efficient Hyperparameter Optimization at Scale

Desiderata for a practical solution to the hyperparameter optimization problem:

- Strong Anytime Performance
- Strong Final Performance
- Scalability
- Robustness & Flexibility
- Computational Efficiency
- Effective Use of Parallel Resources
- Conceptual / Algorithmic Simplicity

# Robust and Efficient Hyperparameter Optimization at Scale

Desiderata for a practical solution to the hyperparameter optimization problem:

- Strong Anytime Performance
- Strong Final Performance
- Scalability
- Robustness & Flexibility
- Computational Efficiency
- Effective Use of Parallel Resources
- Conceptual / Algorithmic Simplicity

To fulfill all of these desiderata, BOHB [Falkner, Klein and Hutter, ICML 2018] combines Bayesian Optimization with Hyperband

# BOHB

- BOHB combines the advantages of Bayesian Optimization and Hyperband
  - ▶ Bayesian Optimization for choosing configurations to achieve strong final performance
  - ▶ Hyperband to choose the budgets for good anytime performance

# BOHB

- BOHB combines the advantages of Bayesian Optimization and Hyperband
  - ▶ Bayesian Optimization for choosing configurations to achieve strong final performance
  - ▶ Hyperband to choose the budgets for good anytime performance
- BOHB replaces the random selection of configurations at the beginning of each HB iteration by a model-based search

# BOHB

- BOHB combines the advantages of Bayesian Optimization and Hyperband
  - ▶ Bayesian Optimization for choosing configurations to achieve strong final performance
  - ▶ Hyperband to choose the budgets for good anytime performance
- BOHB replaces the random selection of configurations at the beginning of each HB iteration by a model-based search
- Details of the model
  - ▶ Variant of the Tree Parzen Estimator, with a product kernel
  - ▶ Models are fitted independently to the data for one budget at a time
    - ★ Specifically, always the highest budget that has enough data points

# BOHB: Algorithm

---

## Pseudocode for sampling in BOHB

---

**Input** : observations  $D$ , fraction of random runs  $\rho$ , percentile  $\gamma$ , number of samples  $N_s$ , minimum number of points  $N_{min}$  to build a model, and bandwidth factor  $b_w$

**Output:** next configuration to evaluate

- 1 **if**  $rand() < \rho$  **then return** *random configuration*
  - 2  $b = \arg \max\{D_b : |D_b| \geq N_{min} + 2\}$
  - 3 **if**  $b = \emptyset$  **then return** *random configuration*
  - 4 Fit KDEs according to Equation ??
  - 5 Draw  $N_s$  samples according to  $l'(\lambda)$
  - 6 **return** *sample with highest ratio  $l(\lambda)/g(\lambda)$*
-

# BOHB: Algorithm

---

## Pseudocode for sampling in BOHB

---

**Input** : observations  $D$ , fraction of random runs  $\rho$ , percentile  $\gamma$ , number of samples  $N_s$ , minimum number of points  $N_{min}$  to build a model, and bandwidth factor  $b_w$

**Output:** next configuration to evaluate

- 1 **if**  $rand() < \rho$  **then return** *random configuration*
  - 2  $b = \arg \max\{D_b : |D_b| \geq N_{min} + 2\}$
  - 3 **if**  $b = \emptyset$  **then return** *random configuration*
  - 4 Fit KDEs according to Equation ??
  - 5 Draw  $N_s$  samples according to  $l'(\lambda)$
  - 6 **return** *sample with highest ratio  $l(\lambda)/g(\lambda)$*
- 

$$l(\lambda) = p(c(\lambda) \leq \gamma | D_b)$$

$$g(\lambda) = p(c(\lambda) > \gamma | D_b)$$

(1)

# BOHB: Algorithm

---

## Pseudocode for sampling in BOHB

---

**Input** : observations  $D$ , fraction of random runs  $\rho$ , percentile  $\gamma$ , number of samples  $N_s$ , minimum number of points  $N_{min}$  to build a model, and bandwidth factor  $b_w$

**Output:** next configuration to evaluate

- 1 **if**  $rand() < \rho$  **then return** *random configuration*
  - 2  $b = \arg \max\{D_b : |D_b| \geq N_{min} + 2\}$
  - 3 **if**  $b = \emptyset$  **then return** *random configuration*
  - 4 Fit KDEs according to Equation ??
  - 5 Draw  $N_s$  samples according to  $l'(\lambda)$
  - 6 **return** *sample with highest ratio  $l(\lambda)/g(\lambda)$*
- 

$$l(\lambda) = p(c(\lambda) \leq \gamma | D_b) \quad g(\lambda) = p(c(\lambda) > \gamma | D_b) \quad (1)$$

- Note:  $l'(\lambda)$  is similar to  $l(\lambda)$  but has larger bandwidths

# BOHB: Empirical Evaluation

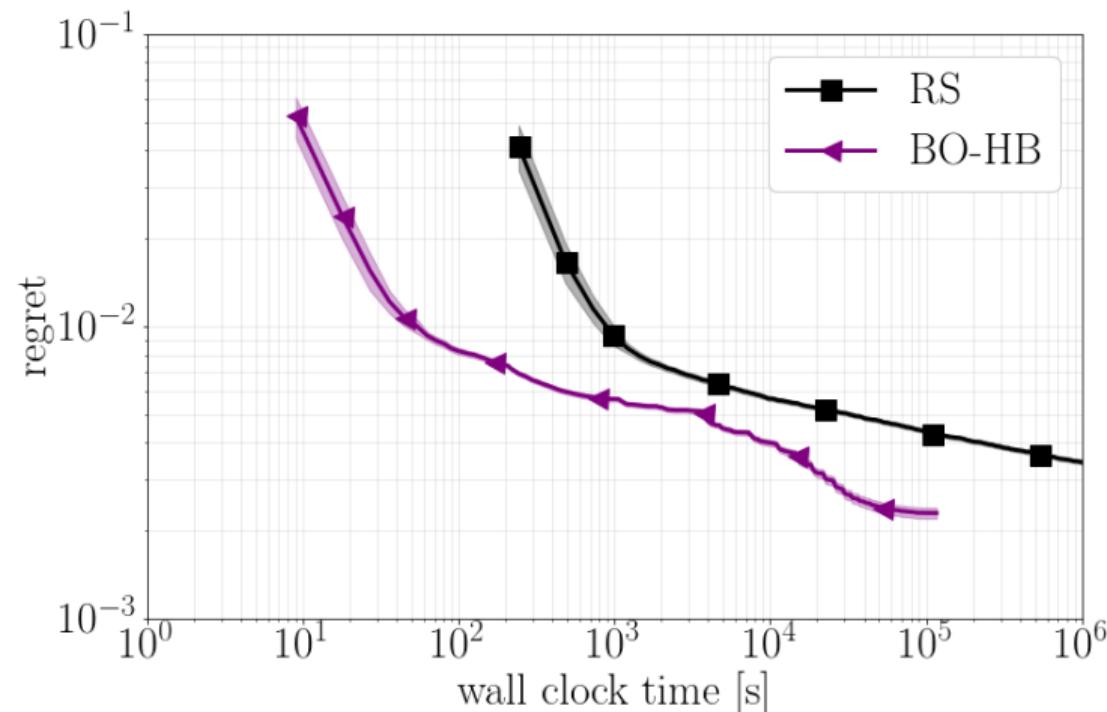


Figure: Performance of RS and BOHB on Auto-Net on dataset Adult

# BOHB: Empirical Evaluation

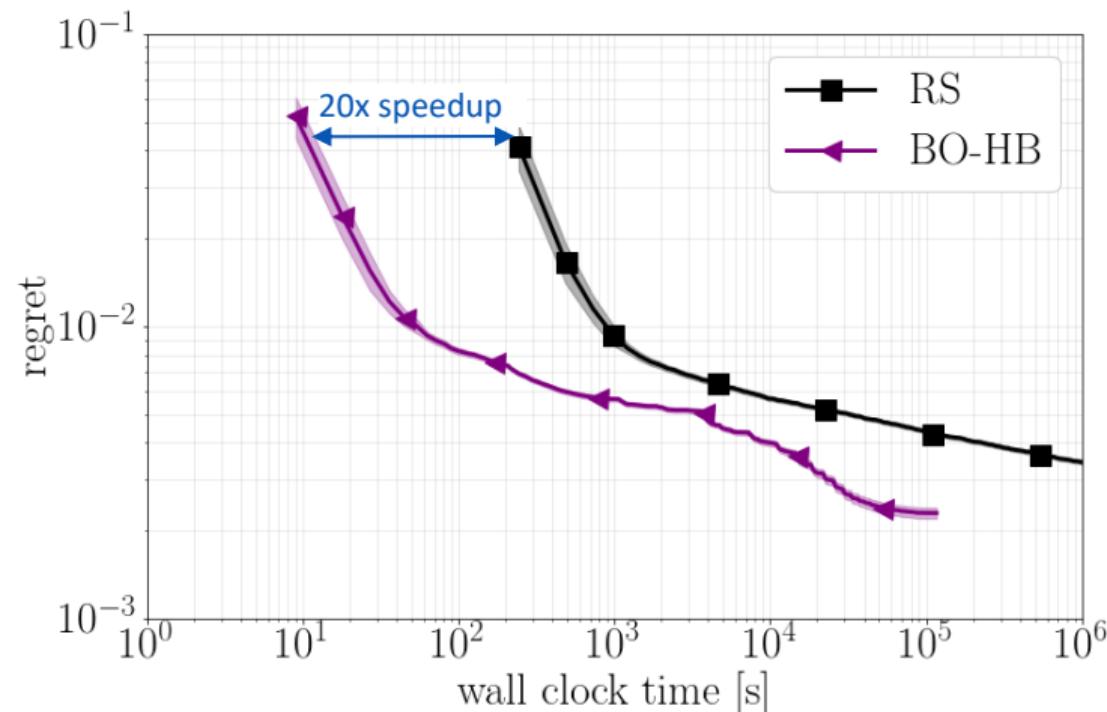


Figure: Performance of RS and BOHB on Auto-Net on dataset Adult

# BOHB: Empirical Evaluation

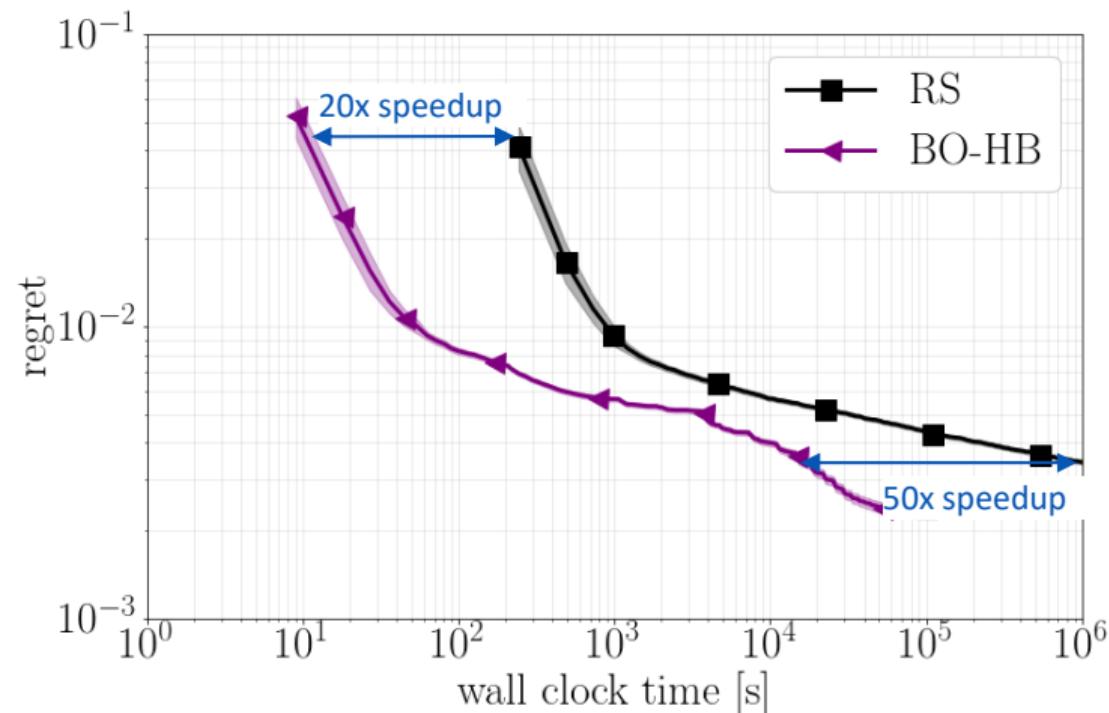


Figure: Performance of RS and BOHB on Auto-Net on dataset Adult

# BOHB: Empirical Evaluation

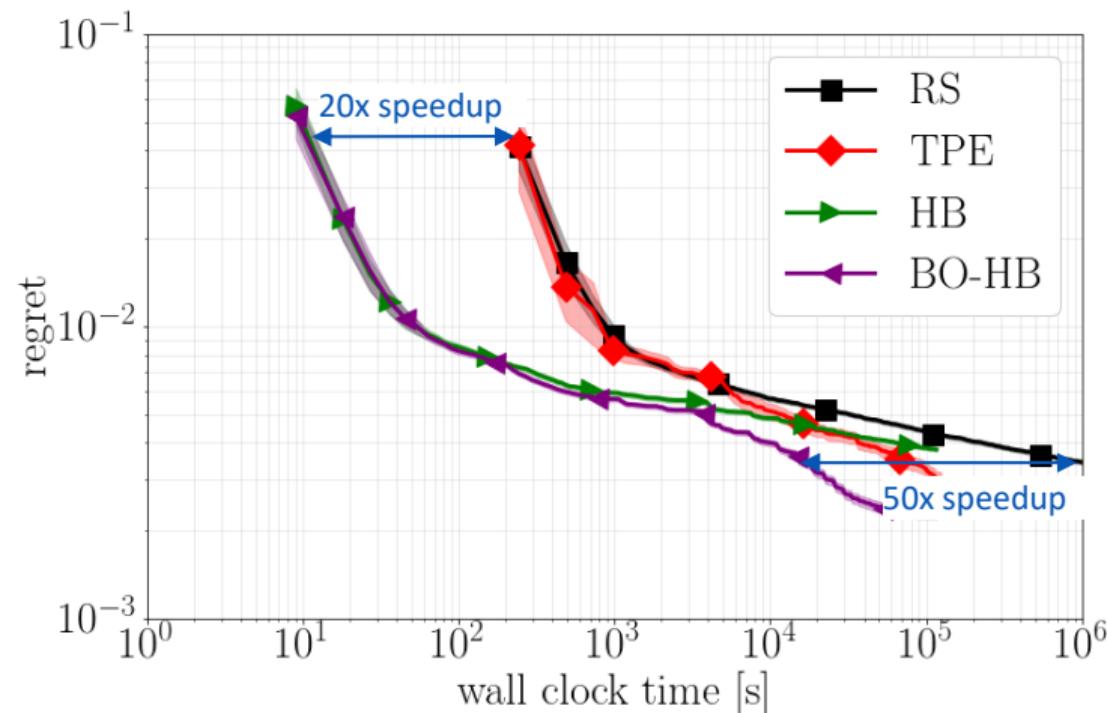
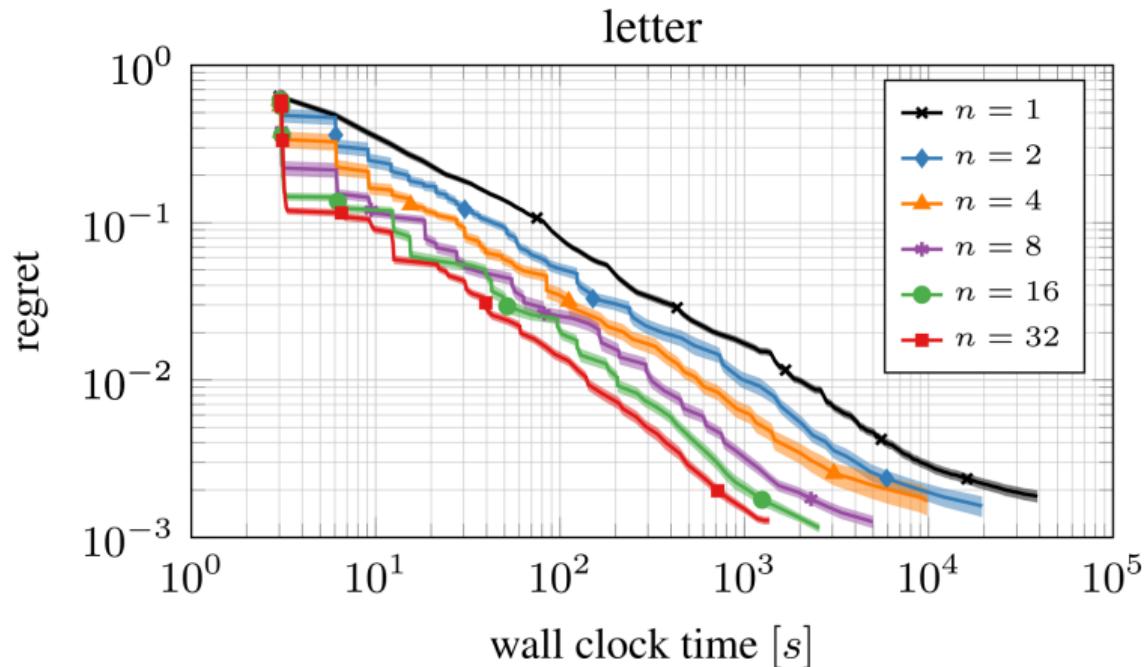


Figure: Performance of RS, TPE, HB and BOHB on Auto-Net on dataset Adult

# BOHB: Different number of parallel workers



**Figure:** Performance of BOHB with different number of parallel workers on the letter surrogate benchmark for 128 iterations

# BOHB: Optimization of a Bayesian Neural Network

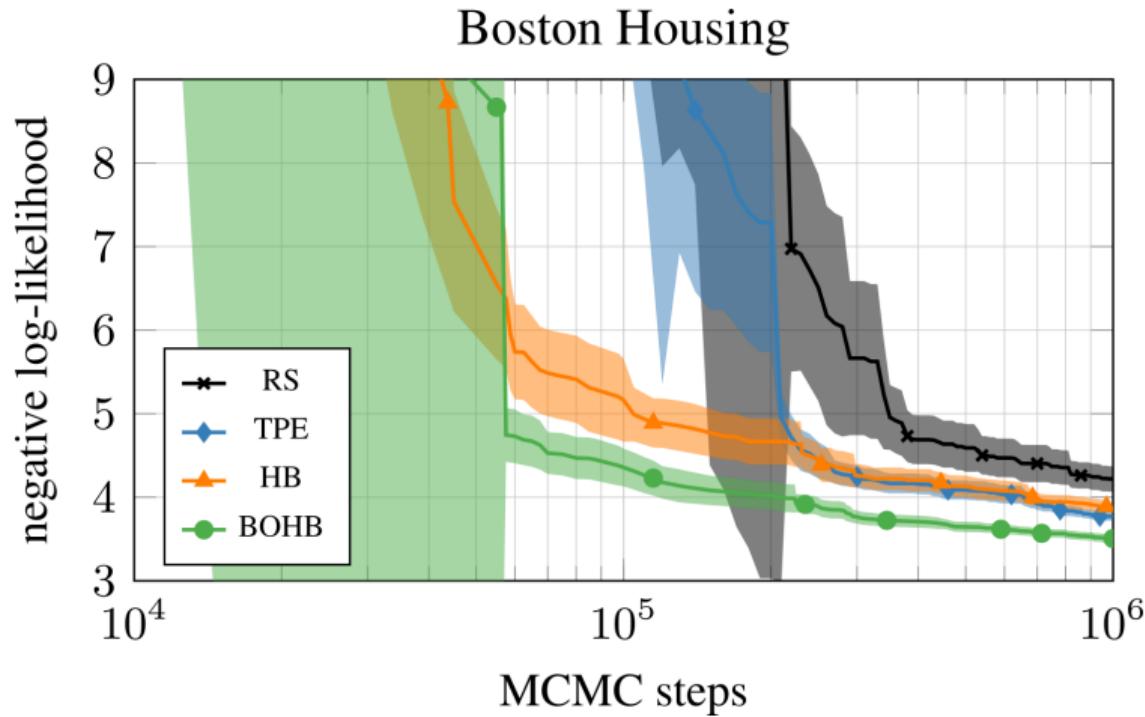


Figure: Optimization of 5 hyperparameters of a Bayesian neural network trained with SGHMC.

# BOHB: Optimization of a Reinforcement Learning Agent

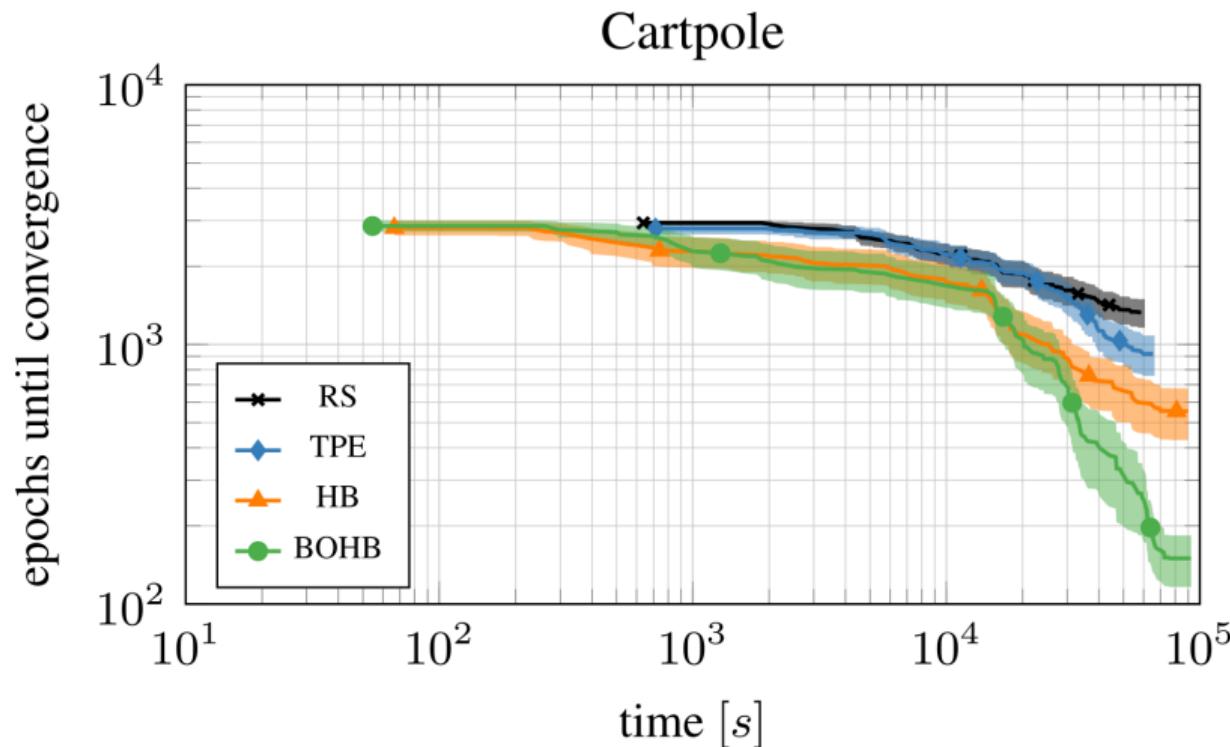


Figure: Hyperparameter optimization of 8 hyperparameters of PPO on the cartpole task.

# BOHB: Optimization of an SVM

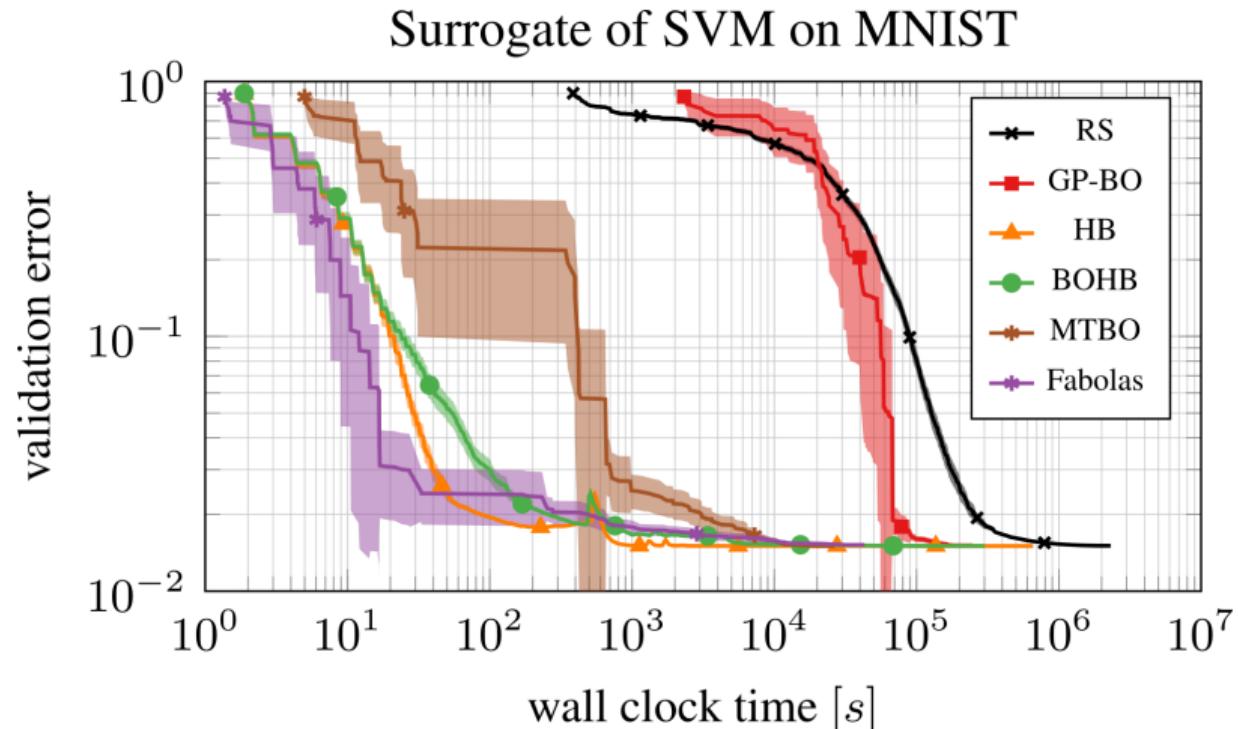


Figure: Comparison on the SVM on MNIST surrogate benchmark

# BOHB: Counting Ones

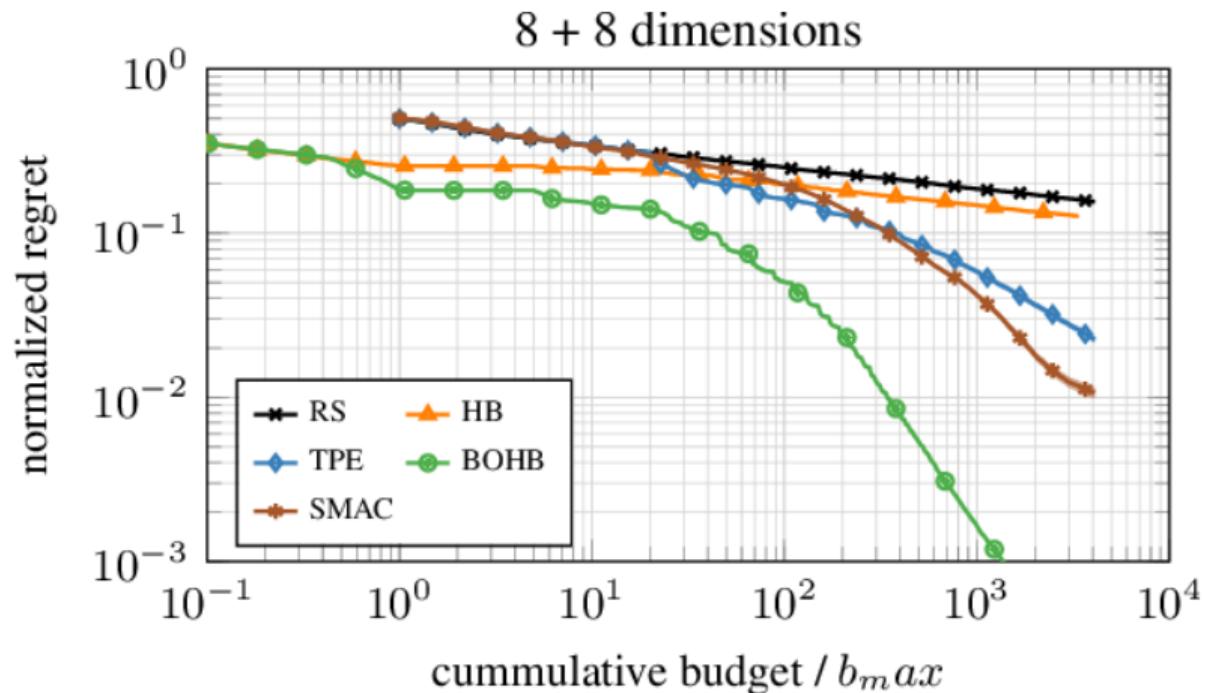


Figure: Results for the counting ones problem in 16 dimensional space with 8 categorical and 8 continuous hyperparameters.

# BOHB: Optimization of a Feedforward Network

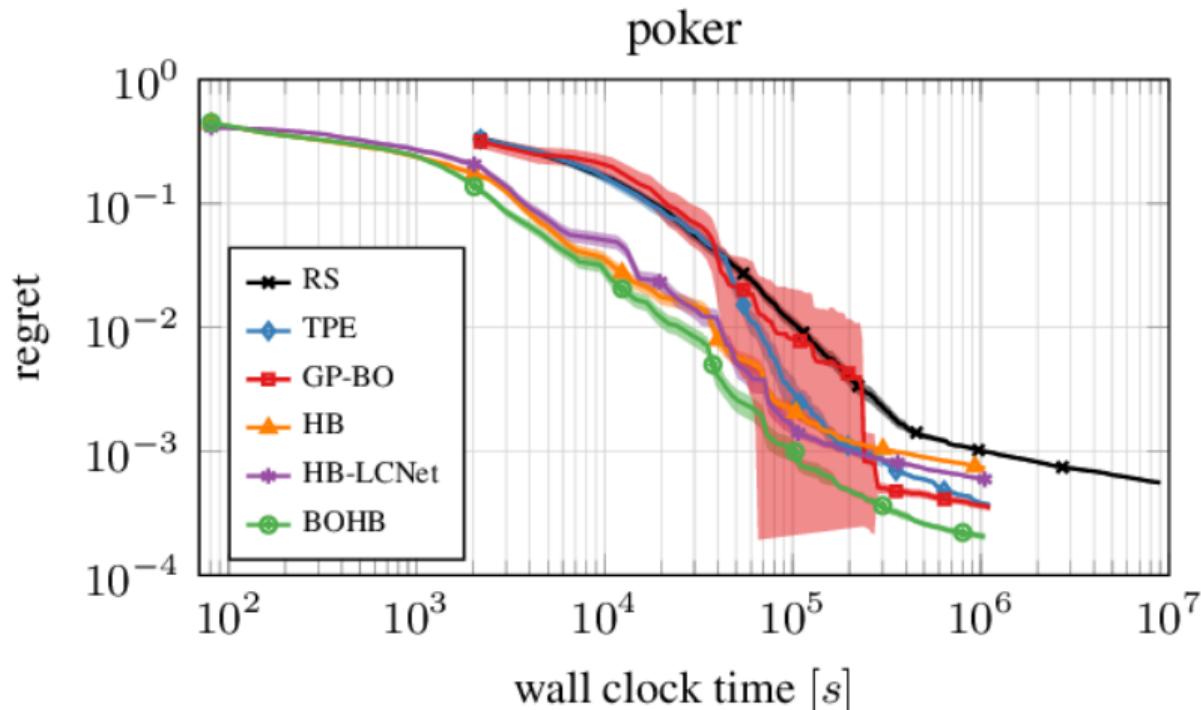


Figure: Optimizing six hyperparameters of a feed-forward neural network on featurized datasets; results are based on surrogate benchmarks.

## Questions to Answer for Yourself / Discuss with Friends

- **Repetition.** Why does BOHB interleave randomly sampled configurations in the optimization process?
- **Discussion.** What are the advantages of the Parzen estimator model over more advanced models such as random forests or Gaussian processes?

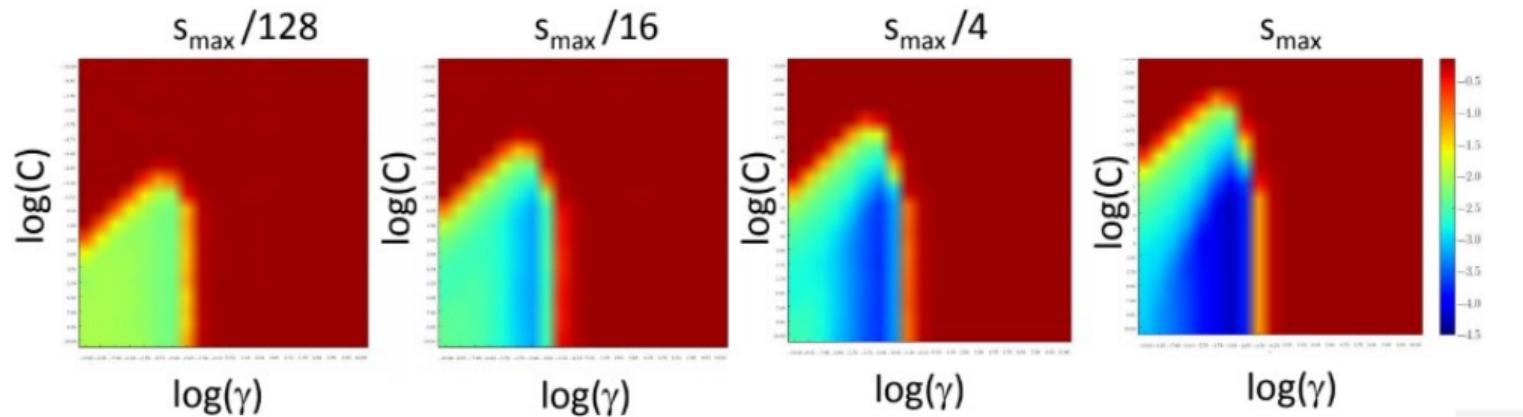
# Speedup Techniques for Hyperparameter Optimization

## Multi-fidelity Bayesian optimization

Bernd Bischl    Frank Hutter    Lars Kotthoff  
Marius Lindauer    Joaquin Vanschoren

## Motivating example

- Performance of an SVM on MNIST and subsets of it:



- ▶ Computational cost grows quadratically in dataset size  $z$
- ▶ Error shrinks smoothly with  $z$
- Evaluations on the smallest subset (about 400 data points) cost  $10\,000\times$  less than on the full data set

## Idea of Multi-fidelity Bayesian optimization [Kandasamy et al. 2017; Klein et al. 2016]

- Recall: standard Bayesian optimization uses a model  $\hat{c}(\lambda) \approx y$  to select the next  $\lambda$

## Idea of Multi-fidelity Bayesian optimization [Kandasamy et al. 2017; Klein et al. 2016]

- Recall: standard Bayesian optimization uses a model  $\hat{c}(\lambda) \approx y$  to select the next  $\lambda$
- Multi-fidelity Bayesian optimization uses a model  $\hat{c}(\lambda, z) \approx y$  to select the next  $(\lambda, z)$ 
  - ▶ Here,  $z \in \mathcal{Z}$  is the fidelity;  $\mathcal{Z}$  is the fidelity space, e.g.,  $\mathcal{Z} = [1, N_\bullet] \times [1, T_\bullet]$

## Idea of Multi-fidelity Bayesian optimization [Kandasamy et al. 2017; Klein et al. 2016]

- Recall: standard Bayesian optimization uses a model  $\hat{c}(\boldsymbol{\lambda}) \approx y$  to select the next  $\boldsymbol{\lambda}$
- Multi-fidelity Bayesian optimization uses a model  $\hat{c}(\boldsymbol{\lambda}, z) \approx y$  to select the next  $(\boldsymbol{\lambda}, z)$ 
  - ▶ Here,  $z \in \mathcal{Z}$  is the fidelity;  $\mathcal{Z}$  is the fidelity space, e.g.,  $\mathcal{Z} = [1, N_{\bullet}] \times [1, T_{\bullet}]$
- Denoting  $z_{\bullet}$  as the maximum fidelity (e.g.,  $z_{\bullet} = [N_{\bullet}, T_{\bullet}]$ ), our goal is to find:

$$\boldsymbol{\lambda}^* = \arg \min_{\boldsymbol{\lambda} \in \Lambda} f(\boldsymbol{\lambda}) = \arg \min_{\boldsymbol{\lambda} \in \Lambda} \hat{c}(\boldsymbol{\lambda}, z_{\bullet})$$

## Idea of Multi-fidelity Bayesian optimization [Kandasamy et al. 2017; Klein et al. 2016]

- Recall: standard Bayesian optimization uses a model  $\hat{c}(\boldsymbol{\lambda}) \approx y$  to select the next  $\boldsymbol{\lambda}$
- Multi-fidelity Bayesian optimization uses a model  $\hat{c}(\boldsymbol{\lambda}, z) \approx y$  to select the next  $(\boldsymbol{\lambda}, z)$ 
  - ▶ Here,  $z \in \mathcal{Z}$  is the fidelity;  $\mathcal{Z}$  is the fidelity space, e.g.,  $\mathcal{Z} = [1, N_{\bullet}] \times [1, T_{\bullet}]$
- Denoting  $z_{\bullet}$  as the maximum fidelity (e.g.,  $z_{\bullet} = [N_{\bullet}, T_{\bullet}]$ ), our goal is to find:

$$\boldsymbol{\lambda}^* = \arg \min_{\boldsymbol{\lambda} \in \Lambda} f(\boldsymbol{\lambda}) = \arg \min_{\boldsymbol{\lambda} \in \Lambda} \hat{c}(\boldsymbol{\lambda}, z_{\bullet})$$

- Implications for Bayesian optimization
  - ▶ Model  $\hat{c}$  needs to be good at extrapolating from small to large  $z$
  - ▶ Acquisition function now also needs to select  $z$  (i.e., take into account cost of evaluations)

## Entropy Search: Reminder

- Define the  $p_{\min}$  distribution given data  $\mathcal{D}$ :

$$p_{\min}(\boldsymbol{\lambda}^* \mid \mathcal{D}) := p(\boldsymbol{\lambda}^* \in \arg \min_{\boldsymbol{\lambda} \in \Lambda} f(\boldsymbol{\lambda}) \mid \mathcal{D})$$

- Entropy search aims to minimize the entropy  $\mathcal{H}[p_{\min}]$  [Hennig and Schuler. 2012]
  - ▶ It aims to be maximally certain about the location of  $\boldsymbol{\lambda}^*$

## Entropy Search: Reminder

- Define the  $p_{\min}$  distribution given data  $\mathcal{D}$ :

$$p_{\min}(\boldsymbol{\lambda}^* \mid \mathcal{D}) := p(\boldsymbol{\lambda}^* \in \arg \min_{\boldsymbol{\lambda} \in \Lambda} f(\boldsymbol{\lambda}) \mid \mathcal{D})$$

- Entropy search aims to minimize the entropy  $\mathcal{H}[p_{\min}]$  [Hennig and Schuler. 2012]
  - ▶ It aims to be maximally certain about the location of  $\boldsymbol{\lambda}^*$
- In a nutshell: select  $\boldsymbol{\lambda}$  that maximizes the following acquisition function:

$$u_{ES}(\boldsymbol{\lambda}) := \mathcal{H}[p_{\min}(\cdot \mid \mathcal{D})] - \mathbb{E}_{p(\tilde{c} \mid \boldsymbol{\lambda}, \mathcal{D})} [\mathcal{H}[p_{\min}(\cdot \mid \mathcal{D} \cup \langle \boldsymbol{\lambda}, \tilde{c} \rangle)]]$$

- We now care about the  $p_{\min}$  distribution for the maximal budget  $z_\bullet$ :

$$p_{\min}(\boldsymbol{\lambda}^* \mid \mathcal{D}) := p(\boldsymbol{\lambda}^* \in \arg \min_{\boldsymbol{\lambda} \in \Lambda} f(\boldsymbol{\lambda}, z_\bullet) \mid \mathcal{D})$$

- We still want to minimize the entropy  $\mathcal{H}[p_{\min}]$

# Entropy Search for Multi-Fidelity Optimization [Klein et al. 2017]

- We now care about the  $p_{\min}$  distribution for the maximal budget  $z_\bullet$ :

$$p_{\min}(\boldsymbol{\lambda}^* \mid \mathcal{D}) := p(\boldsymbol{\lambda}^* \in \arg \min_{\boldsymbol{\lambda} \in \Lambda} f(\boldsymbol{\lambda}, z_\bullet) \mid \mathcal{D})$$

- We still want to minimize the entropy  $\mathcal{H}[p_{\min}]$
- Now we aim for the biggest reduction in entropy per time spent
  - ▶ Now we don't model only  $f$ , but also the cost  $c(\boldsymbol{\lambda}, z)$
  - ▶ We choose the next  $(\boldsymbol{\lambda}, z)$  by maximizing:

$$u_{ES}(\boldsymbol{\lambda}, z \mid \mathcal{D}) := \mathbb{E}_{p(\tilde{c} \mid (\boldsymbol{\lambda}, z), \mathcal{D})} \left[ \frac{\mathcal{H}[p_{\min}(\cdot \mid \mathcal{D})] - \mathcal{H}[p_{\min}(\cdot \mid \mathcal{D} \cup \langle (\boldsymbol{\lambda}, z), \tilde{c} \rangle)]}{c(\boldsymbol{\lambda}, z)} \right]$$

## Entropy Search for Multi-Fidelity Optimization [Klein et al. 2017]

- The entire algorithm iterates the following 2 steps until time is up:

- Select  $(\lambda, z)$  by maximizing:

$$u_{ES}(\lambda, z \mid \mathcal{D}) := \mathbb{E}_{p(\tilde{c} \mid (\lambda, z), \mathcal{D})} \left[ \frac{\mathcal{H}[p_{\min}(\cdot \mid \mathcal{D})] - \mathcal{H}[p_{\min}(\cdot \mid \mathcal{D} \cup \langle (\lambda, z), \tilde{c} \rangle)]}{c(\lambda, z)} \right]$$

- Observe performance  $f(\lambda, z)$  and cost  $c(\lambda, z)$  and update models for  $f$  and  $c$

# Entropy Search for Multi-Fidelity Optimization [Klein et al. 2017]

- The entire algorithm iterates the following 2 steps until time is up:

- ① Select  $(\lambda, z)$  by maximizing:

$$u_{ES}(\lambda, z \mid \mathcal{D}) := \mathbb{E}_{p(\tilde{c} \mid (\lambda, z), \mathcal{D})} \left[ \frac{\mathcal{H}[p_{\min}(\cdot \mid \mathcal{D})] - \mathcal{H}[p_{\min}(\cdot \mid \mathcal{D} \cup \langle (\lambda, z), \tilde{c} \rangle)]}{c(\lambda, z)} \right]$$

- ② Observe performance  $f(\lambda, z)$  and cost  $c(\lambda, z)$  and update models for  $f$  and  $c$
- The algorithm originally focussed on data subsets and is therefore dubbed **Fabolas**: “Fast Bayesian Optimization on Large Datasets”

# Entropy Search for Multi-Fidelity Optimization [Klein et al. 2017]

- The entire algorithm iterates the following 2 steps until time is up:

- Select  $(\lambda, z)$  by maximizing:

$$u_{ES}(\lambda, z \mid \mathcal{D}) := \mathbb{E}_{p(\tilde{c} \mid (\lambda, z), \mathcal{D})} \left[ \frac{\mathcal{H}[p_{\min}(\cdot \mid \mathcal{D})] - \mathcal{H}[p_{\min}(\cdot \mid \mathcal{D} \cup \langle (\lambda, z), \tilde{c} \rangle)]}{c(\lambda, z)} \right]$$

- Observe performance  $f(\lambda, z)$  and cost  $c(\lambda, z)$  and update models for  $f$  and  $c$
- The algorithm originally focussed on data subsets and is therefore dubbed **Fabolas**: “Fast Bayesian Optimization on Large Datasets”

## Advantages

- ▶ Conceptually beautiful
- ▶ 1 000-fold speedups for optimizing SVMs on MNIST

# Entropy Search for Multi-Fidelity Optimization [Klein et al. 2017]

- The entire algorithm iterates the following 2 steps until time is up:
  - Select  $(\lambda, z)$  by maximizing:

$$u_{ES}(\lambda, z \mid \mathcal{D}) := \mathbb{E}_{p(\tilde{c} \mid (\lambda, z), \mathcal{D})} \left[ \frac{\mathcal{H}[p_{\min}(\cdot \mid \mathcal{D})] - \mathcal{H}[p_{\min}(\cdot \mid \mathcal{D} \cup ((\lambda, z), \tilde{c}))]}{c(\lambda, z)} \right]$$

- Observe performance  $f(\lambda, z)$  and cost  $c(\lambda, z)$  and update models for  $f$  and  $c$
- The algorithm originally focussed on data subsets and is therefore dubbed **Fabolas**: “Fast Bayesian Optimization on Large Datasets”

## Advantages

- Conceptually beautiful
- 1 000-fold speedups for optimizing SVMs on MNIST

## Disadvantages

- Scalability of GPs is a big problem (limits size of initial design)
- Limited applicability of Gaussian processes

## Questions to Answer for Yourself / Discuss with Friends

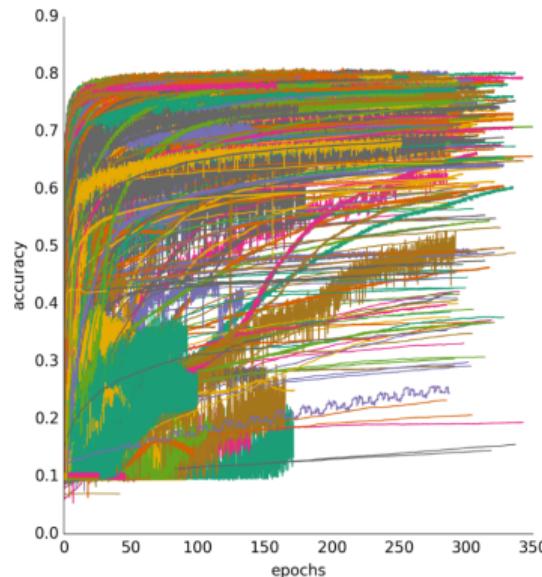
- Discussion. What kind of cost model would you use in Fabolas?
- Discussion. Could one use an acquisition function other than entropy search for Fabolas?

# Speedup Techniques for Hyperparameter Optimization

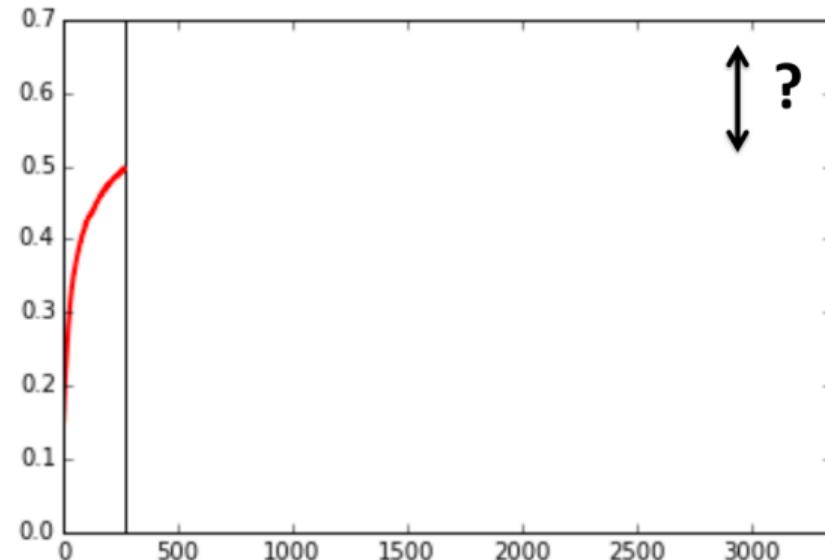
## Predicting Learning Curves

Bernd Bischl    Frank Hutter    Lars Kotthoff  
Marius Lindauer    Joaquin Vanschoren

# Learning Curves

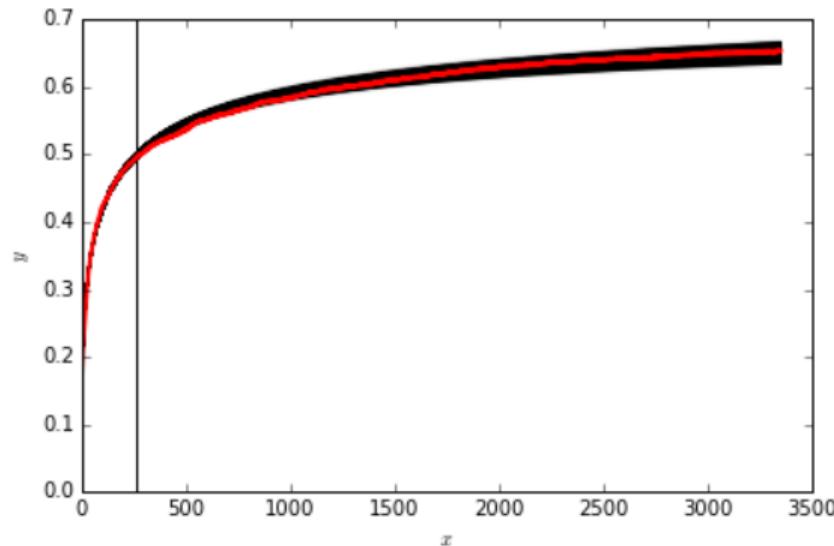


# Learning Curve Predictions



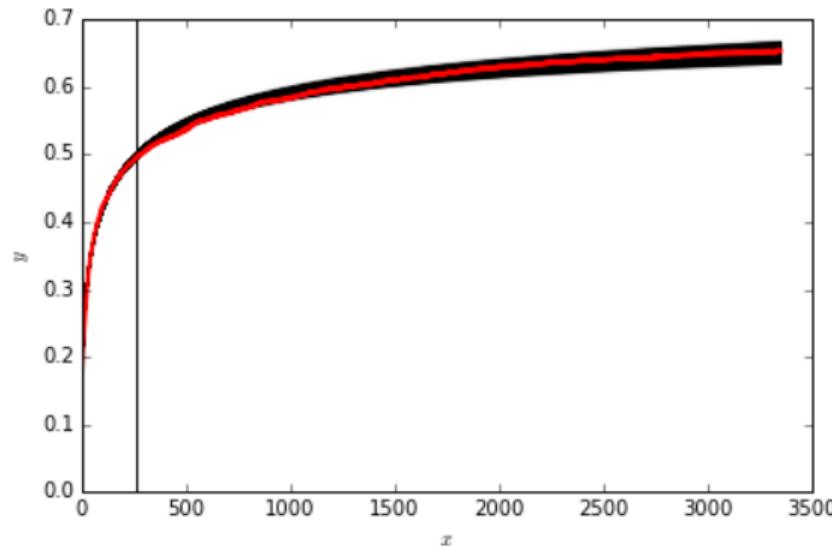
- ① Observe learning curve for the first  $n$  steps (here  $n = 250$ )

# Learning Curve Predictions



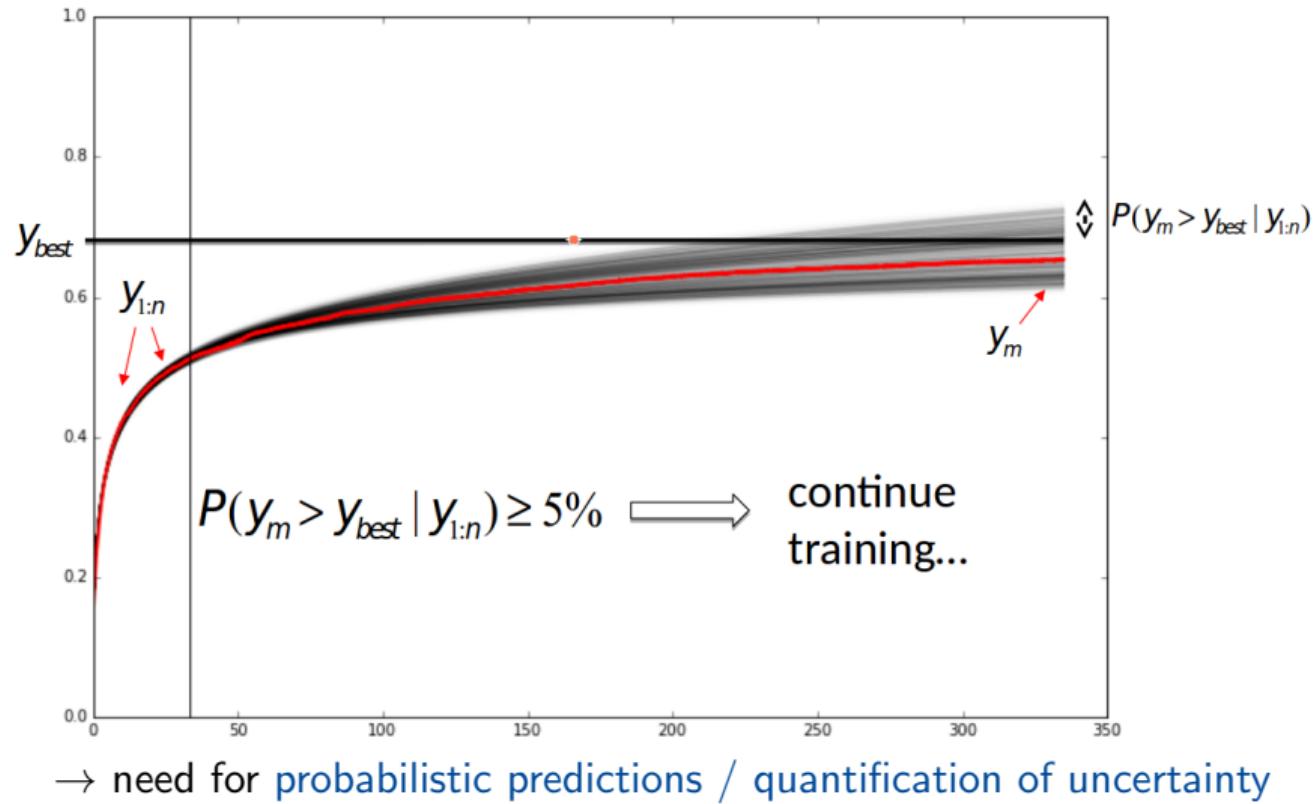
- ① Observe learning curve for the first  $n$  steps (here  $n = 250$ )
- ② **Extrapolation:** fit parametric model on partial learning curve to predict remaining learning curve

# Learning Curve Predictions

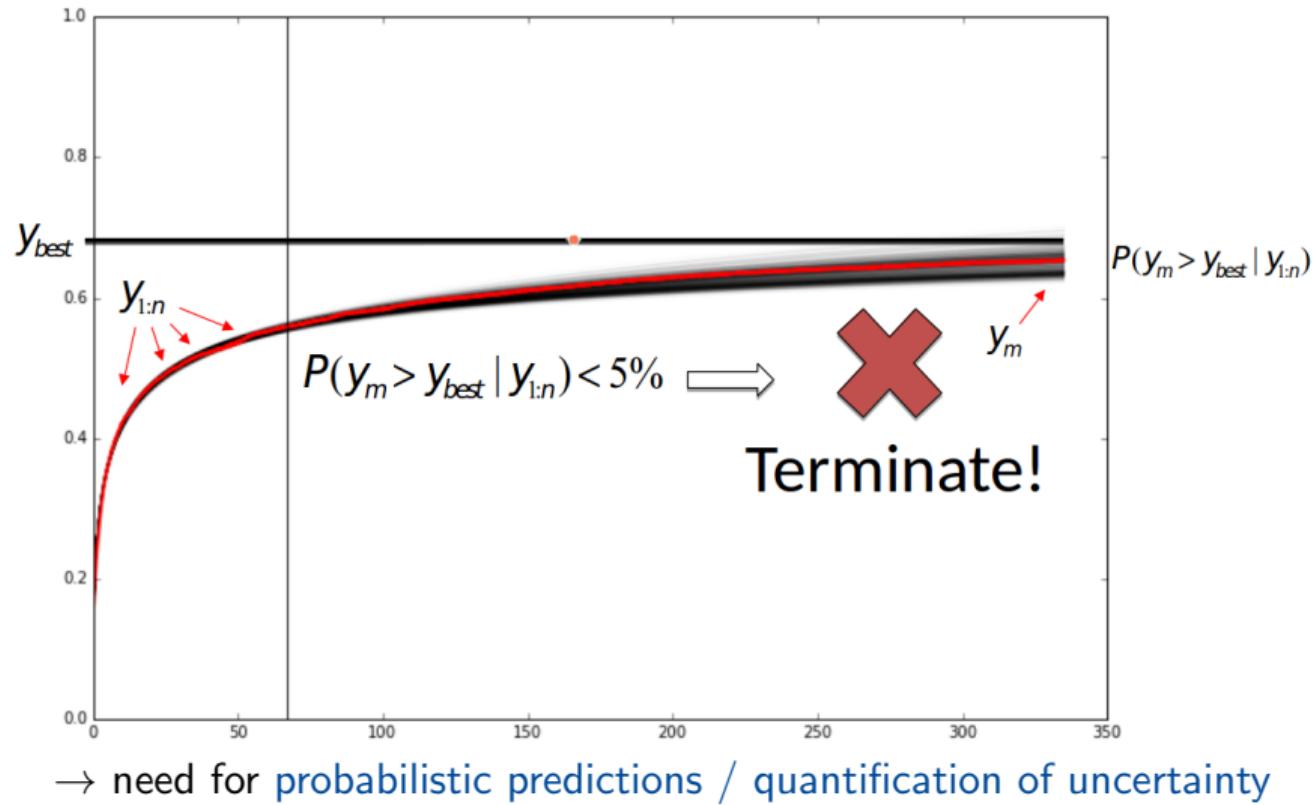


- ① Observe learning curve for the first  $n$  steps (here  $n = 250$ )
- ② **Extrapolation:** fit parametric model on partial learning curve to predict remaining learning curve
  - ▶ Various models can be used (see following slides)

# Learning Curves: Early Termination



# Learning Curves: Early Termination



## Parametric Learning Curves [Domhan et al. 2015]

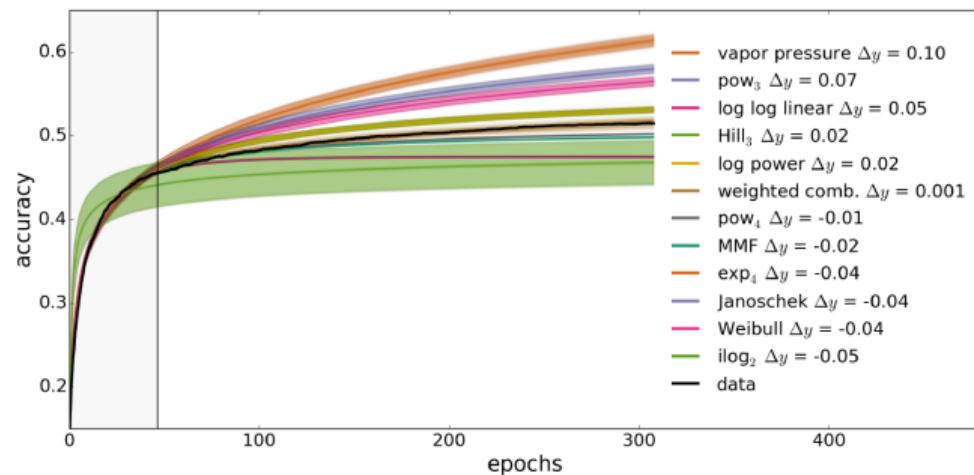
- Use a parametric model  $f_k$  with parameters  $\theta$  to model performance at step  $t$  as:  
 $y_t = f_k(t|\theta) + \epsilon$ , with  $\epsilon \sim \mathcal{N}(0, \sigma^2)$ .

## Parametric Learning Curves [Domhan et al. 2015]

- Use a parametric model  $f_k$  with parameters  $\theta$  to model performance at step  $t$  as:  
 $y_t = f_k(t|\theta) + \epsilon$ , with  $\epsilon \sim \mathcal{N}(0, \sigma^2)$ .
- Linear combination of  $K = 11$  parametric types of models:  
 $f_{comb}(t|\xi) = \sum_{k=1}^K w_k f_k(t|\theta_k)$ , where  $\xi = (w_1, \dots, w_K, \theta_1, \dots, \theta_K, \sigma^2)$

# Parametric Learning Curves [Domhan et al. 2015]

- Use a parametric model  $f_k$  with parameters  $\theta$  to model performance at step  $t$  as:  
 $y_t = f_k(t|\theta) + \epsilon$ , with  $\epsilon \sim \mathcal{N}(0, \sigma^2)$ .
- Linear combination of  $K = 11$  parametric types of models:  
 $f_{comb}(t|\xi) = \sum_{k=1}^K w_k f_k(t|\theta_k)$ , where  $\xi = (w_1, \dots, w_K, \theta_1, \dots, \theta_K, \sigma^2)$

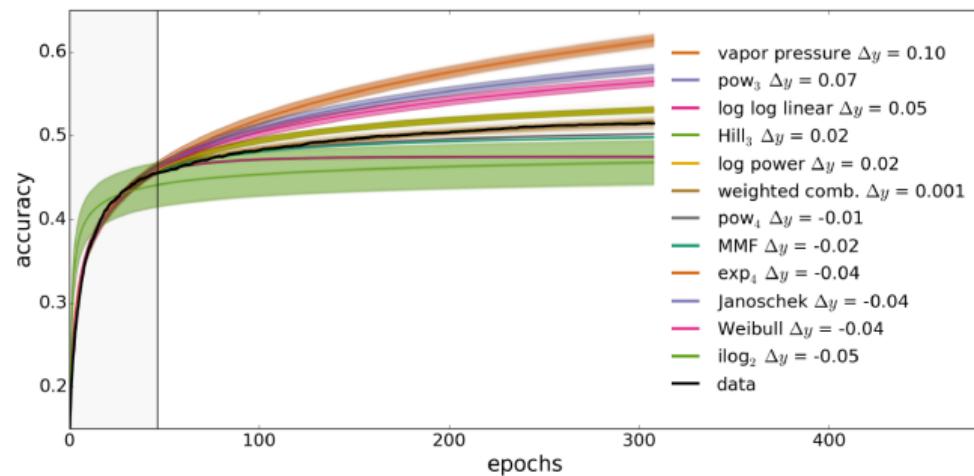


Reference name	Formula
vapor pressure	$\exp(a + \frac{b}{x} + c \log(x))$
pow <sub>3</sub>	$c - ax^{-\alpha}$
log log linear	$\log(a \log(x) + b)$
Hill <sub>3</sub>	$\frac{y_{\max} x^{\eta}}{\kappa^{\eta} + x^{\eta}}$
log power	$\frac{a}{1 + (\frac{x}{e^b})^c}$
pow <sub>4</sub>	$c - (ax + b)^{-\alpha}$
MMF	$\alpha - \frac{\alpha - \beta}{1 + (\kappa x)^{\delta}}$
exp <sub>4</sub>	$c - e^{-ax^{\alpha} + b}$
Janoschek	$\alpha - (\alpha - \beta)e^{-\kappa x^{\delta}}$
Weibull	$\alpha - (\alpha - \beta)e^{-(\kappa x)^{\delta}}$
ilog <sub>2</sub>	$c - \frac{a}{\log x}$

$K = 11$  parametric families for modelling learning curves

# Parametric Learning Curves [Domhan et al. 2015]

- Use a parametric model  $f_k$  with parameters  $\theta$  to model performance at step  $t$  as:  
 $y_t = f_k(t|\theta) + \epsilon$ , with  $\epsilon \sim \mathcal{N}(0, \sigma^2)$ .
- Linear combination of  $K = 11$  parametric types of models:  
 $f_{comb}(t|\xi) = \sum_{k=1}^K w_k f_k(t|\theta_k)$ , where  $\xi = (w_1, \dots, w_K, \theta_1, \dots, \theta_K, \sigma^2)$

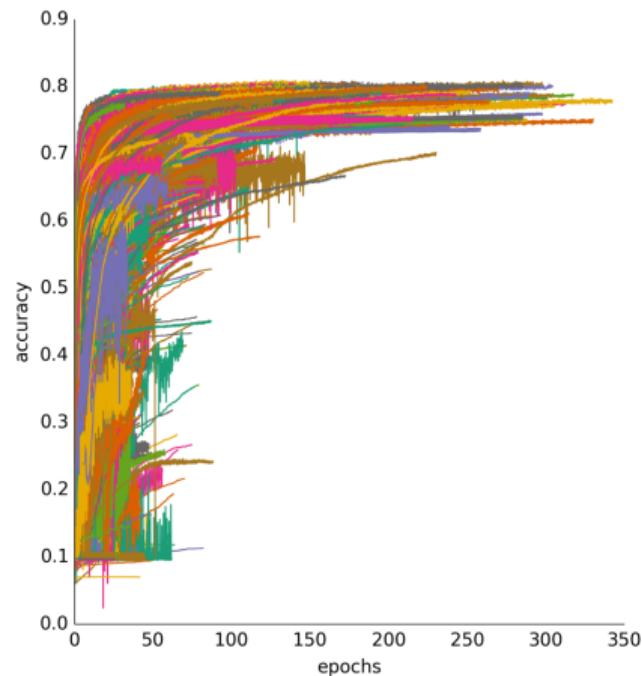
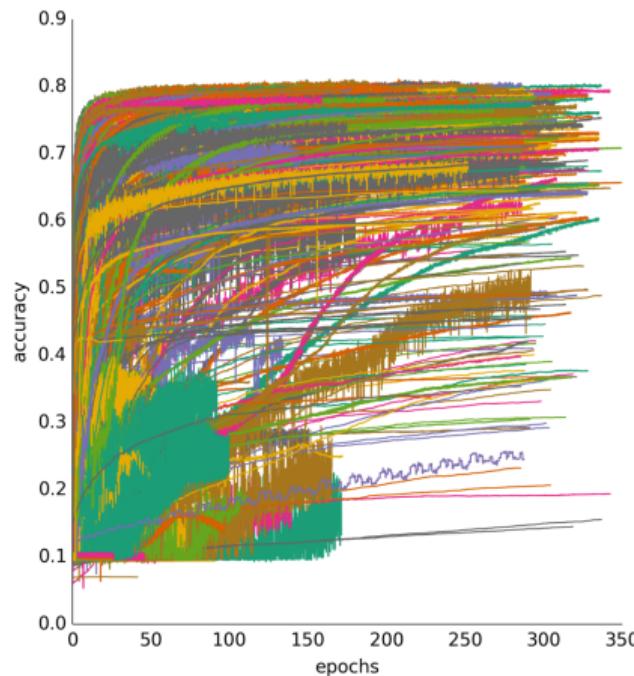


Reference name	Formula
vapor pressure	$\exp(a + \frac{b}{x} + c \log(x))$
pow3	$c - ax^{-\alpha}$
log log linear	$\log(a \log(x) + b)$
Hill3	$\frac{y_{\max} x^{\eta}}{\kappa^{\eta} + x^{\eta}}$
log power	$\frac{a}{1 + (\frac{x}{e^b})^c}$
pow4	$c - (ax + b)^{-\alpha}$
MMF	$\alpha - \frac{\alpha - \beta}{1 + (\kappa x)^{\delta}}$
exp4	$c - e^{-ax^{\alpha} + b}$
Janoschek	$\alpha - (\alpha - \beta)e^{-\kappa x^{\delta}}$
Weibull	$\alpha - (\alpha - \beta)e^{-(\kappa x)^{\delta}}$
ilog2	$c - \frac{a}{\log x}$

$K = 11$  parametric families for modelling learning curves

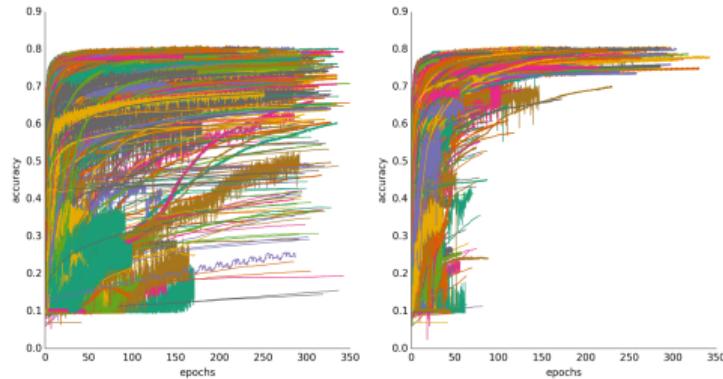
- Use Markov Chain Monte Carlo sampling of  $\xi$  to obtain uncertainties

# Predictive Termination



All learning curves vs. learning curves with early termination

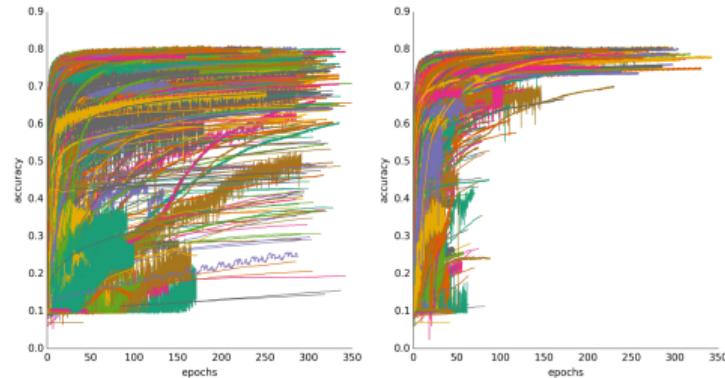
# Predictive Termination



All learning curves vs. learning curves with early termination

- Disadvantages of this model?

# Predictive Termination

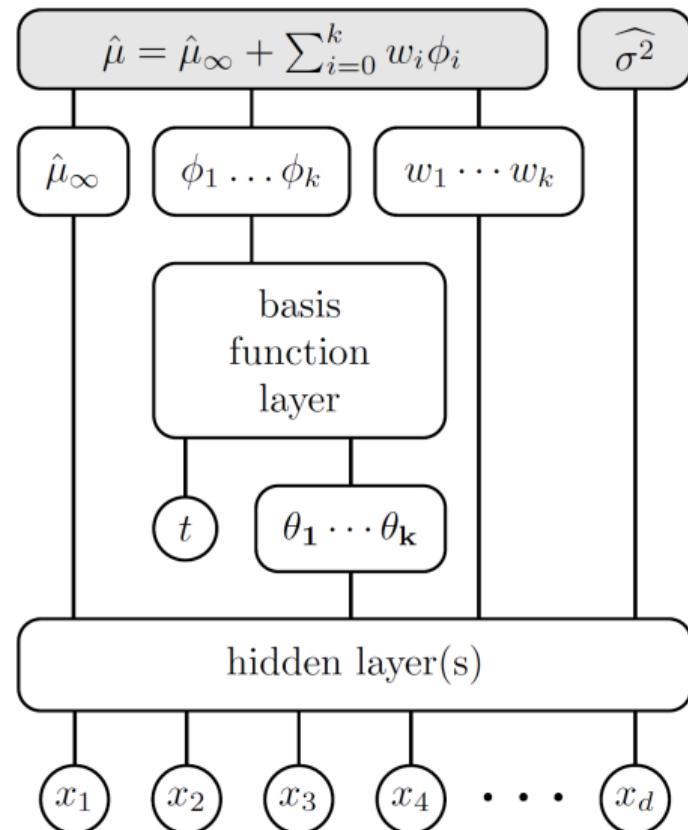


All learning curves vs. learning curves with early termination

- Disadvantages of this model?
  - ▶ Relies on manually-selected parametric families of curves
  - ▶ Does not take into account hyperparameters used
    - can't learn across hyperparameters

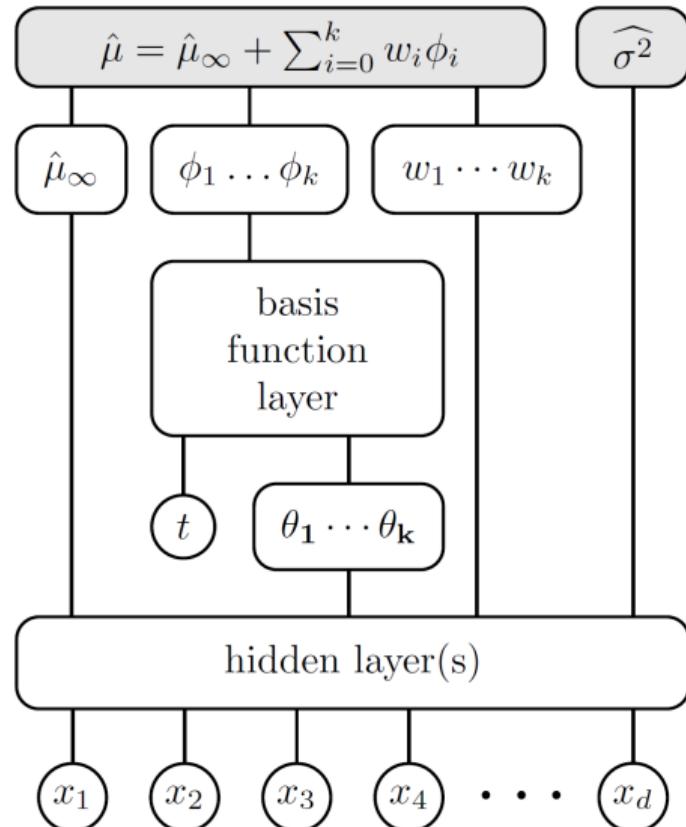
## LC-Net [Klein et al. 2017]

- Make a layer out of the parametric learning curves by Domhan et al.
- Also support hyperparameters as inputs (in the figure denoted by  $x_1, \dots, x_d$ )



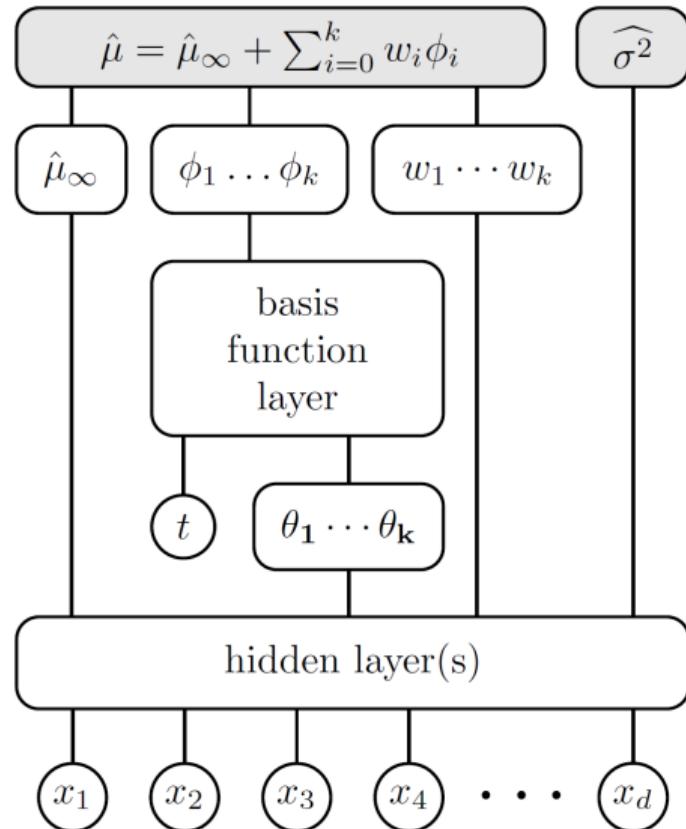
## LC-Net [Klein et al. 2017]

- Make a layer out of the parametric learning curves by Domhan et al.
- Also support hyperparameters as inputs (in the figure denoted by  $x_1, \dots, x_d$ )
- Disadvantages of this model?



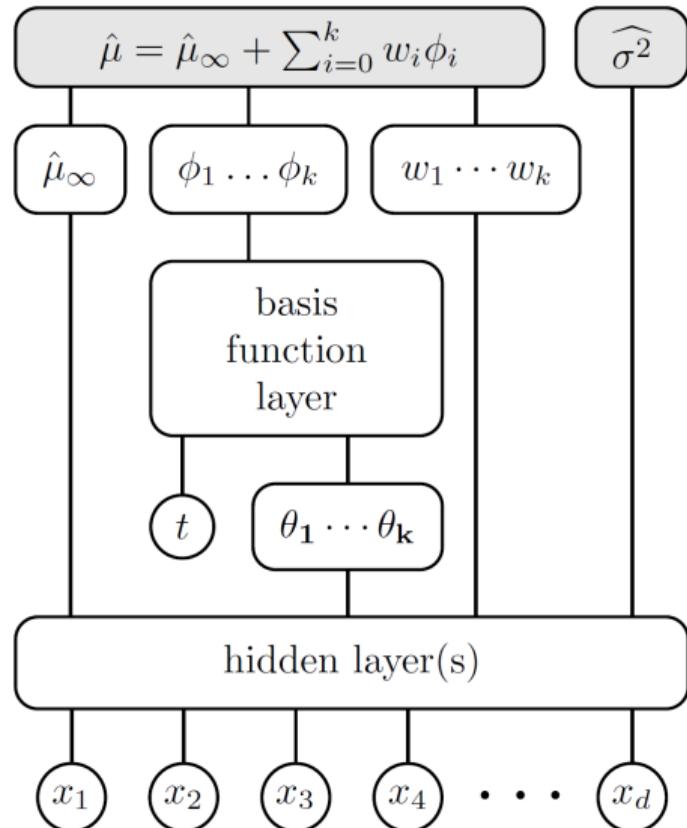
## LC-Net [Klein et al. 2017]

- Make a layer out of the parametric learning curves by Domhan et al.
- Also support hyperparameters as inputs (in the figure denoted by  $x_1, \dots, x_d$ )
- Disadvantages of this model?
  - ▶ Relies on manually-selected parametric families of curves
  - ▶ Cannot quickly integrate new information from extending the current curve (or from new runs)



## LC-Net [Klein et al. 2017]

- Make a layer out of the parametric learning curves by Domhan et al.
- Also support hyperparameters as inputs (in the figure denoted by  $x_1, \dots, x_d$ )
- Disadvantages of this model?
  - ▶ Relies on manually-selected parametric families of curves
  - ▶ Cannot quickly integrate new information from extending the current curve (or from new runs)
  - ▶ Also, the model is very hard to train

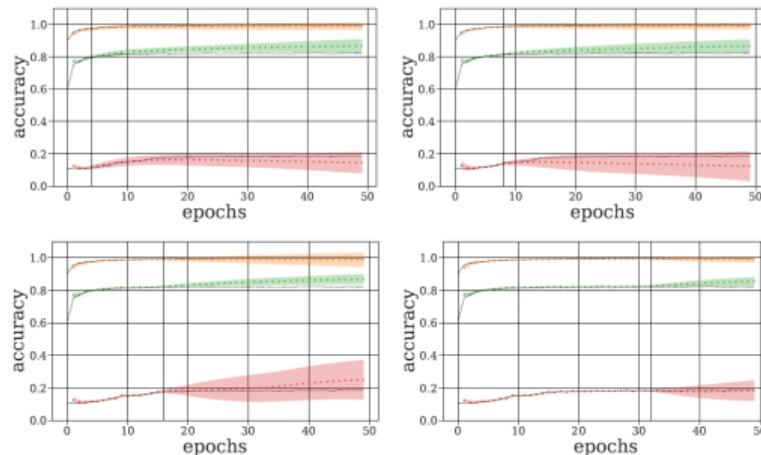


## Sequence Models (e.g., Bayesian RNN) [Gargiani et al. 2019]

- Learning curves are **sequences**
  - ▶ Previous models don't treat them like this
  - ▶ We can use an RNN (in particular, an LSTM) to predict the next value from a given sequence
  - ▶ We can use variational dropout to obtain uncertainty estimates:

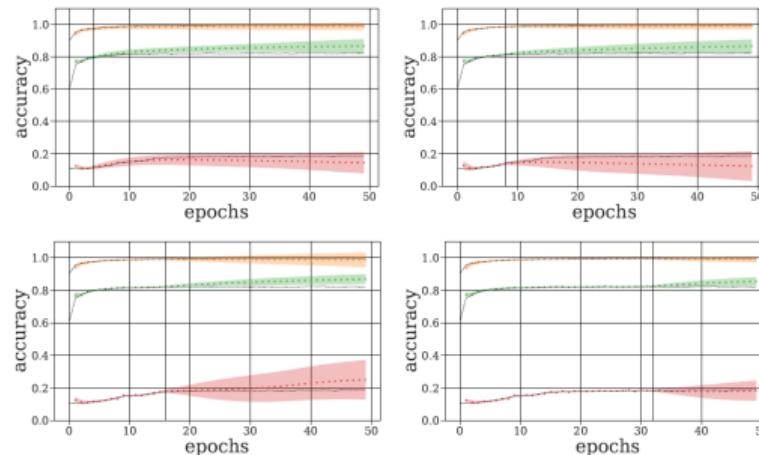
# Sequence Models (e.g., Bayesian RNN) [Gargiani et al. 2019]

- Learning curves are **sequences**
  - ▶ Previous models don't treat them like this
  - ▶ We can use an RNN (in particular, an LSTM) to predict the next value from a given sequence
  - ▶ We can use variational dropout to obtain uncertainty estimates:



# Sequence Models (e.g., Bayesian RNN) [Gargiani et al. 2019]

- Learning curves are **sequences**
  - ▶ Previous models don't treat them like this
  - ▶ We can use an RNN (in particular, an LSTM) to predict the next value from a given sequence
  - ▶ We can use variational dropout to obtain uncertainty estimates:



Note: we can also use a simpler model

- E.g., a random forest to map from a fixed-size window to the next value

## Compare: Baker et al, 2017 [Baker et al. 2018]

- Idea: map from configurations (including architectural hyperparameters) and partial learning curves to the final performance
- Advantages
  - ▶ Much simpler idea than all the approaches just discussed:  
no need to model the entire learning curve
  - ▶ Much easier to implement
- Disadvantage?

## Compare: Baker et al, 2017 [Baker et al. 2018]

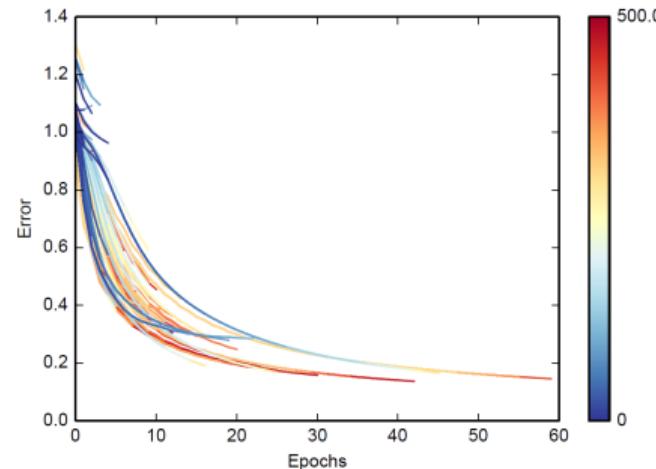
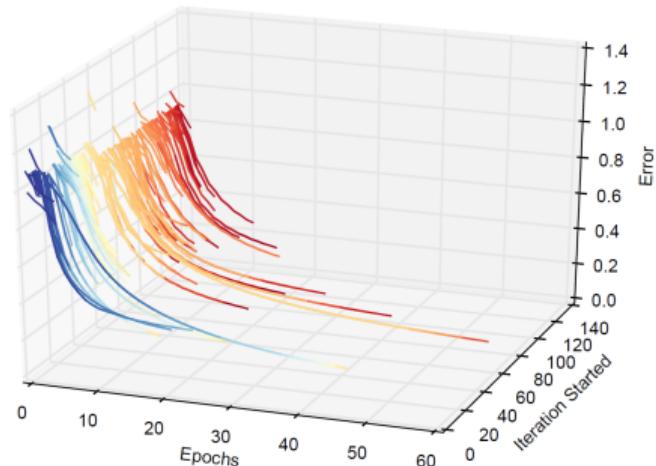
- Idea: map from configurations (including architectural hyperparameters) and partial learning curves to the final performance
- Advantages
  - ▶ Much simpler idea than all the approaches just discussed:  
no need to model the entire learning curve
  - ▶ Much easier to implement
- Disadvantage? → requires many (e.g., 100) fully-evaluated learning curves as training data
  - ▶ After 100 full function evaluations we want to be pretty much converged in practice
  - ▶ But definitely helpful for speeding up RL

## Freeze-Thaw Bayesian Optimization [Swersky et al. 2014]

- Use a Gaussian process with inputs  $\lambda$  and  $t$ ; special kernel for  $t$
- For  $N$  configurations and  $T$  epochs each:  $O(N^3t^3) \rightarrow$  approximation
- Iteratively: either extend existing configuration or try new one

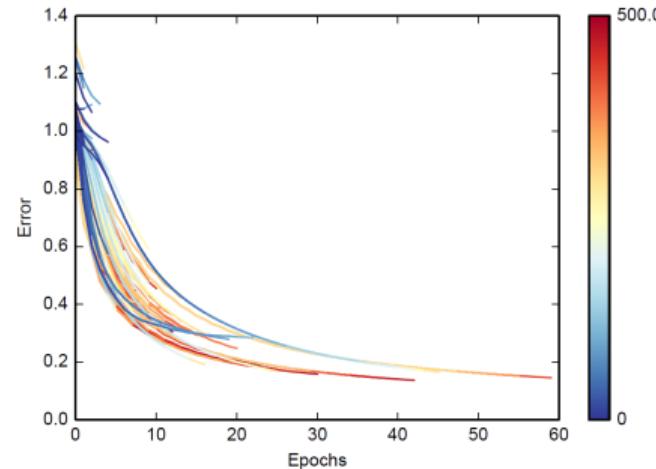
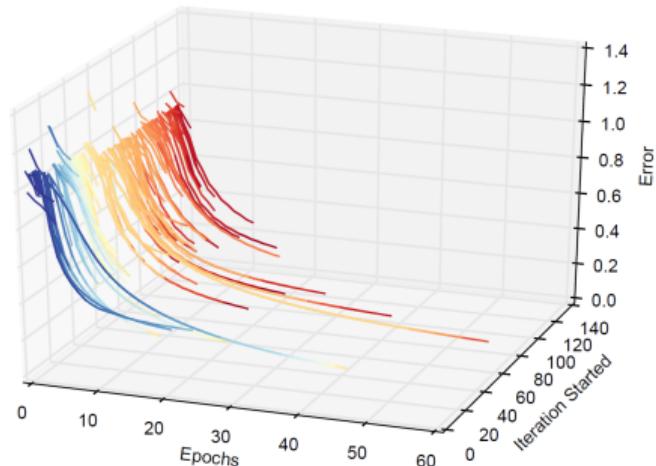
# Freeze-Thaw Bayesian Optimization [Swersky et al. 2014]

- Use a Gaussian process with inputs  $\lambda$  and  $t$ ; special kernel for  $t$
- For  $N$  configurations and  $T$  epochs each:  $O(N^3t^3) \rightarrow$  approximation
- Iteratively: either extend existing configuration or try new one
- Result for probabilistic matrix factorization:



# Freeze-Thaw Bayesian Optimization [Swersky et al. 2014]

- Use a Gaussian process with inputs  $\lambda$  and  $t$ ; special kernel for  $t$
- For  $N$  configurations and  $T$  epochs each:  $O(N^3t^3) \rightarrow$  approximation
- Iteratively: either extend existing configuration or try new one
- Result for probabilistic matrix factorization:



- Unfortunately, no results for DNNs; no code available

## Questions to Answer for Yourself / Discuss with Friends

- **Repetition.** List all learning curve prediction methods you recall, along with their pros and cons.
- **Discussion.** Could predictive termination cut off evaluations early that would turn out to be the best?
- **Discussion.** How would you determine a learning curve prediction method's own hyperparameters (such as the 5% for early learning curve termination), in practice?
- **Discussion.** How could we exploit additional side information we gain about the learning curve, such as, e.g., statistics for the size of the gradients and activations over time?

# Speedup Techniques for Hyperparameter Optimization

## Success Stories and Practical Recommendations

Bernd Bischl    Frank Hutter    Lars Kotthoff  
Marius Lindauer    Joaquin Vanschoren

# Large-scale Meta-Learning for HPO in Industry (Facebook)

- Facebook has an internal self-service machine learning (ML) system
  - ▶ Non-ML departments can integrate highly optimized ML models into their workflow
  - ▶ Hyperparameters of the ML models are optimized with Bayesian optimization

# Large-scale Meta-Learning for HPO in Industry (Facebook)

- Facebook has an internal self-service machine learning (ML) system
  - ▶ Non-ML departments can integrate highly optimized ML models into their workflow
  - ▶ Hyperparameters of the ML models are optimized with Bayesian optimization
- Training data for the models changes over time
  - ▶ Hyperparameters are constantly re-optimized
  - ▶ For efficiency: meta-learning Bayesian optimization, as described in [Feurer et al. 2018]

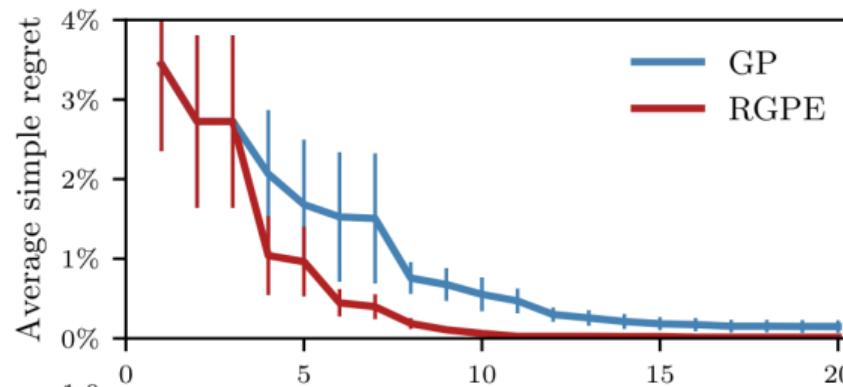
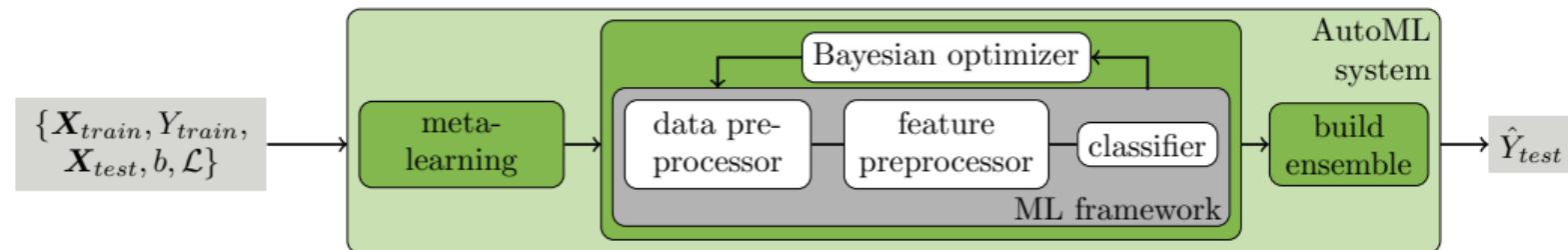


Figure: Bayesian optimization with meta-learning (RGPE) vs. vanilla Bayesian optimization (GP)

# Auto-sklearn [Feurer et al. 2015]

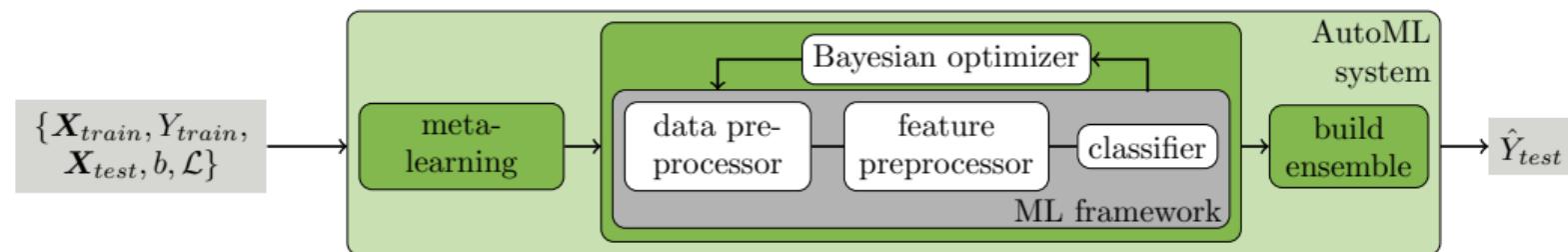
Extension of Auto-WEKA with focus on speed improvements and robustness:



- Uses meta-learning to warmstart Bayesian optimization
- Won the 1st AutoML challenge

# Auto-sklearn [Feurer et al. 2015]

Extension of Auto-WEKA with focus on speed improvements and robustness:



- Uses meta-learning to warmstart Bayesian optimization
- Won the 1st AutoML challenge
- Open source (BSD) and trivial to use



```
>>> import autosklearn.classification  
>>> cls = autosklearn.classification.AutoSklearnClassifier()  
>>> cls.fit(X_train, y_train)  
>>> predictions = cls.predict(X_test)
```

Available at <https://automl.github.io/auto-sklearn>; frequently used in industry

## BOHB [Falkner et al. 2018]

- Robust and efficient
- Only published in 2018, adopted by the community very quickly

Cited by 129



### Scholar articles

[BOHB: Robust and efficient hyperparameter optimization at scale](#)  
S Falkner, A Klein, F Hutter - arXiv preprint arXiv:1807.01774, 2018  
[Cited by 129](#) [Related articles](#) [All 8 versions](#)

- Available at <https://github.com/automl/HpBandSter>

Unwatch ▾

23

Star

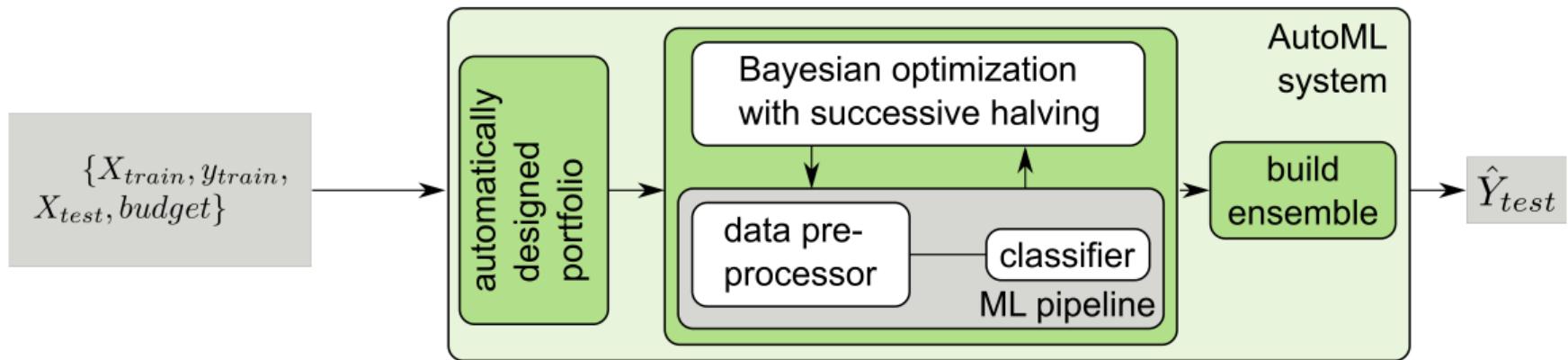
371

Fork

80

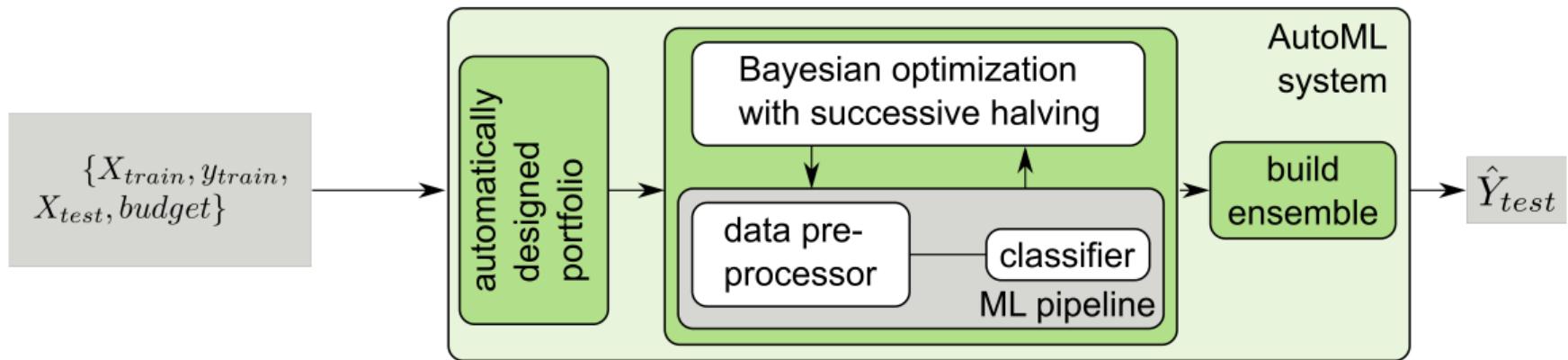
# PoSH-Auto-sklearn [Feurer et al. 2018]

Idea: integrate warmstarting and a BOHB-like approach for Auto-sklearn



# PoSH-Auto-sklearn [Feurer et al. 2018]

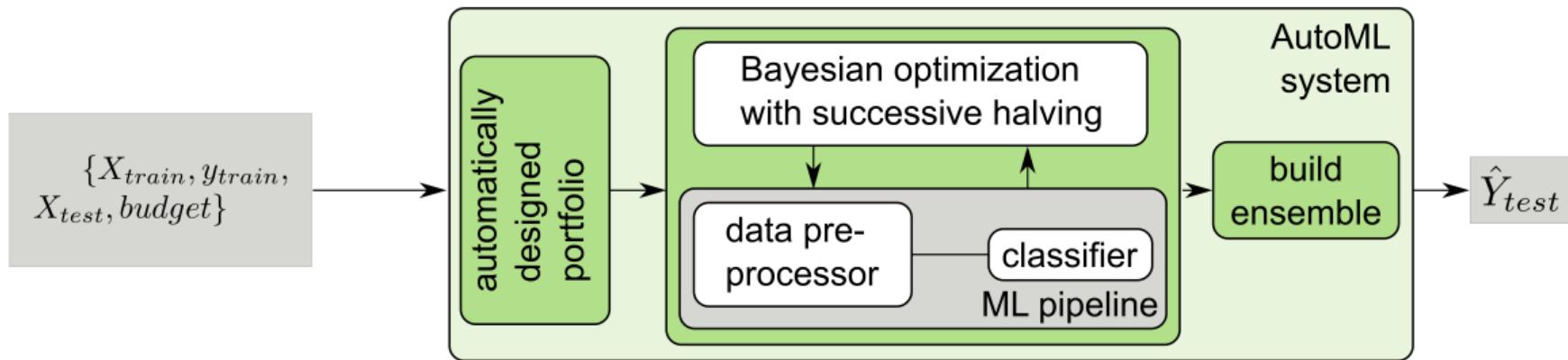
Idea: integrate warmstarting and a BOHB-like approach for Auto-sklearn



- Uses task-independent meta-learning to warmstart Bayesian optimization
  - ▶ Therefore, no need for (potentially unreliable) meta-features

# PoSH-Auto-sklearn [Feurer et al. 2018]

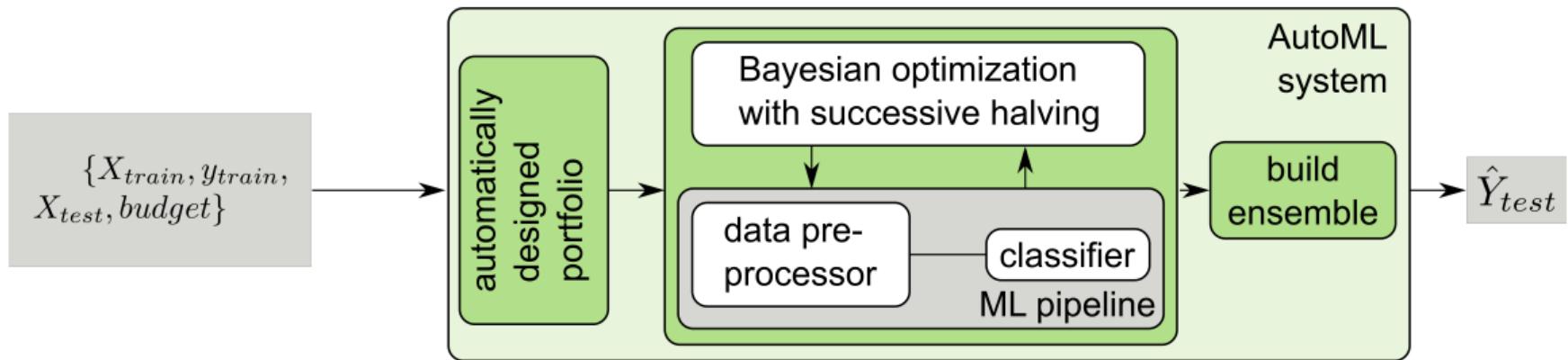
Idea: integrate warmstarting and a BOHB-like approach for Auto-sklearn



- Uses task-independent meta-learning to warmstart Bayesian optimization
  - ▶ Therefore, no need for (potentially unreliable) meta-features
- Uses successive halving to quickly go through proposed configurations
  - ▶ Therefore, scales better to larger datasets

# PoSH-Auto-sklearn [Feurer et al. 2018]

Idea: integrate warmstarting and a BOHB-like approach for Auto-sklearn



- Uses task-independent meta-learning to warmstart Bayesian optimization
  - ▶ Therefore, no need for (potentially unreliable) meta-features
- Uses successive halving to quickly go through proposed configurations
  - ▶ Therefore, scales better to larger datasets
- Followed by BOHB-like approach (uses successive halving instead of Hyperband)
- Won the 2nd AutoML challenge

# Auto-sklearn 2.0

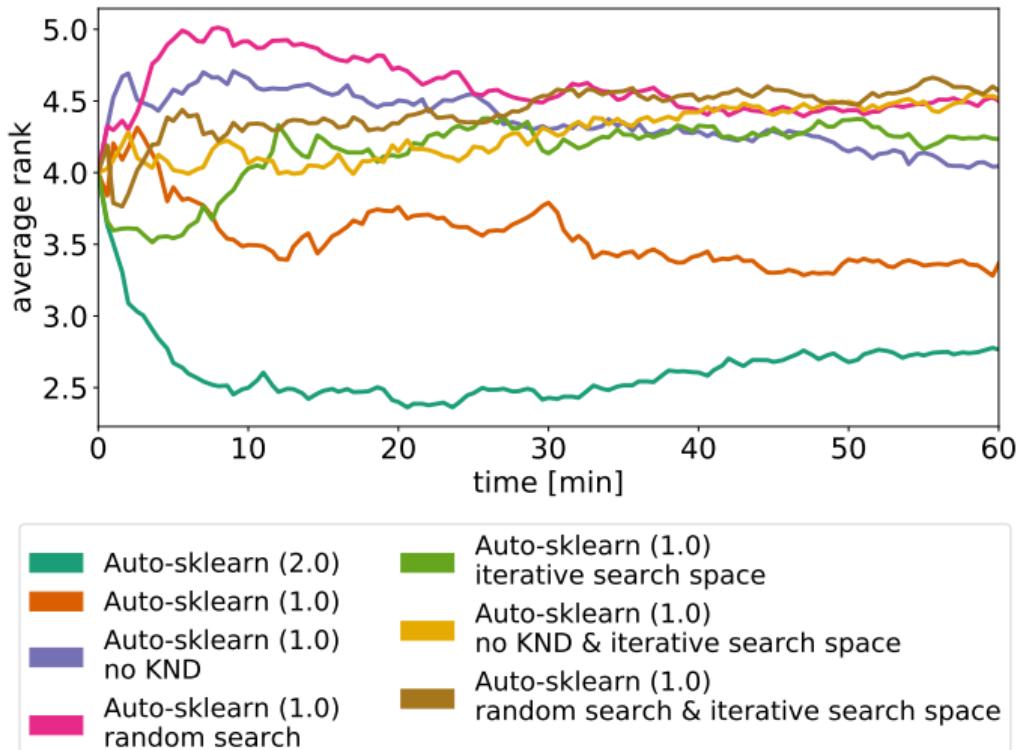
- Idea: automatically choose on a per-dataset basis
  - ▶ holdout or cross-validation
  - ▶ optimization on the full budget or optimization with successive halving

# Auto-sklearn 2.0

- Idea: automatically choose on a per-dataset basis
  - ▶ holdout or cross-validation
  - ▶ optimization on the full budget or optimization with successive halving
- Can be done based on algorithm selection

# Auto-sklearn 2.0

- Idea: automatically choose on a per-dataset basis
  - ▶ holdout or cross-validation
  - ▶ optimization on the full budget or optimization with successive halving
- Can be done based on algorithm selection
- Substantial improvements over Auto-sklearn 1.0
  - ▶ 5× reduction of average error
  - ▶ 6× speedup (same performance in 10 minutes as Auto-sklearn 1.0 in 1 hour)



# Practical Recommendations Which HPO Method to Use [Feurer and Hutter. 2019]

- If multiple fidelities available: BOHB [Falkner et al. 2018]
- Otherwise
  - ▶ Low-dimensional continuous parameter space:
    - ★ GP-based BO, e.g., Spearmint [Snoek et al. 2012]
  - ▶ High-dimensional discrete parameter space:
    - ★ RF-based BO, e.g., SMAC [Hutter et al. 2011]
  - ▶ Purely continuous, cheap function evaluations:
    - ★ CMA-ES [Hansen et al. 2001]; evaluated for HPO by [Loshchilov and Hutter. 2016]

# Practical Recommendations Which HPO Method to Use [Feurer and Hutter. 2019]

- If multiple fidelities available: BOHB [Falkner et al. 2018]
- Otherwise
  - ▶ Low-dimensional continuous parameter space:
    - ★ GP-based BO, e.g., Spearmint [Snoek et al. 2012]
  - ▶ High-dimensional discrete parameter space:
    - ★ RF-based BO, e.g., SMAC [Hutter et al. 2011]
  - ▶ Purely continuous, cheap function evaluations:
    - ★ CMA-ES [Hansen et al. 2001]; evaluated for HPO by [Loshchilov and Hutter. 2016]
- Just submitted: **DEHB** combines differential evolution and Hyperband and largely dominates BOHB. Especially good for high dimensions.

## Questions to Answer for Yourself / Discuss with Friends

- Repetition. Discuss several success stories of speeding up Bayesian optimization.
- Repetition. What differs between Auto-sklearn 1.0 and Auto-sklearn 2.0?

# Speedup Techniques for Hyperparameter Optimization

## Further Reading

Bernd Bischl    [Frank Hutter](#)    Lars Kotthoff  
Marius Lindauer    Joaquin Vanschoren

## Further Reading

Survey on hyperparameter optimization: [Feurer and Hutter 2019]