Automated Machine Learning (AutoML): A Tutorial

Matthias Feurer
University of Freiburg
feurerm@cs.uni-freiburg.de
@__mfeurer__

Thomas Elsken
Bosch Center for Artificial Intelligence
& University of Freiburg
Thomas.Elsken@de.bosch.com

@automlfreiburg

Slides based on material from Frank Hutter and Joaquin Vanschoren
Tutorial based on Chapters 1-3 of the book *Automated Machine Learning*
Slides available at automl.org/events/tutorials -> AutoML Tutorial
(all references are clickable links)

Motivation: Successes of Deep Learning

Computer vision in self-driving cars

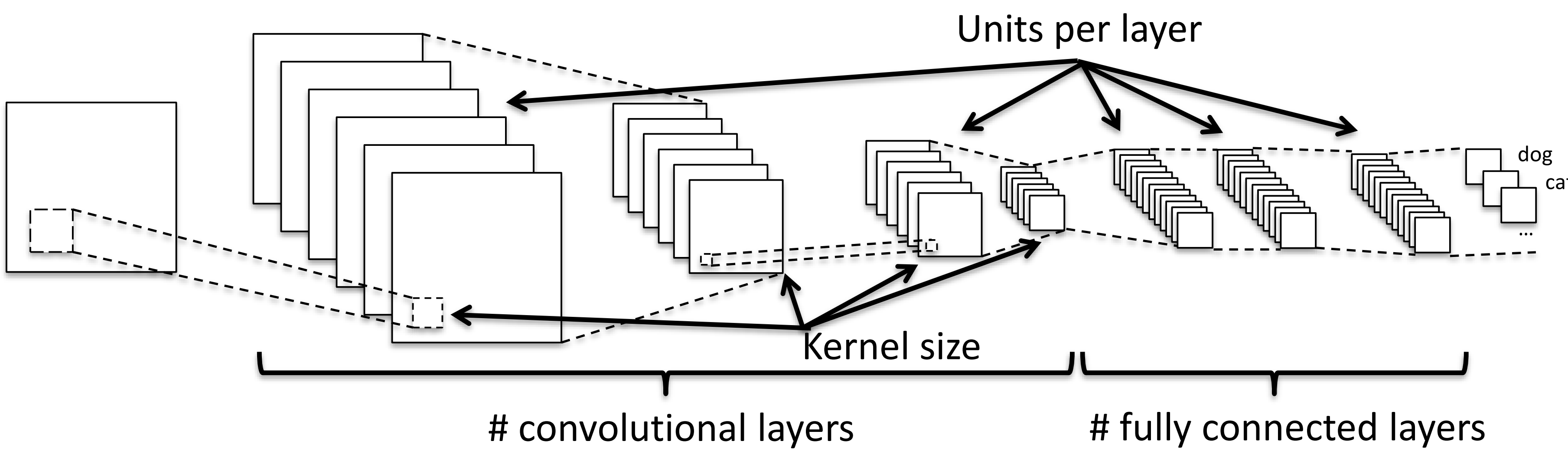
Speech recognition

Reasoning in games

One Problem of Deep Learning

Performance is very **sensitive** to **many hyperparameters**

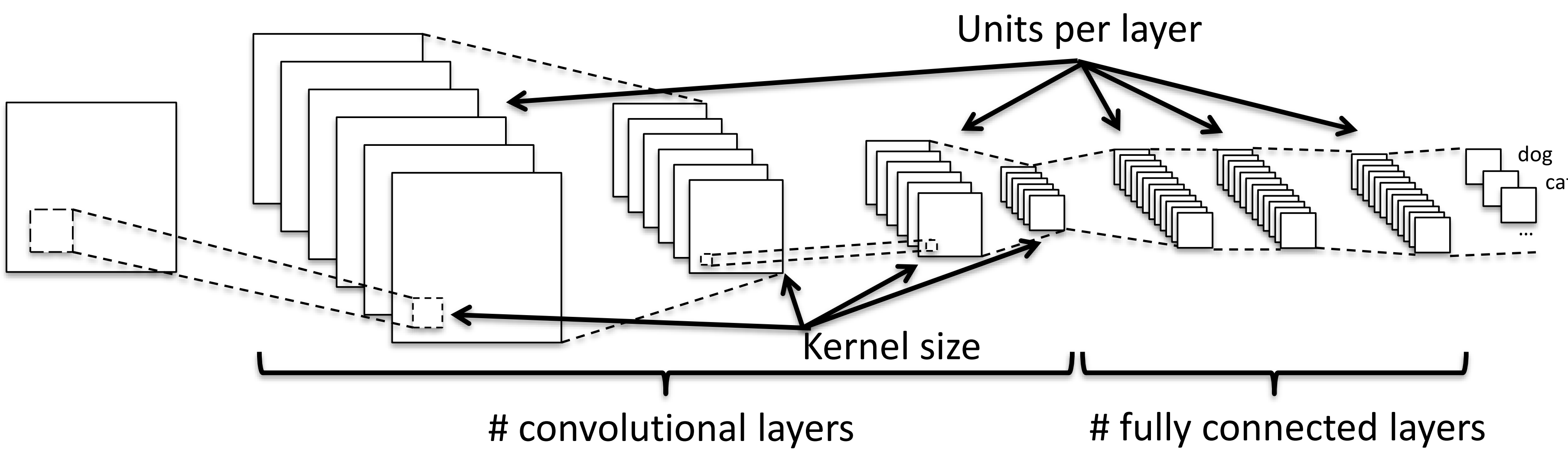
- Architectural hyperparameters



One Problem of Deep Learning

Performance is very **sensitive** to **many hyperparameters**

- Architectural hyperparameters



- Optimization algorithm, learning rates, momentum, batch normalization, batch sizes, dropout rates, weight decay, data augmentation, …

→ Easily 20-50 design decisions

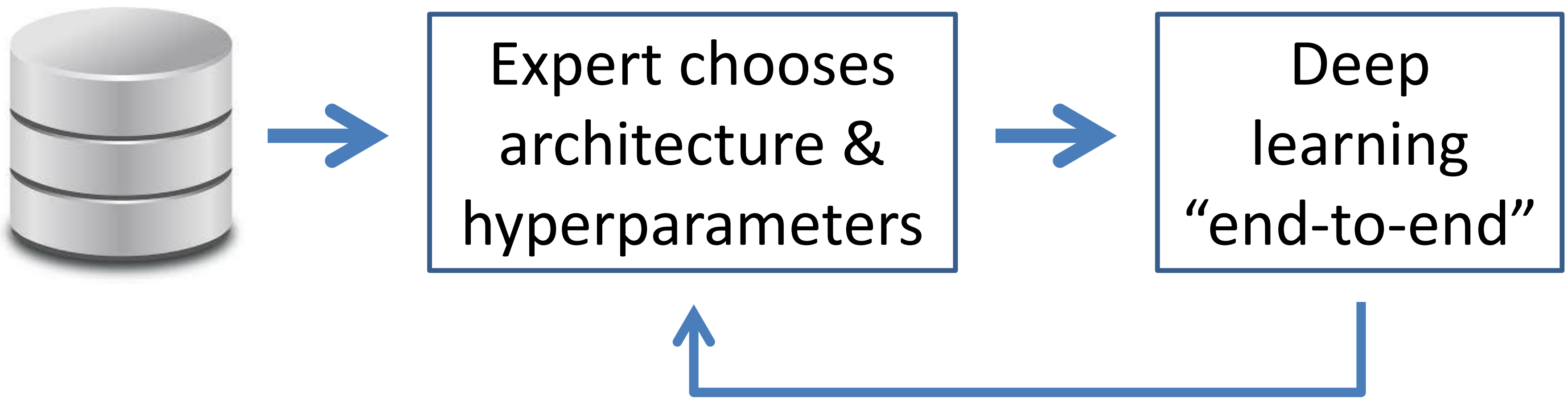
Deep Learning and AutoML

Current deep learning practice

→ Expert chooses architecture & hyperparameters → Deep learning "end-to-end"

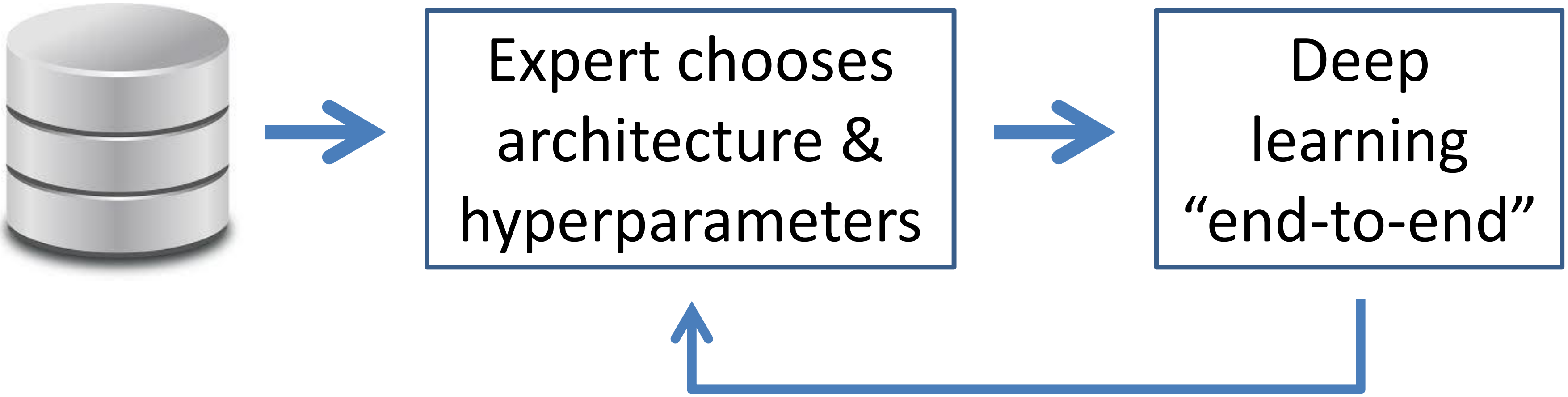
Deep Learning and AutoML

Current deep learning practice

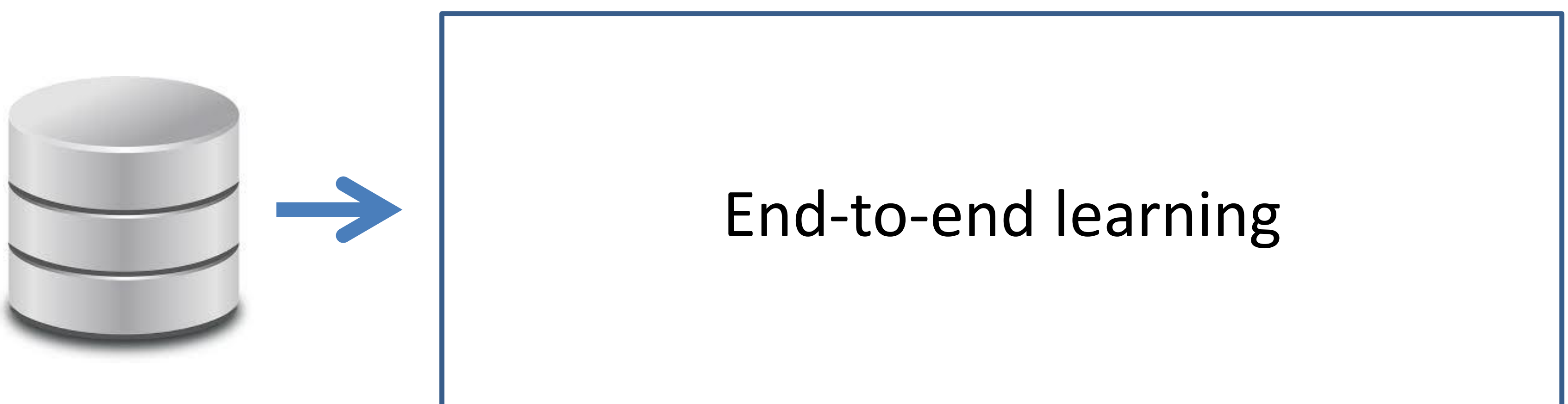


Deep Learning and AutoML

Current deep learning practice

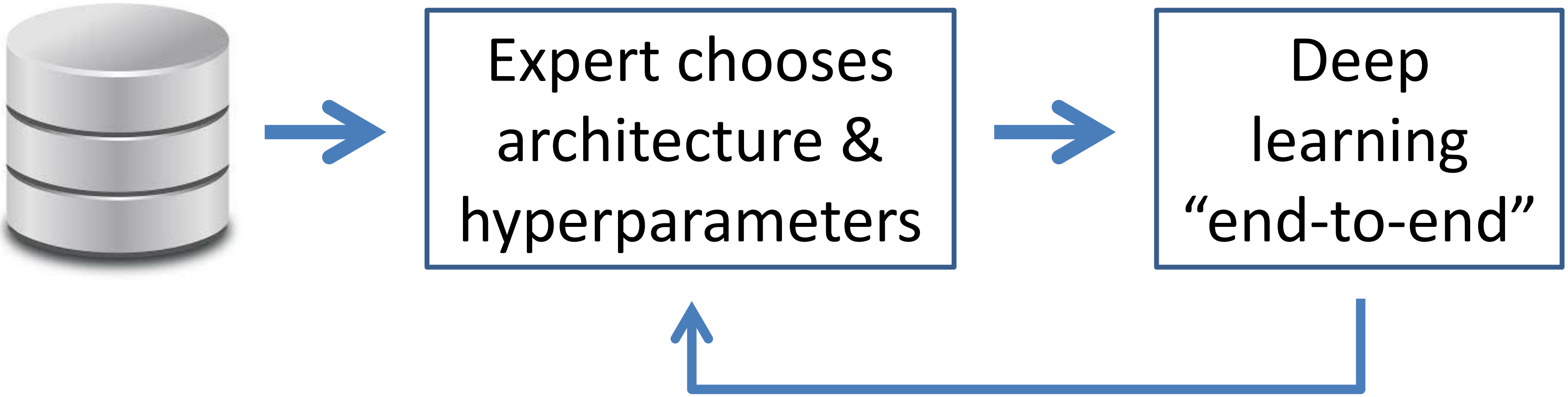


AutoML: true end-to-end learning

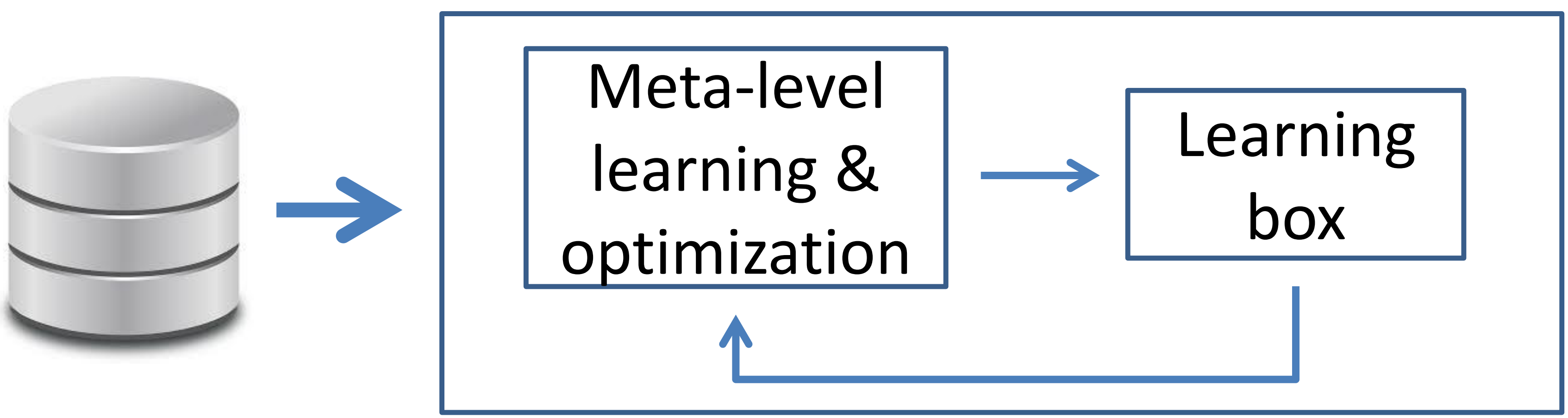


Deep Learning and AutoML

Current deep learning practice



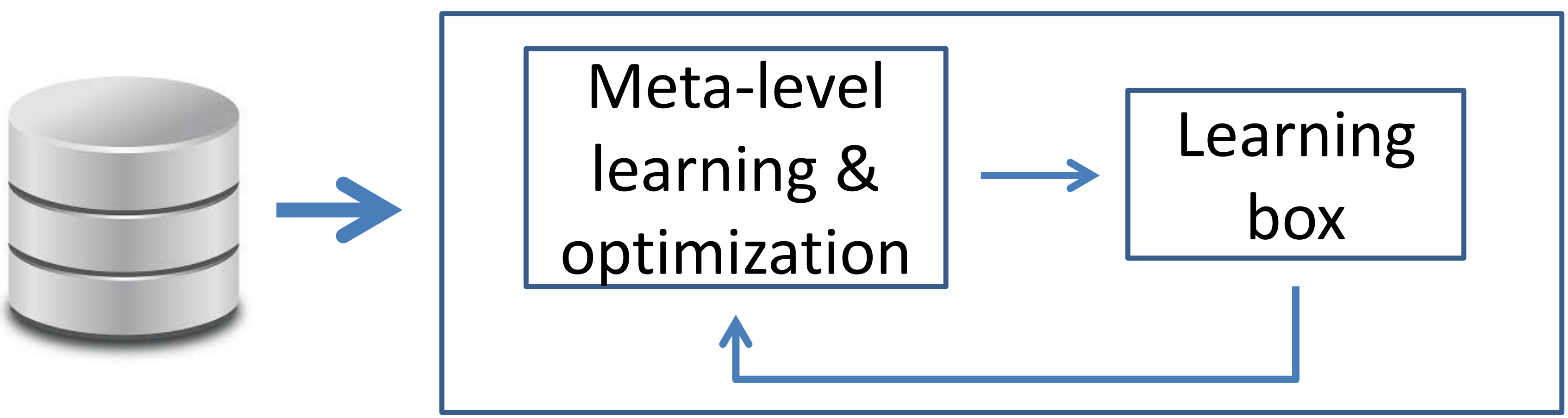
AutoML: true end-to-end learning



Learning box is not restricted to deep learning

- Traditional machine learning pipeline:
  - Clean & preprocess the data
  - Select / engineer better features
  - Select a model family
  - Set the hyperparameters
  - Construct ensembles of models
  - …

AutoML: true end-to-end learning



Outline

Part 1: General AutoML (by me, now)

1. AutoML by Hyperparameter Optimization
2. Black-box Hyperparameter Optimization
3. Beyond black-box optimization
4. Meta-learning
5. Examples of AutoML
6. Open issues and future work
7. Wrap-up & Conclusion

Part 2: Neural Architecture Search & Meta-Learning
(by Thomas Elsken, after the break)

Outline

Part 1: General AutoML

1. AutoML by Hyperparameter Optimization
2. Black-box Hyperparameter Optimization
3. Beyond black-box optimization
4. Meta-learning
5. Examples of AutoML
6. Open issues and future work
7. Wrap-up & Conclusion

Part 2: Neural Architecture Search & Meta-Learning

Hyperparameter Optimization

Definition: Hyperparameter Optimization (HPO)

Let

- $\boldsymbol{\lambda}$ be the hyperparameters of a ML algorithm A with domain $\boldsymbol{\Lambda}$,
- $\mathcal{L}(A_{\boldsymbol{\lambda}}, D_{train}, D_{valid})$ denote the loss of A, using hyperparameters $\boldsymbol{\lambda}$ trained on D_{train} and evaluated on D_{valid}.

The hyperparameter optimization (HPO) problem is to find a hyperparameter configuration $\boldsymbol{\lambda}^*$ that minimizes this loss:

$$\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} \mathcal{L}(A_{\boldsymbol{\lambda}}, D_{train}, D_{valid})$$

Types of Hyperparameters

- {SVM, RF, NN}
- Example 2: activation function ∈ {ReLU, Leaky ReLU, tanh}
- Example 3: operator ∈ {conv3x3, separable conv3x3, max pool, …}

– Special case: binary

Types of Hyperparameters

- Continuous

  Example: learning rate in NNs or GBMs

    - {SVM, RF, NN}
    - Example 2: activation function ∈ {ReLU, Leaky ReLU, tanh}
    - Example 3: operator ∈ {conv3x3, separable conv3x3, max pool, …}

  – Special case: binary

Types of Hyperparameters

- Continuous

  Example: learning rate in NNs or GBMs
- Integer

  Example: #units, #trees in GBM

    - {SVM, RF, NN}
    - Example 2: activation function ∈ {ReLU, Leaky ReLU, tanh}
    - Example 3: operator ∈ {conv3x3, separable conv3x3, max pool, …}

  – Special case: binary

Types of Hyperparameters

{conv3x3, separable conv3x3, max pool, ...}

{ReLU, Leaky ReLU, tanh}

{SVM, RF, NN}

- Continuous

  Example: learning rate in NNs or GBMs
- Integer

  Example: #units, #trees in GBM
- Categorical
  - Finite domain, unordered
  - Special case: binary
    - Example 2: activation function ∈ {ReLU, Leaky ReLU, tanh}
    - Example 3: operator ∈ {conv3x3, separable conv3x3, max pool, ...}
  - Special case: binary

Conditional hyperparameters

Conditional hyperparameters B are only active if other hyperparameters A are set a certain way

Conditional hyperparameters

Conditional hyperparameters B are only active if other hyperparameters A are set a certain way

- Example 1:
  - A = choice of optimizer (Adam or SGD)
  - B = Adam‘s second momentum hyperparameter (only active if A=Adam)

Conditional hyperparameters

Conditional hyperparameters B are only active if other hyperparameters A are set a certain way

- Example 1:
  - A = choice of optimizer (Adam or SGD)
  - B = Adam‘s second momentum hyperparameter (only active if A=Adam)
- Example 2:
  - A = number of layers in a deep neural network
  - B = number of units in layer k (only active if A >= k)

Conditional hyperparameters

Conditional hyperparameters B are only active if other hyperparameters A are set a certain way

- Example 1:
  - A = choice of optimizer (Adam or SGD)
  - B = Adam‘s second momentum hyperparameter (only active if A=Adam)
- Example 2:
  - A = number of layers in a deep neural network
  - B = number of units in layer k (only active if A >= k)
- Example 3:
  - A = choice of classifier (RF or SVM)
  - B = SVM‘s kernel hyperparameter (only active if A = SVM)

AutoML as Hyperparameter Optimization

Definition: Combined Algorithm Selection and Hyperparameter Optimization (CASH)

Let

- $\mathcal{A} = \{A^{(1)}, \dots, A^{(n)}\}$ be a set of algorithms
- $\mathbf{\Lambda}^{(i)}$ denote the hyperparameter space of $A^{(i)}$, for $i = 1, \dots, n$
- $\mathcal{L}(A^{(i)}_{\boldsymbol{\lambda}}, D_{train}, D_{valid})$ denote the loss of $A^{(i)}$, using $\lambda \in \mathbf{\Lambda}^{(i)}$ trained on D_{train} and evaluated on D_{valid}.

The Combined Algorithm Selection and Hyperparameter Optimization (CASH) problem is to find a combination of algorithm $A^* = A^{(i)}$ and hyperparameter configuration $\boldsymbol{\lambda}^* \in \mathbf{\Lambda}^{(i)}$ that minimizes this loss:

$$A^*_{\boldsymbol{\lambda}^*} \in \operatorname*{arg\,min}_{A^{(i)} \in \mathcal{A}, \boldsymbol{\lambda} \in \mathbf{\Lambda}^{(i)}} \mathcal{L}(A^{(i)}_{\boldsymbol{\lambda}}, D_{train}, D_{valid})$$

AutoML as Hyperparameter Optimization

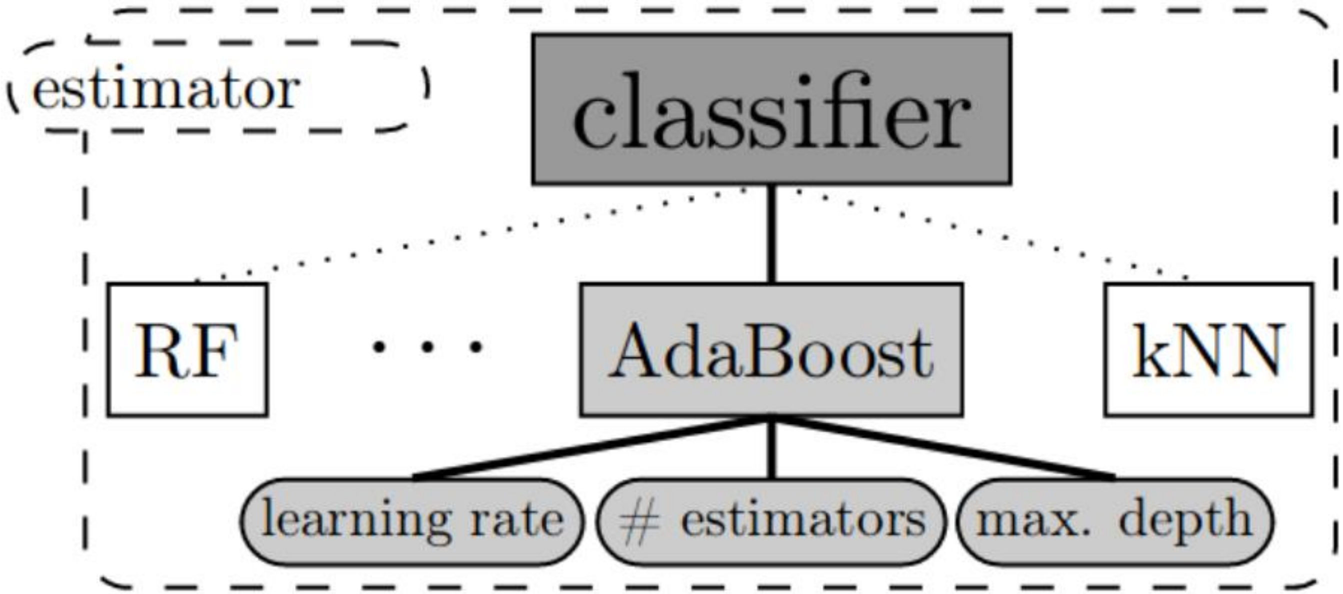
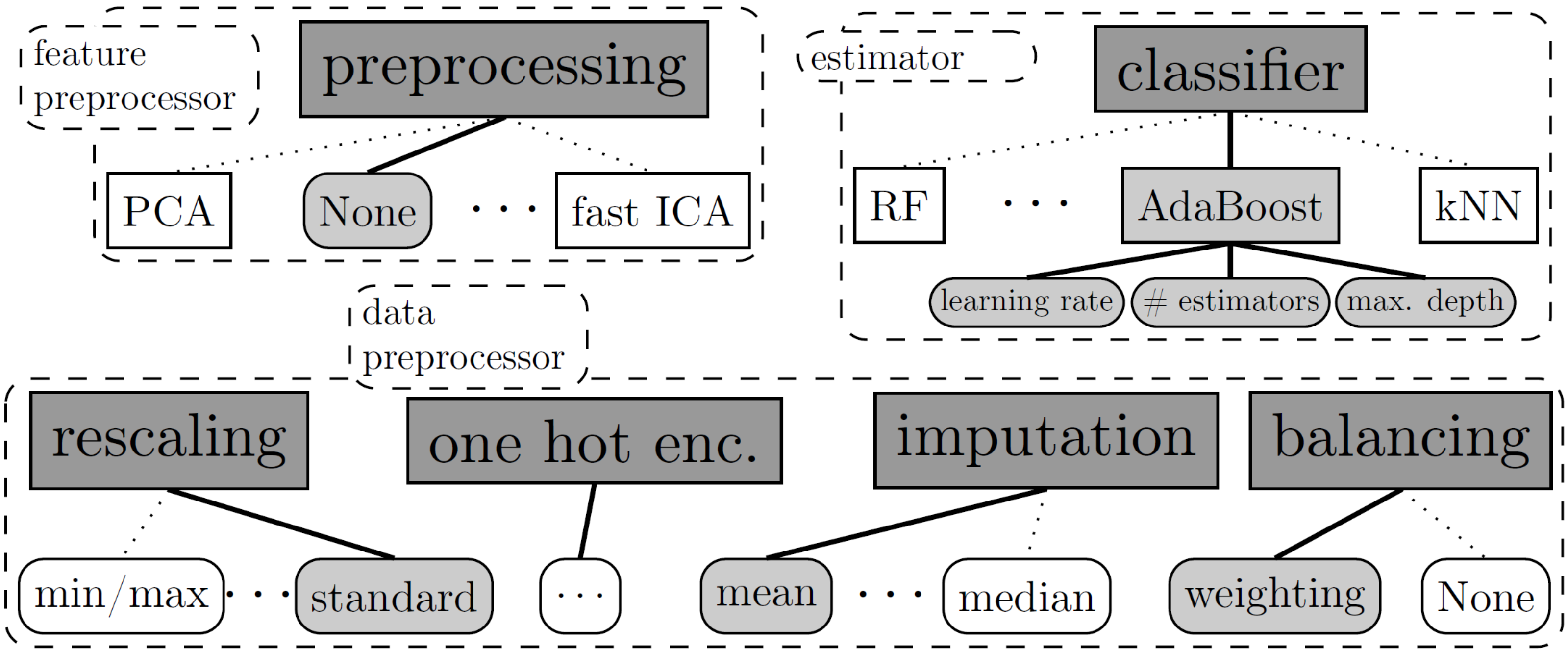


Illustration of the CASH problem in Auto-sklearn:

- 15 base classifiers
- Up to ten hyperparameters each
- Four levels of conditionality

AutoML as Hyperparameter optimization

Not limited to the classification algorithm:



See also Thornton et al. (KDD 2013) which introduced the CASH problem.

Outline

Part 1: General AutoML

1. AutoML by Hyperparameter Optimization
2. Black-box Hyperparameter Optimization
3. Beyond black-box optimization
4. Meta-learning
5. Examples of AutoML
6. Open issues and future work
7. Wrap-up & Conclusion

Part 2: Neural Architecture Search & Meta-Learning

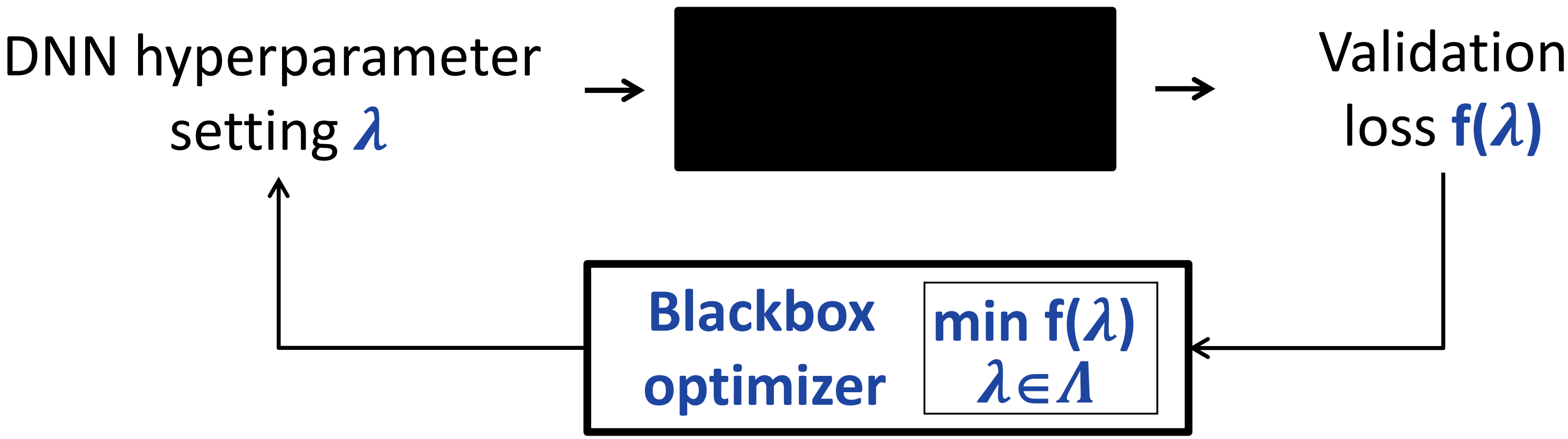
Blackbox Hyperparameter Optimization

DNN hyperparameter setting λ	$\rightarrow$	Train DNN and validate it	$\rightarrow$	Validation loss $\mathbf{f}(\lambda)$

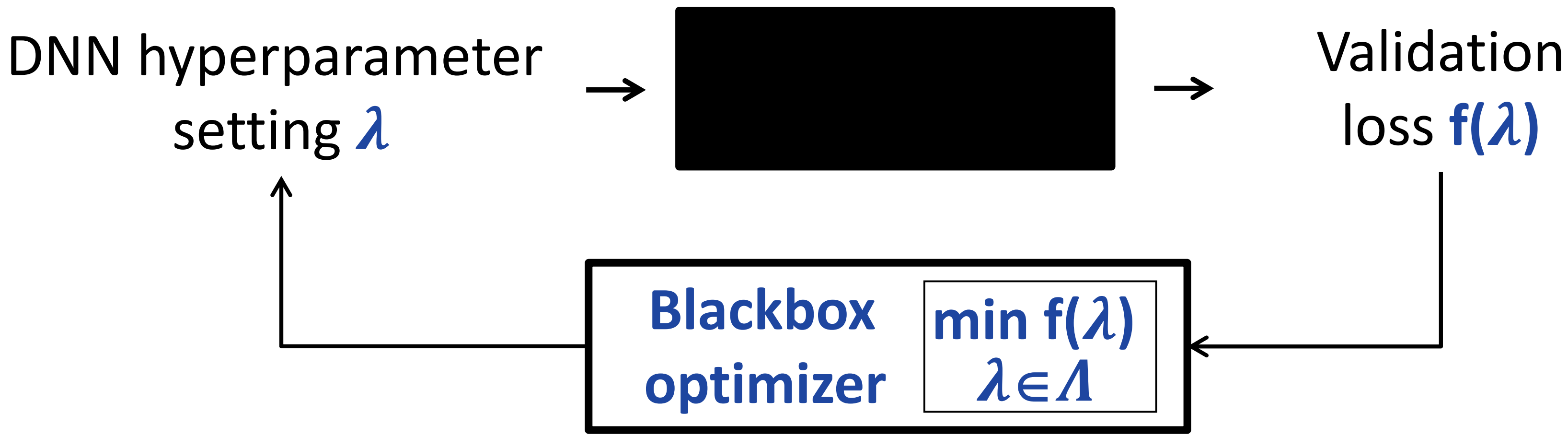
Blackbox Hyperparameter Optimization

DNN hyperparameter setting λ → ■ → Validation loss $\mathbf{f}(\lambda)$

Blackbox Hyperparameter Optimization



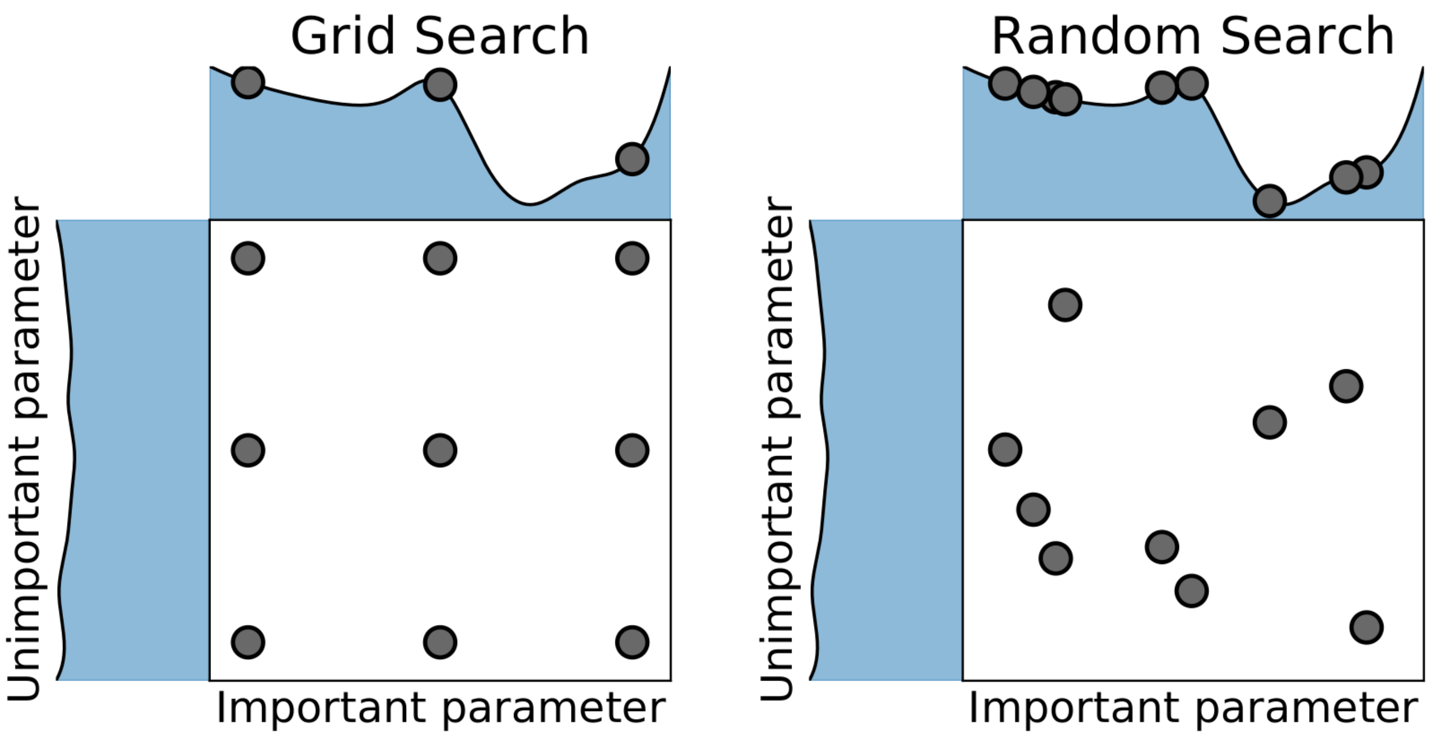
Blackbox Hyperparameter Optimization



- The blackbox function is expensive to evaluate
  $\rightarrow$ sample efficiency is important

Grid Search and Random Search

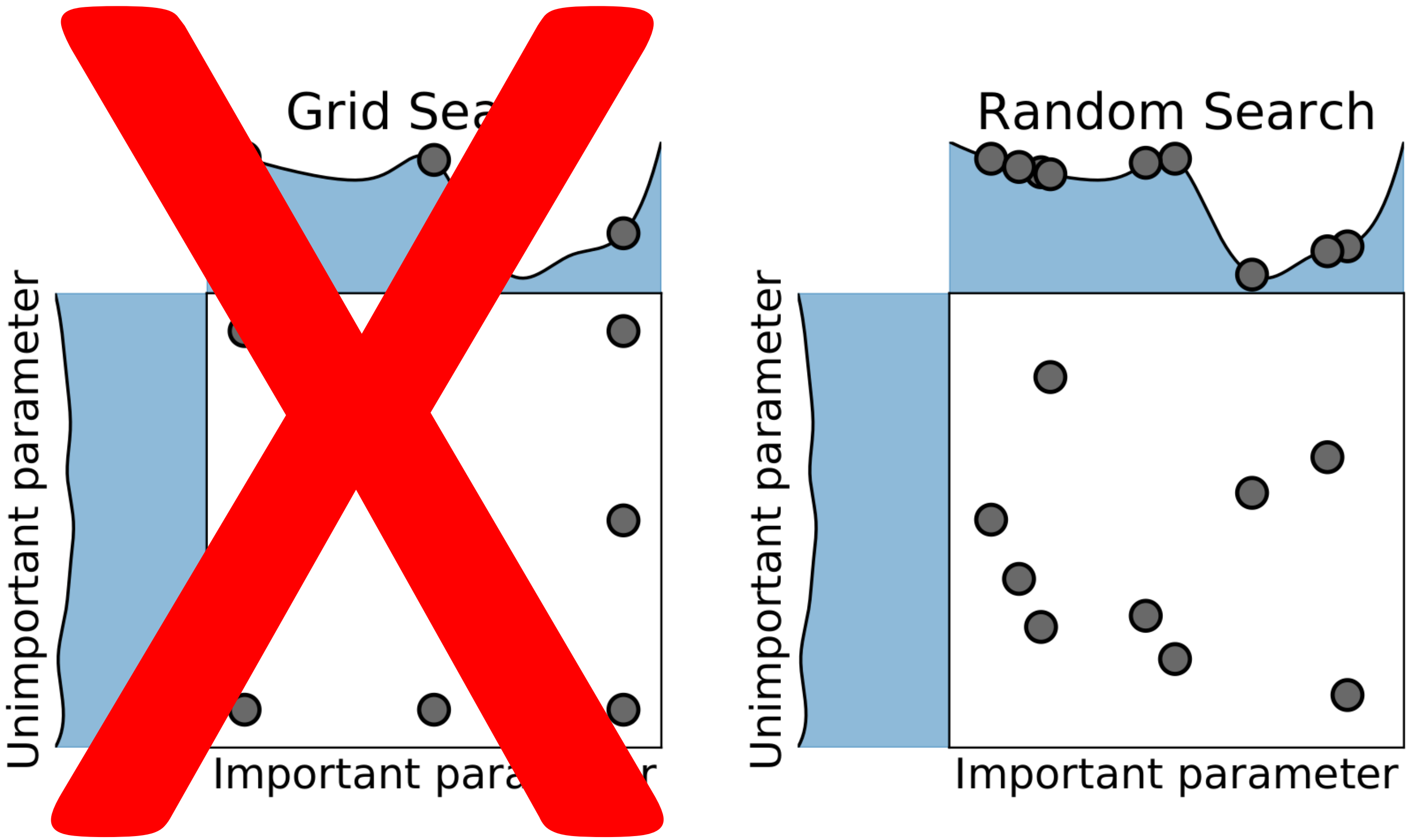
- Both completely uninformed
- Grid search suffers from the curse of dimensionality
- Random search handles low intrinsic dimensionality better
- Example: an additive function ($y = f(x) + g(x)$)



Bergstra and Bengio, JMLR 2012; Image source: Feurer & Hutter, CC-BY 4.0

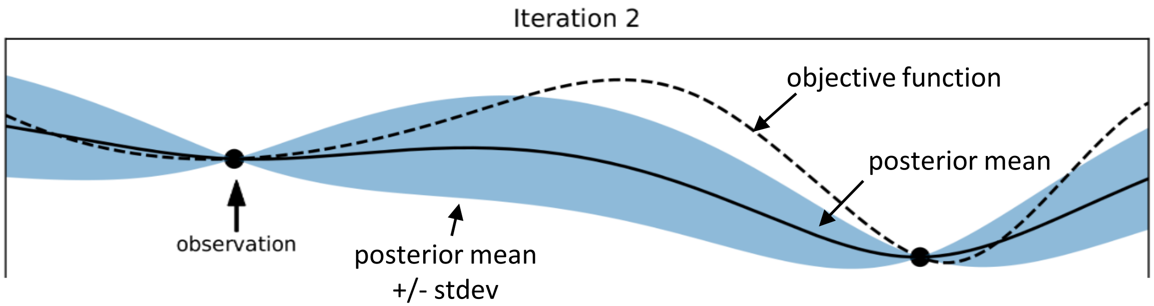
Grid Search and Random Search

- Both completely uninformed
- Grid search suffers from the curse of dimensionality
- Random search handles low intrinsic dimensionality better
- Example: an additive function ($y = f(x) + g(x)$)

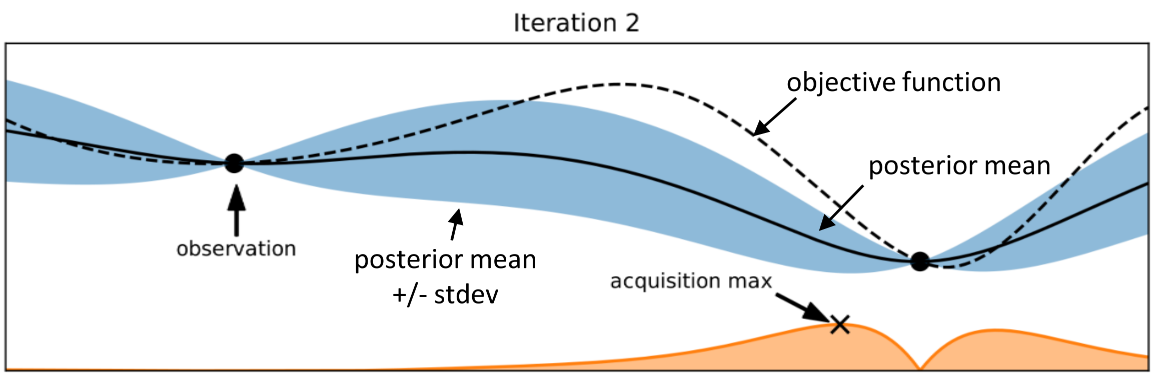


Bergstra and Bengio, JMLR 2012; Image source: Feurer & Hutter, CC-BY 4.0

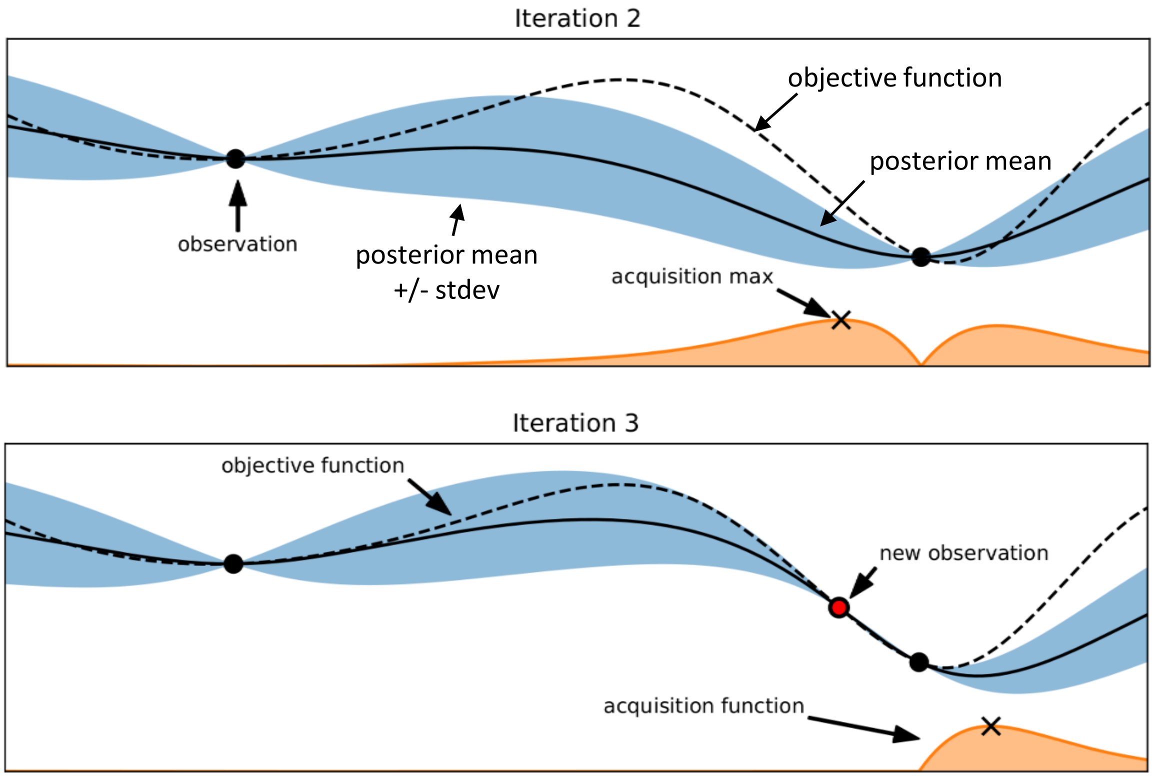
Bayesian Optimization



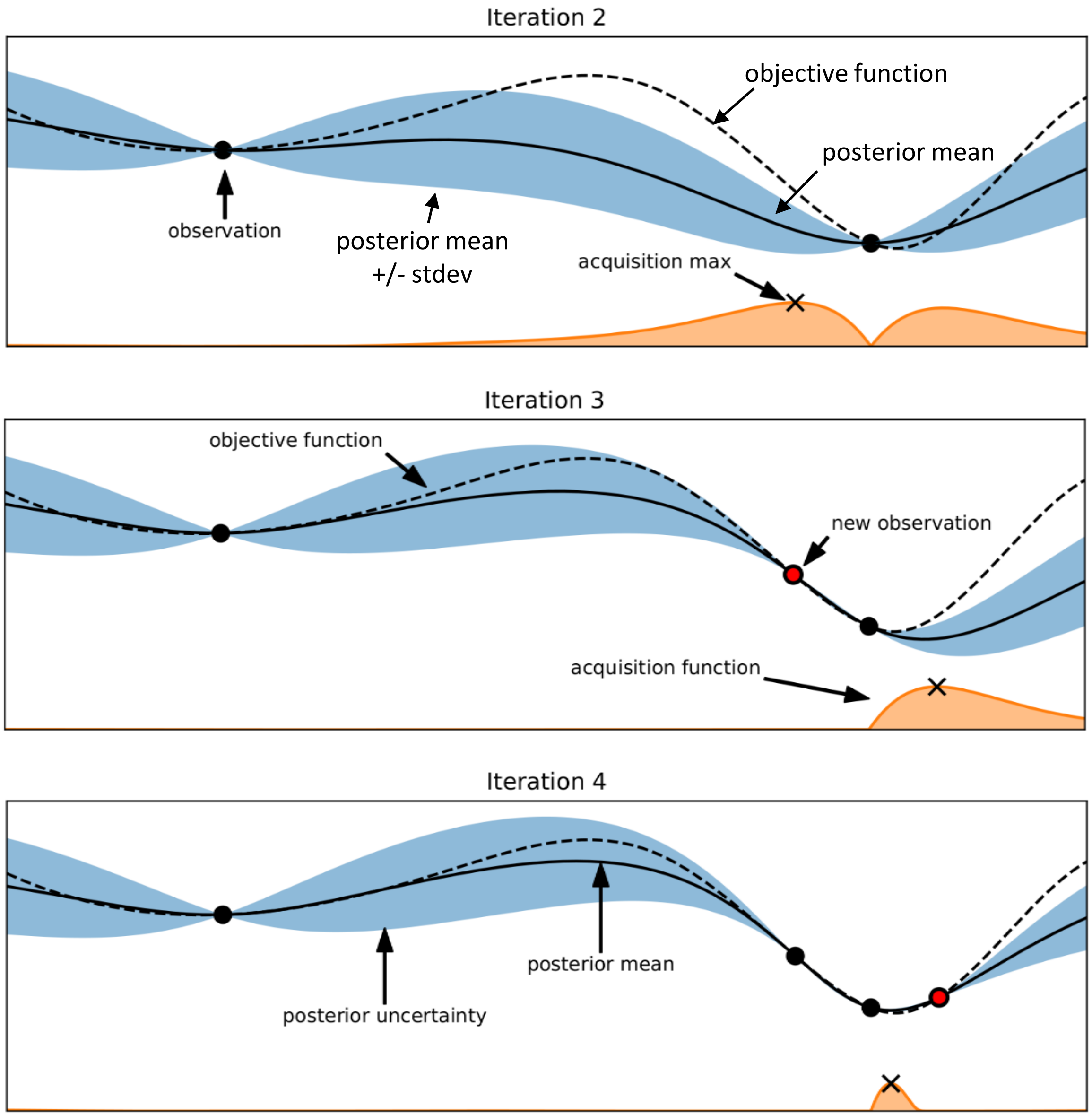
Bayesian Optimization



Bayesian Optimization



Bayesian Optimization



Acquisition Function: Expected Improvement

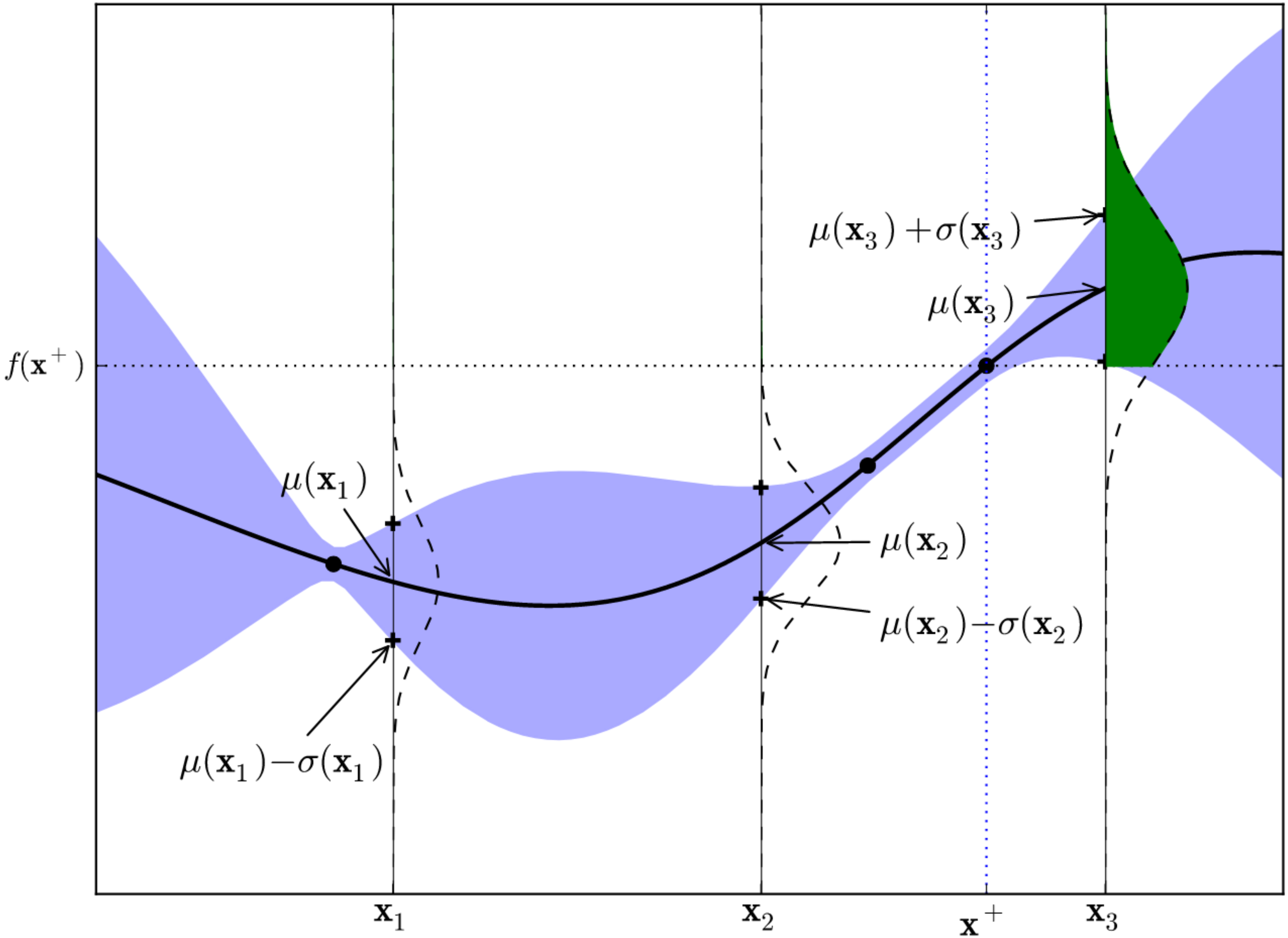


Image source: Brochu et al., arXiv:1012.2599

Bayesian Optimization

Approach

- Conduct an initial design
- Iteratively:
  - Fit a proabilistic model to the function evaluations $\langle \lambda, f(\lambda) \rangle$, most often a Gaussian process
  - Use that model to trade off Exploration vs. Exploitation in an acquisition function

Popular since Mockus [1974]

- Sample-efficient
- Works when objective is nonconvex, noisy, has unknown derivatives, etc
- Recent convergence results [Srinivas et al, 2010; Bull 2011; de Freitas et al, 2012; Kawaguchi et al, 2016; Nguyen et al., 2017; Berkenkamp et al., 2019]
- Excellent reviews by Shahriari et al. (IEEE, 2016) and Frazier (arXiv:1807.02811)

Example: Bayesian Optimization in AlphaGo

- During the development of AlphaGo, its many hyperparameters were tuned with Bayesian optimization multiple times.

- This automatic tuning process resulted in substantial improvements in playing strength. For example, prior to the match with Lee Sedol, we tuned the latest AlphaGo agent and this improved its win-rate from 50% to 66.5% in self-play games. This tuned version was deployed in the final match.

- Of course, since we tuned AlphaGo many times during its development cycle, the compounded contribution was even higher than this percentage.

[Chen et al., arXiv:1812.06855]

AutoML Challenges for Bayesian Optimization

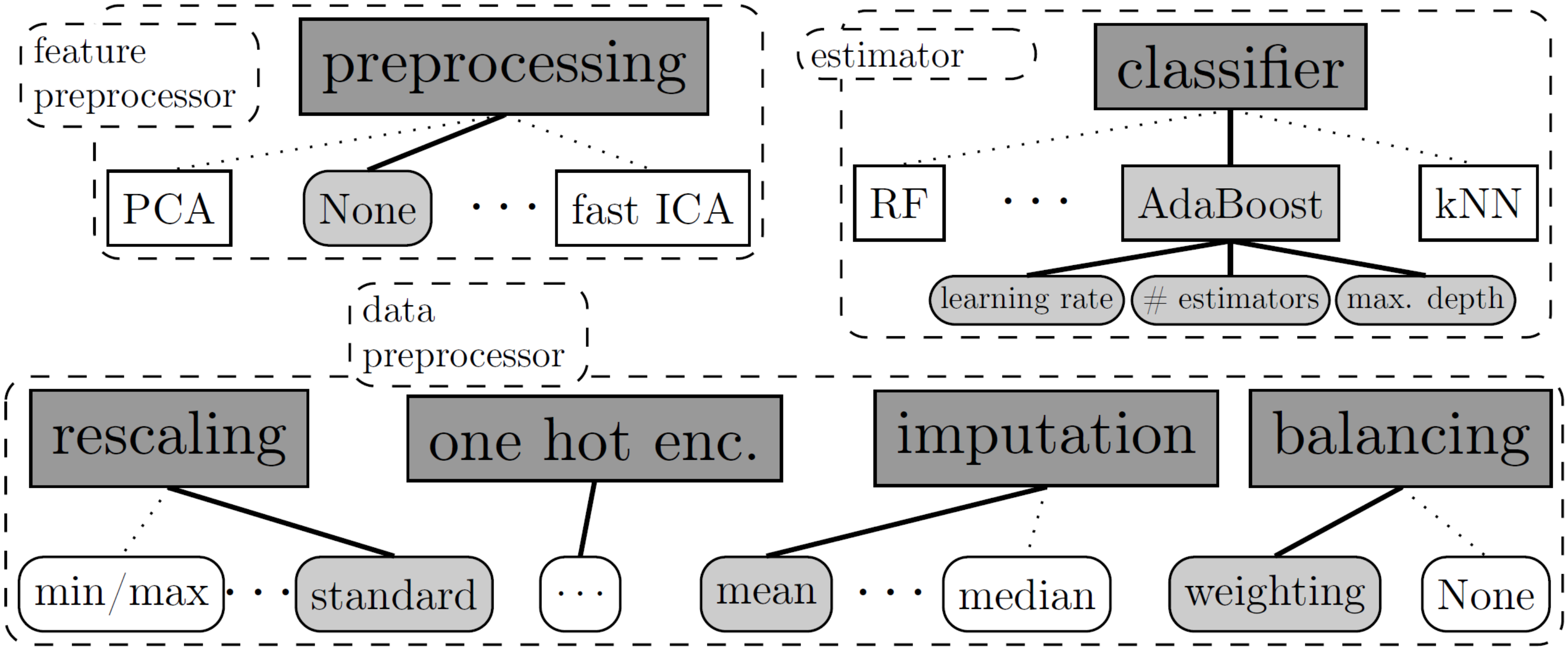
- Problems for standard Gaussian Process (GP) approach:
  - Complex hyperparameter space
    - High-dimensional (low effective dimensionality) [e.g., Wang et al., 2013]
    - Mixed continuous/discrete hyperparameters [e.g., Hutter et al., 2011]
    - Conditional hyperparameters [e.g., Jenatton et al., 2017]
  - Noise: sometimes heteroscedastic, large, non-Gaussian
  - Model overhead (budget is runtime, not #function evaluations)

AutoML Challenges for Bayesian Optimization

- Problems for standard Gaussian Process (GP) approach:
  - Complex hyperparameter space
    - High-dimensional (low effective dimensionality) [e.g., Wang et al., 2013]
    - Mixed continuous/discrete hyperparameters [e.g., Hutter et al., 2011]
    - Conditional hyperparameters [e.g., Jenatton et al., 2017]
  - Noise: sometimes heteroscedastic, large, non-Gaussian
  - Model overhead (budget is runtime, not #function evaluations)

- Simple solution used in SMAC: random forests [Breiman, 2001]
  - Frequentist uncertainty estimate:
    variance across individual trees' predictions [Hutter et al, 2011]

AutoML Challenges for Bayesian Optimization



- Simple solution used in SMAC: random forests [Breiman, 2001]
  - Frequentist uncertainty estimate: variance across individual trees' predictions [Hutter et al, 2011]

Other methods

– is good) and p(λ is bad), rather than p(y|λ)

Other methods

- Two recent promising models for Bayesian optimization
  - Neural networks with Bayesian linear regression using the features in the output layer [Snoek et al, ICML 2015]
  - Fully Bayesian neural networks, trained with stochastic gradient Hamiltonian Monte Carlo [Springenberg et al, NIPS 2016]

  - is good) and p(λ is bad), rather than p(y|λ)

Other methods

- $\lambda\lambda$ is bad), rather than p(y|λ)
- Two recent promising models for Bayesian optimization
  - Neural networks with Bayesian linear regression using the features in the output layer [Snoek et al, ICML 2015]
  - Fully Bayesian neural networks, trained with stochastic gradient Hamiltonian Monte Carlo [Springenberg et al, NIPS 2016]
- Tree Parzen Estimator [Bergstra et al., 2011]
  - Ratio is proportional to Expected Improvement

Other methods

- $\lambda\lambda$ is bad), rather than p(y|λ)
- Two recent promising models for Bayesian optimization
  - Neural networks with Bayesian linear regression using the features in the output layer [Snoek et al, ICML 2015]
  - Fully Bayesian neural networks, trained with stochastic gradient Hamiltonian Monte Carlo [Springenberg et al, NIPS 2016]
- Tree Parzen Estimator [Bergstra et al., 2011]
  - Ratio is proportional to Expected Improvement
- Population-based methods
  - Genetic algorithms, evolutionary algorithms, evolutionary strategies, particle swarm optimization
  - Embarassingly parallel, conceptually simple

Other methods

- $\lambda\lambda$ is bad), rather than p(y|λ)
- Two recent promising models for Bayesian optimization
  - Neural networks with Bayesian linear regression using the features in the output layer [Snoek et al, ICML 2015]
  - Fully Bayesian neural networks, trained with stochastic gradient Hamiltonian Monte Carlo [Springenberg et al, NIPS 2016]
- Tree Parzen Estimator [Bergstra et al., 2011]
  - Ratio is proportional to Expected Improvement
- Population-based methods
  - Genetic algorithms, evolutionary algorithms, evolutionary strategies, particle swarm optimization
  - Embarassingly parallel, conceptually simple
- See Chapter 1 of the AutoML book for more information.

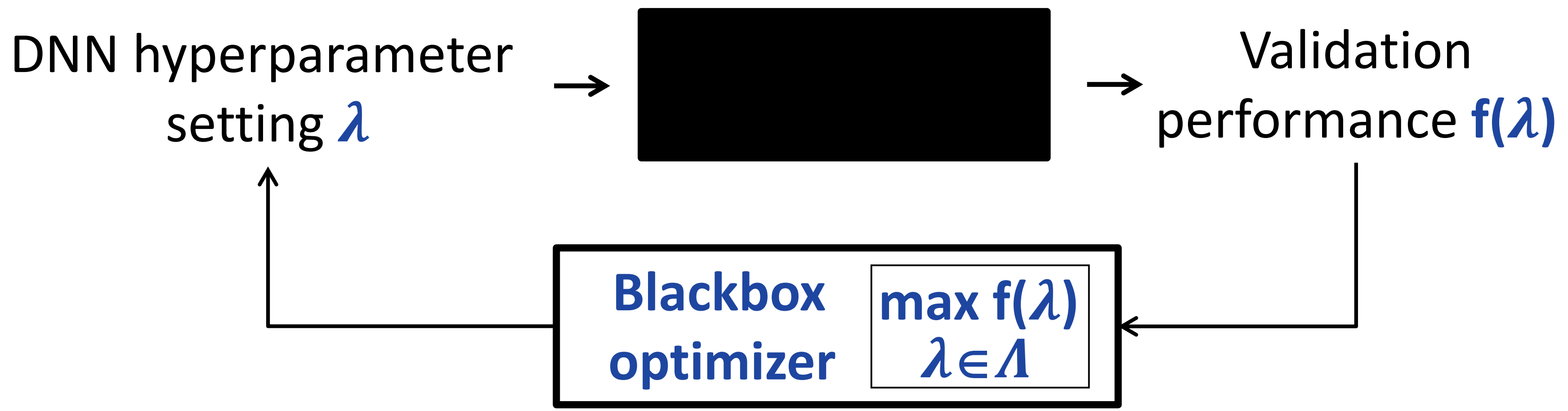
Outline

Part 1: General AutoML

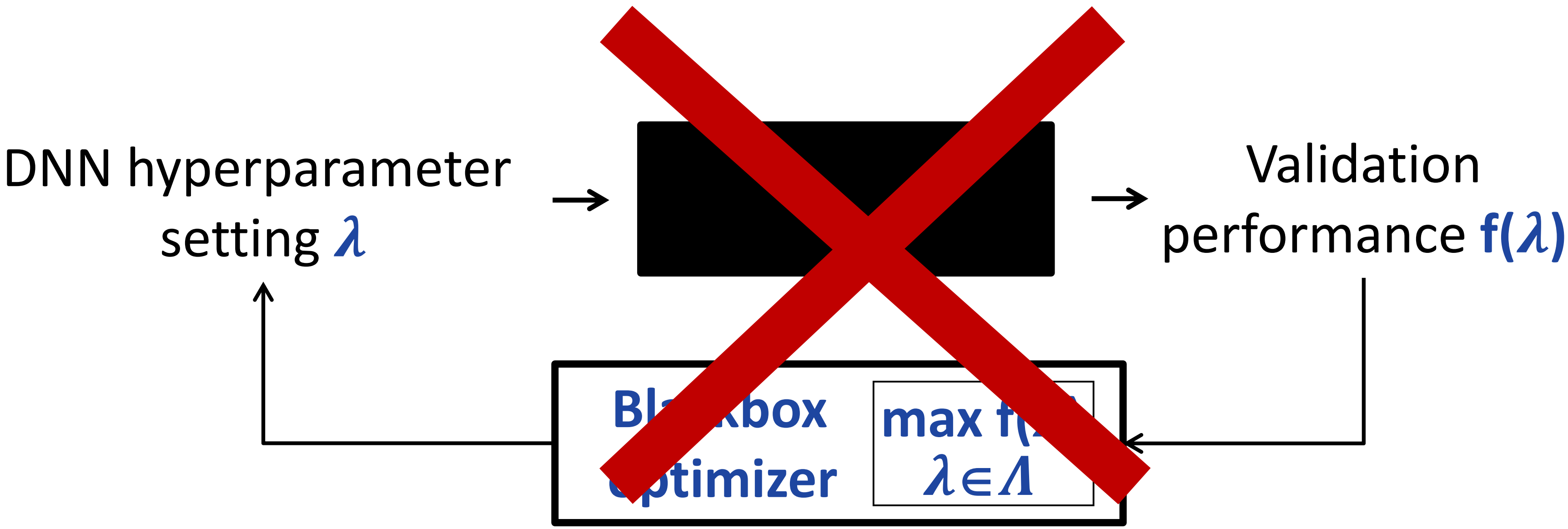
1. AutoML by Hyperparameter Optimization
2. Black-box Hyperparameter Optimization
3. Beyond black-box optimization
4. Meta-learning
5. Examples of AutoML
6. Open issues and future work
7. Wrap-up & Conclusion

Part 2: Neural Architecture Search & Meta-Learning

Beyond Blackbox Hyperparameter Optimization



Beyond Blackbox Hyperparameter Optimization

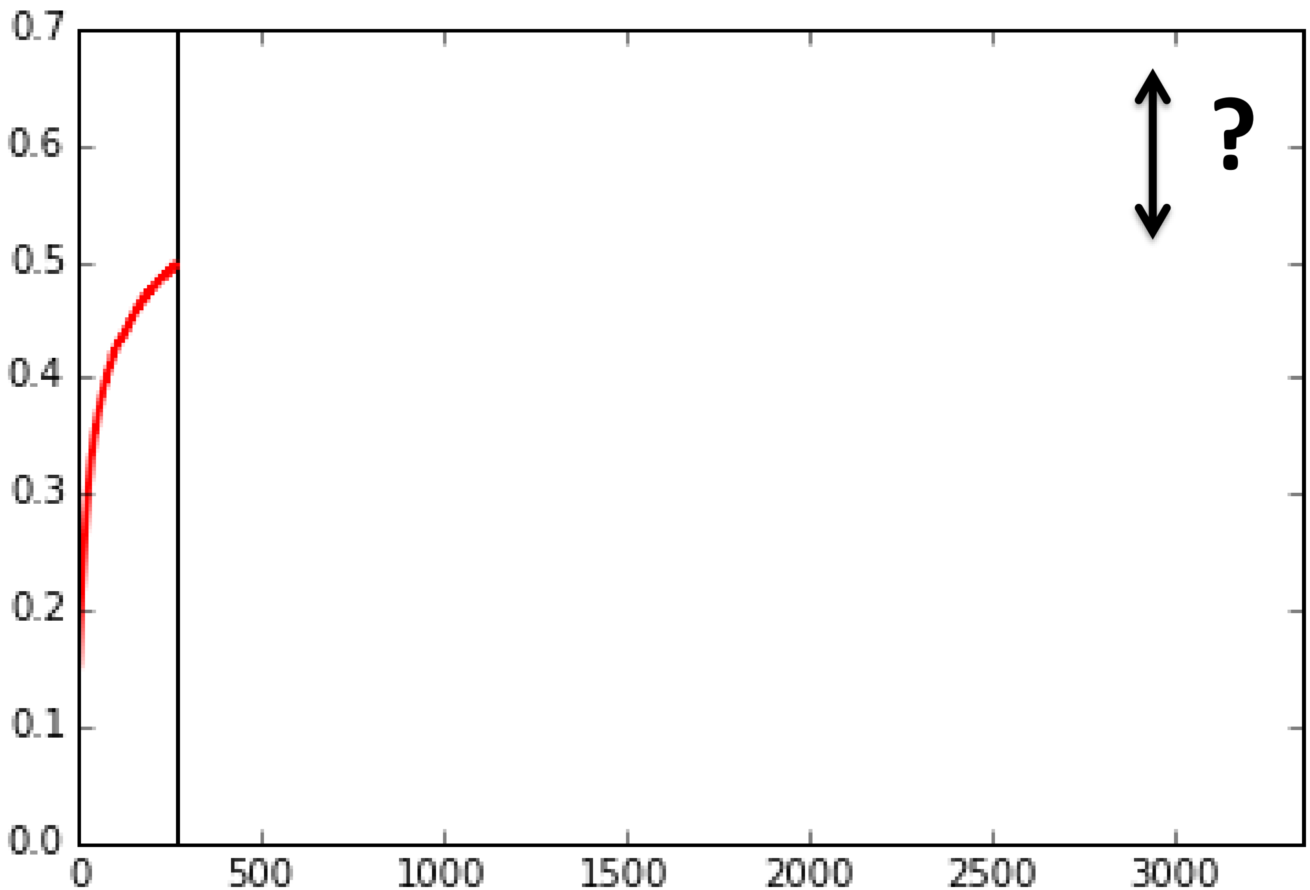


Too slow for DL / big data

Main Approaches Going Beyond Blackbox HPO

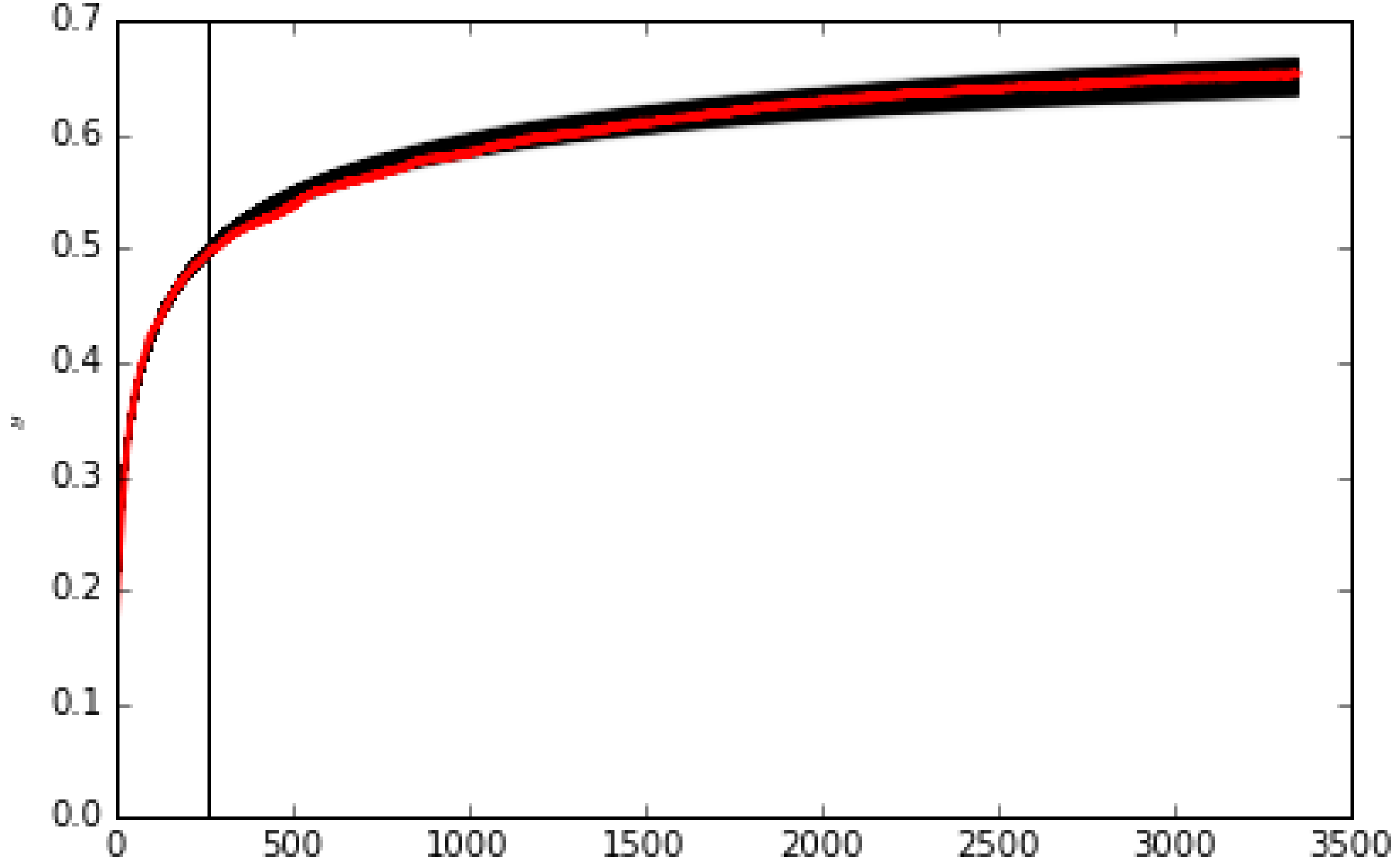
- Extrapolation of learning curves
- Multi-fidelity optimization
- Meta-learning [next part]
- Hyperparameter gradient descent [see AutoML book]

Probabilistic Extrapolation of Learning Curves



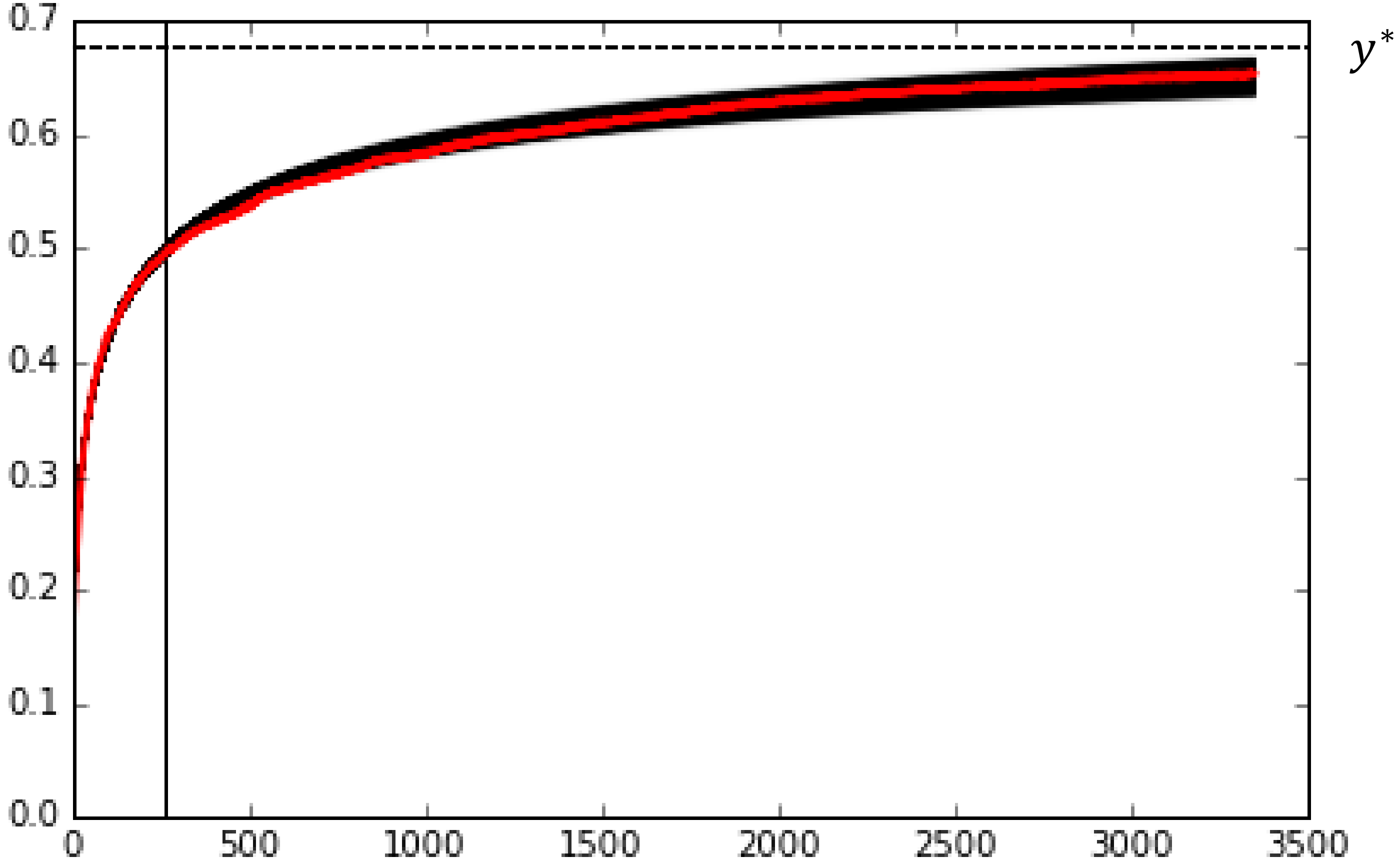
- Parametric learning curve models [Domhan et al, IJCAI 2015]
- Bayesian neural networks [Klein et al, ICLR 2017]
- Linear combination of previous curves [Chandrashekaran and Lane, ECML2017]

Probabilistic Extrapolation of Learning Curves



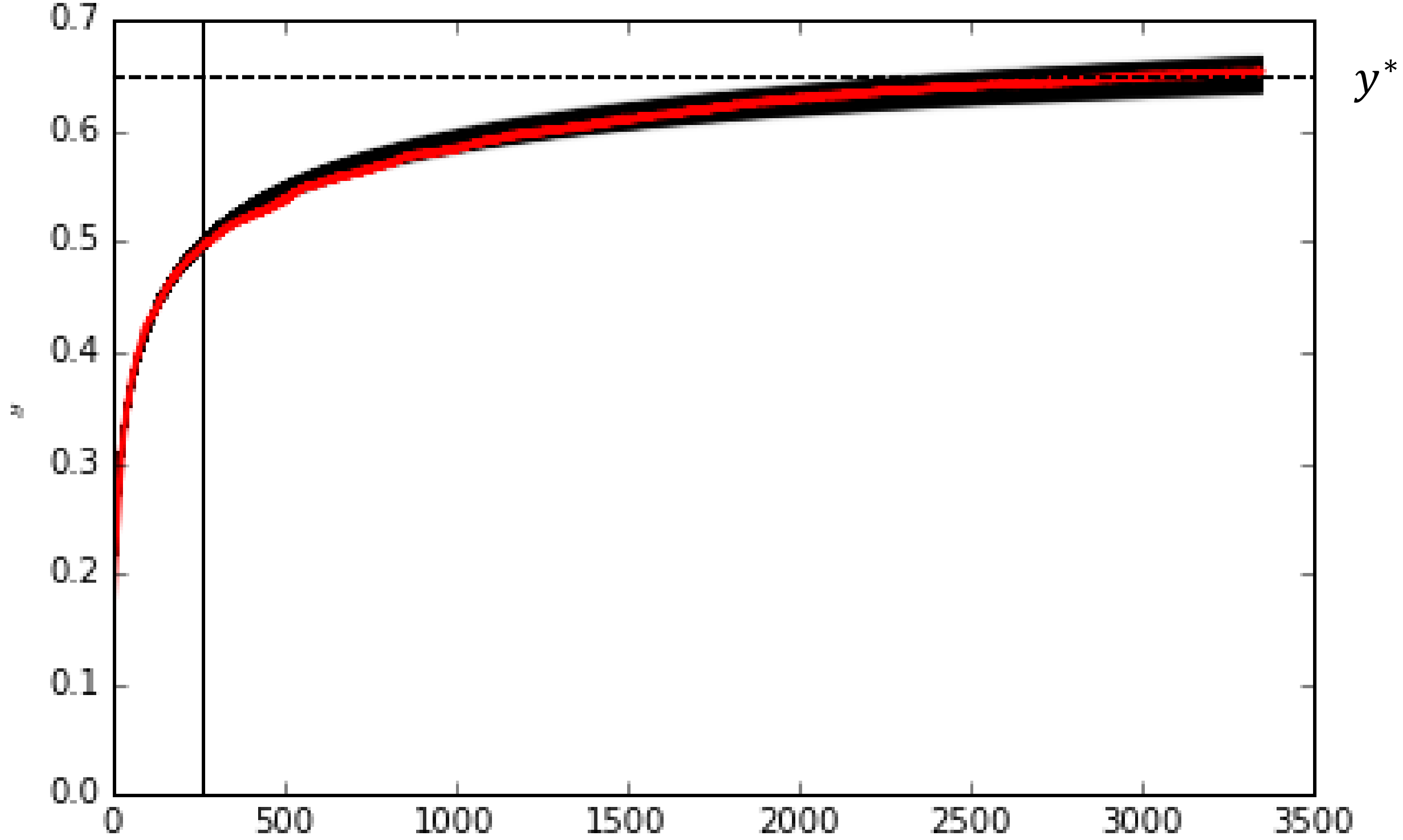
- Parametric learning curve models [Domhan et al, IJCAI 2015]
- Bayesian neural networks [Klein et al, ICLR 2017]
- Linear combination of previous curves [Chandrashekaran and Lane, ECML2017]

Probabilistic Extrapolation of Learning Curves



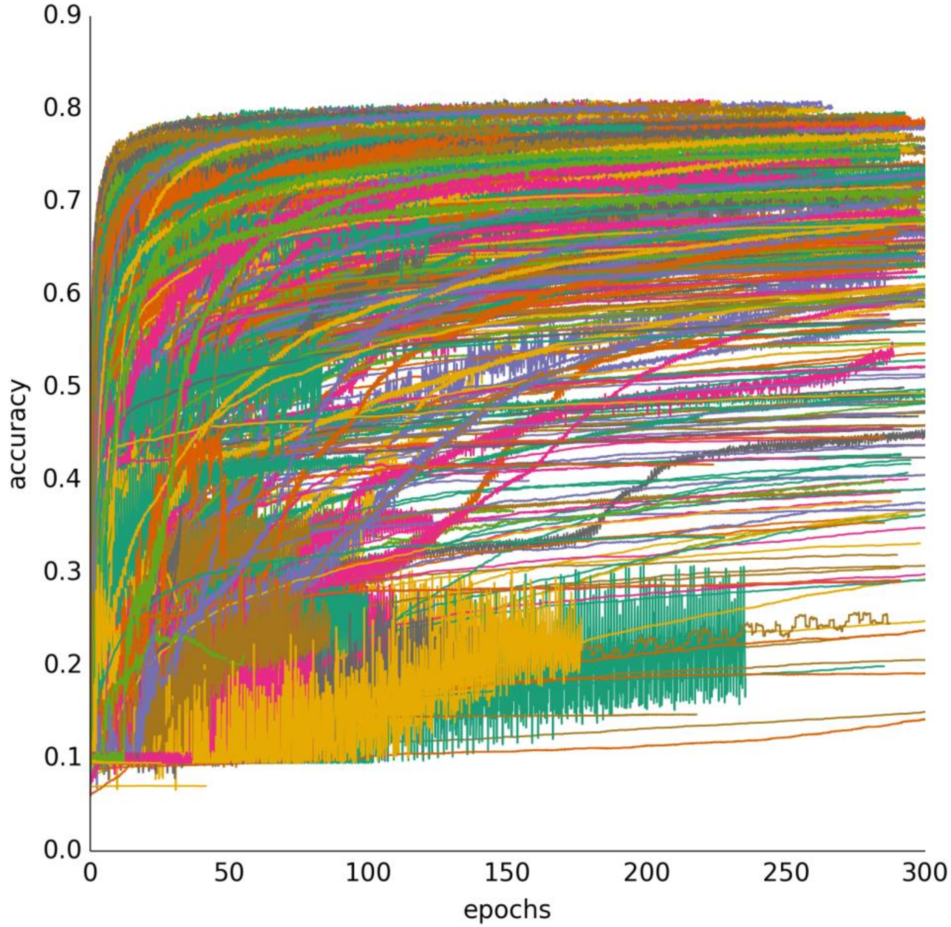
- Parametric learning curve models [Domhan et al, IJCAI 2015]
- Bayesian neural networks [Klein et al, ICLR 2017]
- Linear combination of previous curves [Chandrashekaran and Lane, ECML2017]

Probabilistic Extrapolation of Learning Curves



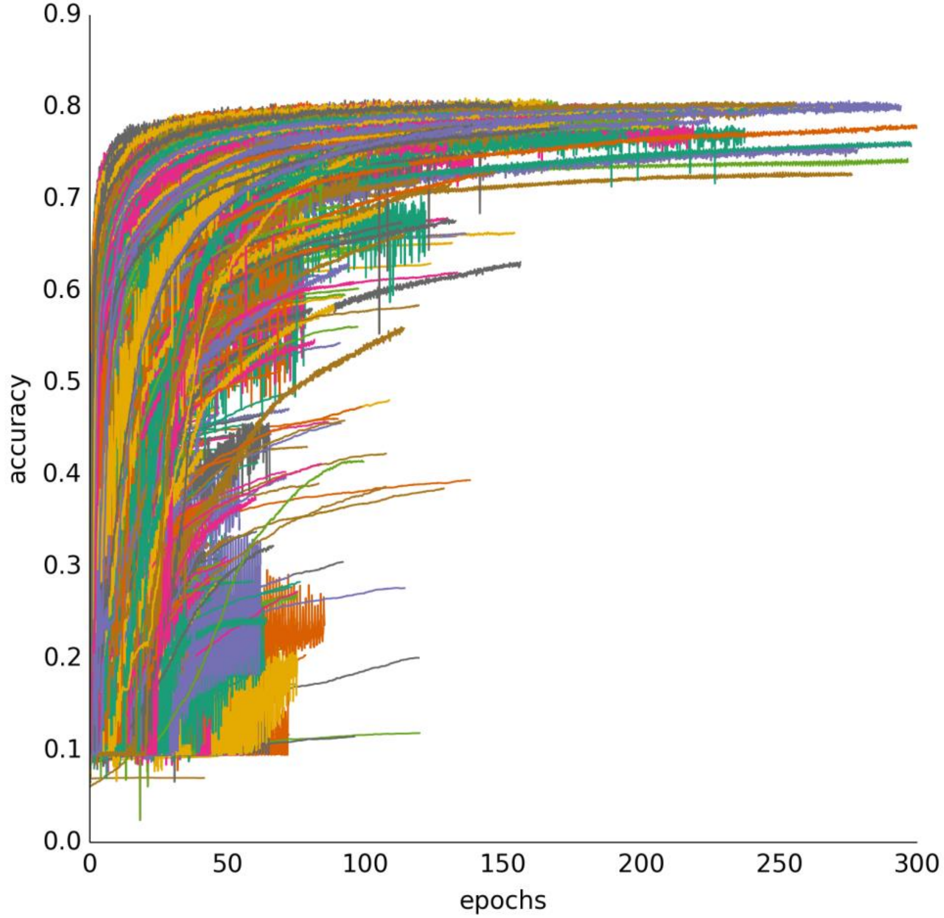
- Parametric learning curve models [Domhan et al, IJCAI 2015]
- Bayesian neural networks [Klein et al, ICLR 2017]
- Linear combination of previous curves [Chandrashekaran and Lane, ECML2017]

Probabilistic Extrapolation of Learning Curves



- Parametric learning curve models [Domhan et al, IJCAI 2015]
- Bayesian neural networks [Klein et al, ICLR 2017]
- Linear combination of previous curves [Chandrashekaran and Lane, ECML2017]

Probabilistic Extrapolation of Learning Curves



- Parametric learning curve models [Domhan et al, IJCAI 2015]
- Bayesian neural networks [Klein et al, ICLR 2017]
- Linear combination of previous curves [Chandrashekaran and Lane, ECML2017]

Multi-Fidelity Optimization

- Use cheap approximations of the blackbox, performance on which correlates with the blackbox, e.g.
  - Subsets of the data
  - Fewer epochs of iterative training algorithms (e.g., SGD)
  - Fewer trials in deep reinforcement learning
  - Downsampled images in object recognition

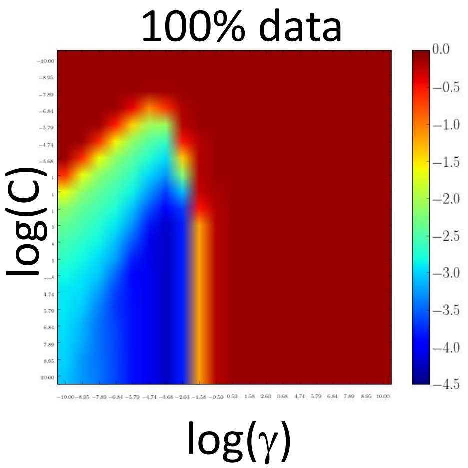
Multi-Fidelity Optimization

- Use cheap approximations of the blackbox, performance on which correlates with the blackbox, e.g.
  - Subsets of the data
  - Fewer epochs of iterative training algorithms (e.g., SGD)
  - Fewer trials in deep reinforcement learning
  - Downsampled images in object recognition

  - Also applicable in different domains, e.g., fluid simulations:
    - Less particles
    - Shorter simulations

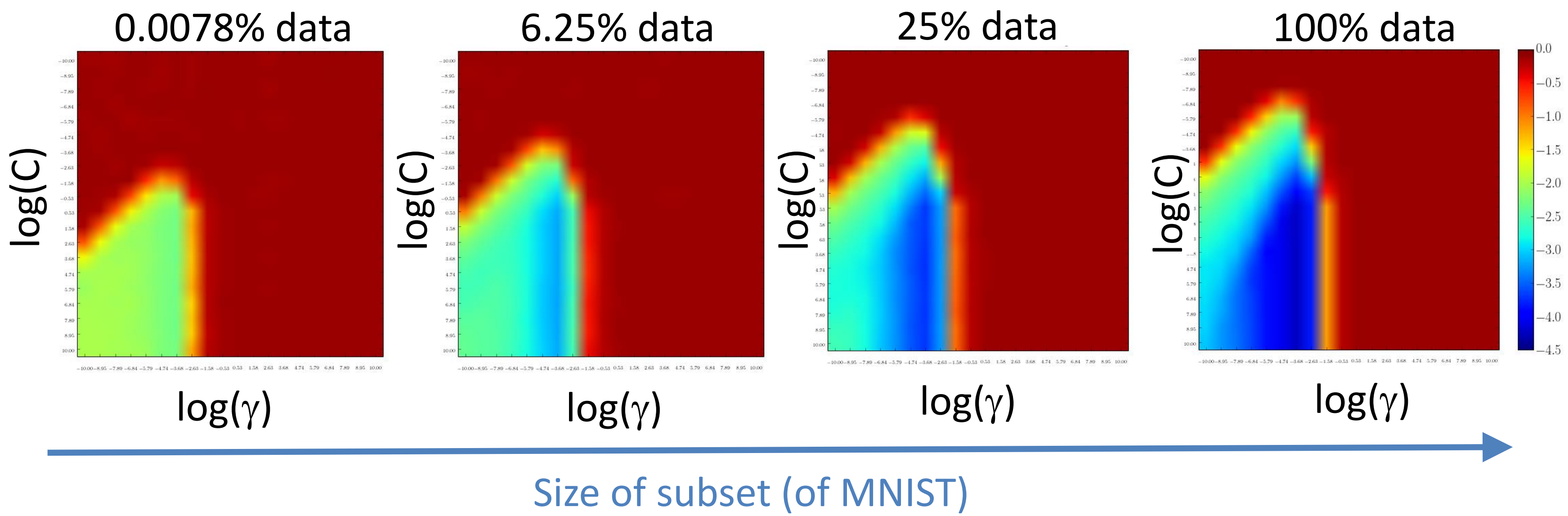
Multi-fidelity Optimization

- **Make use of cheap low-fidelity evaluations**
  - E.g.: subsets of the data (here: SVM on MNIST)



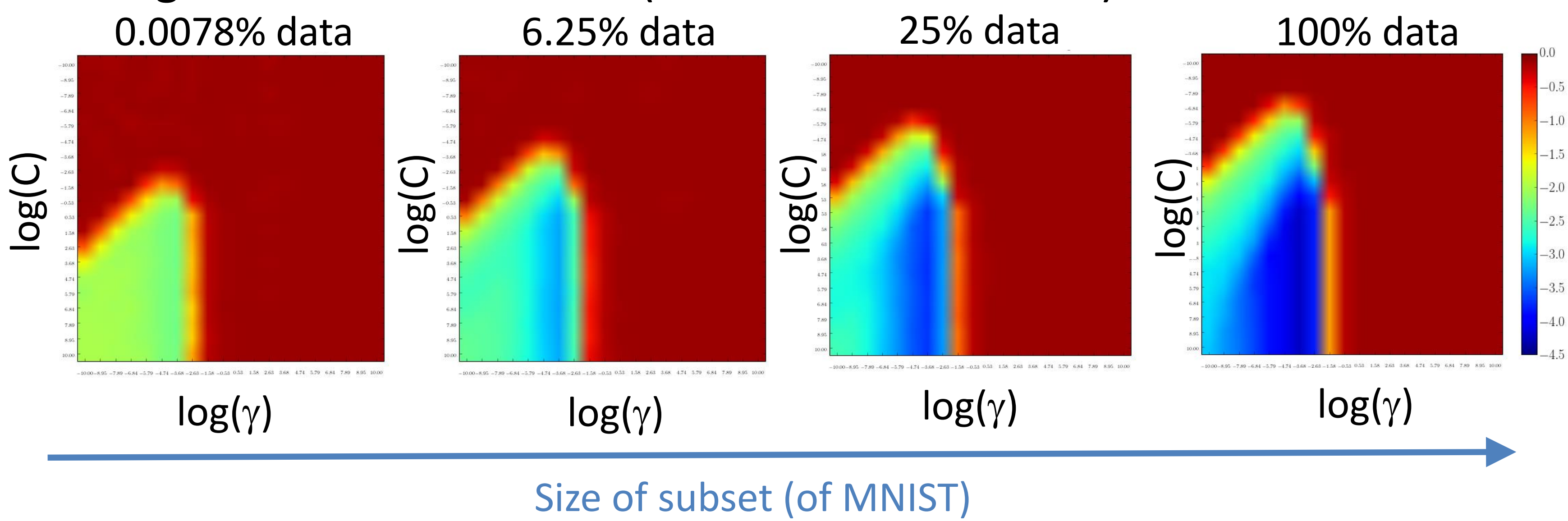
Multi-fidelity Optimization

- **Make use of cheap low-fidelity evaluations**
  - E.g.: subsets of the data (here: SVM on MNIST)



Multi-fidelity Optimization

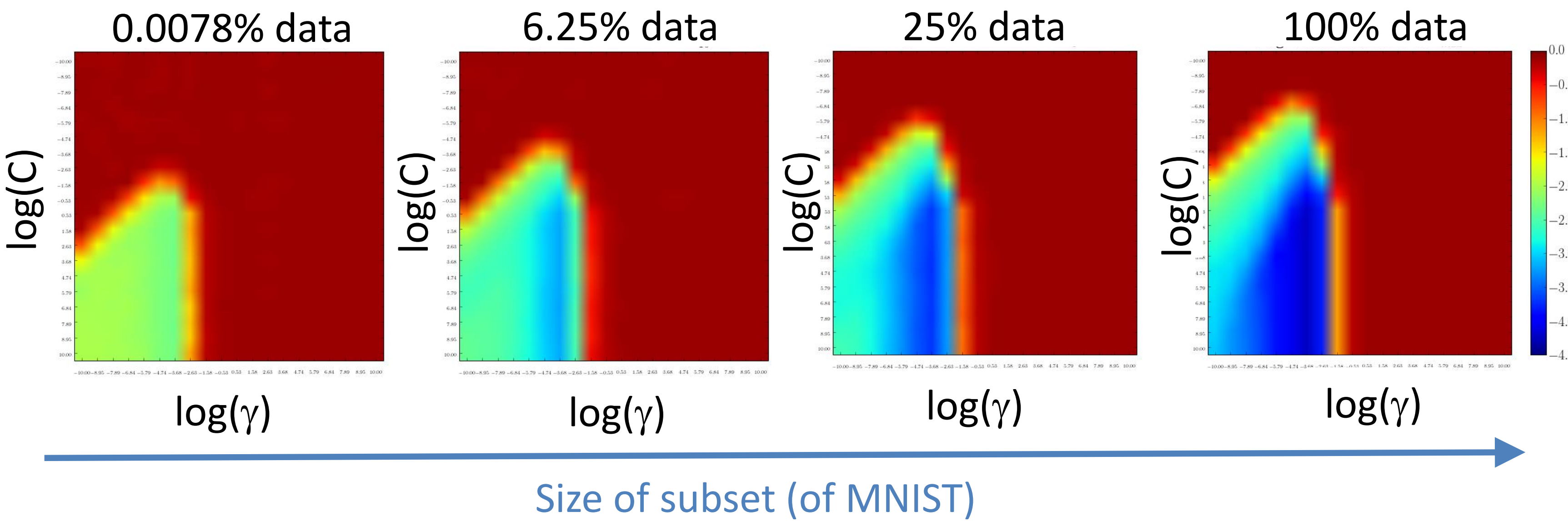
- **Make use of cheap low-fidelity evaluations**
  - E.g.: subsets of the data (here: SVM on MNIST)



  - Many cheap evaluations on small subsets
  - Few expensive evaluations on the full data
  - **Up to 1000x speedups** [Klein et al, AISTATS 2017]

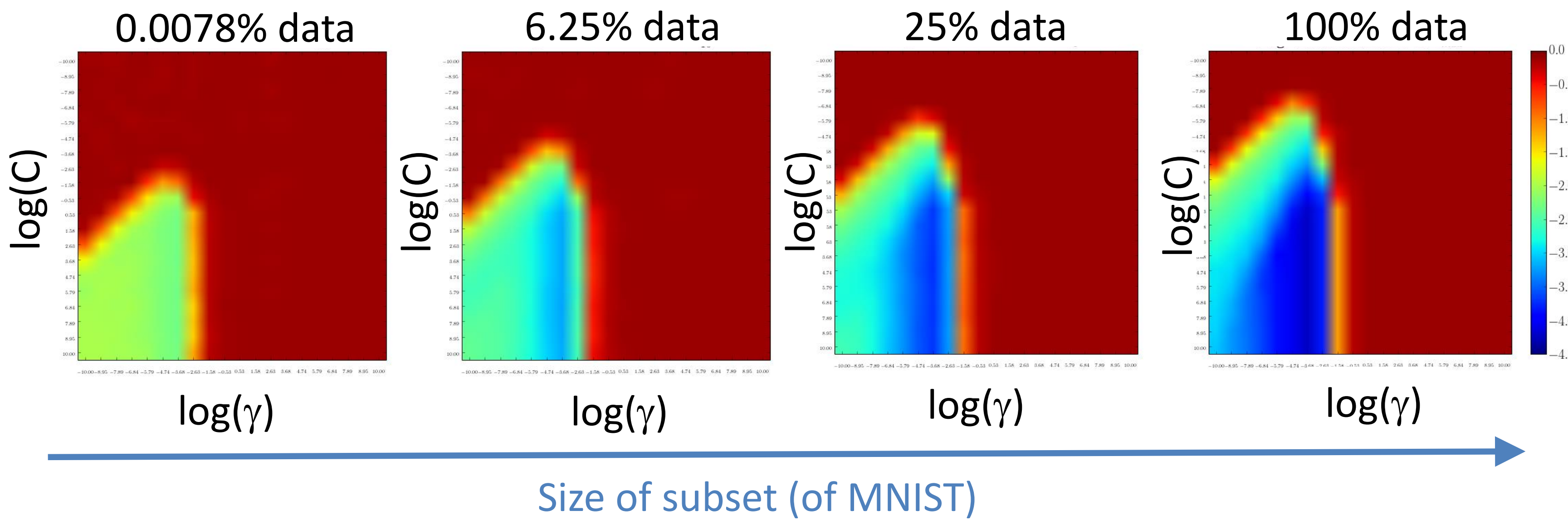
Multi-fidelity Optimization

- **Make use of cheap low-fidelity evaluations**
  - E.g.: subsets of the data (here: SVM on MNIST)



Multi-fidelity Optimization

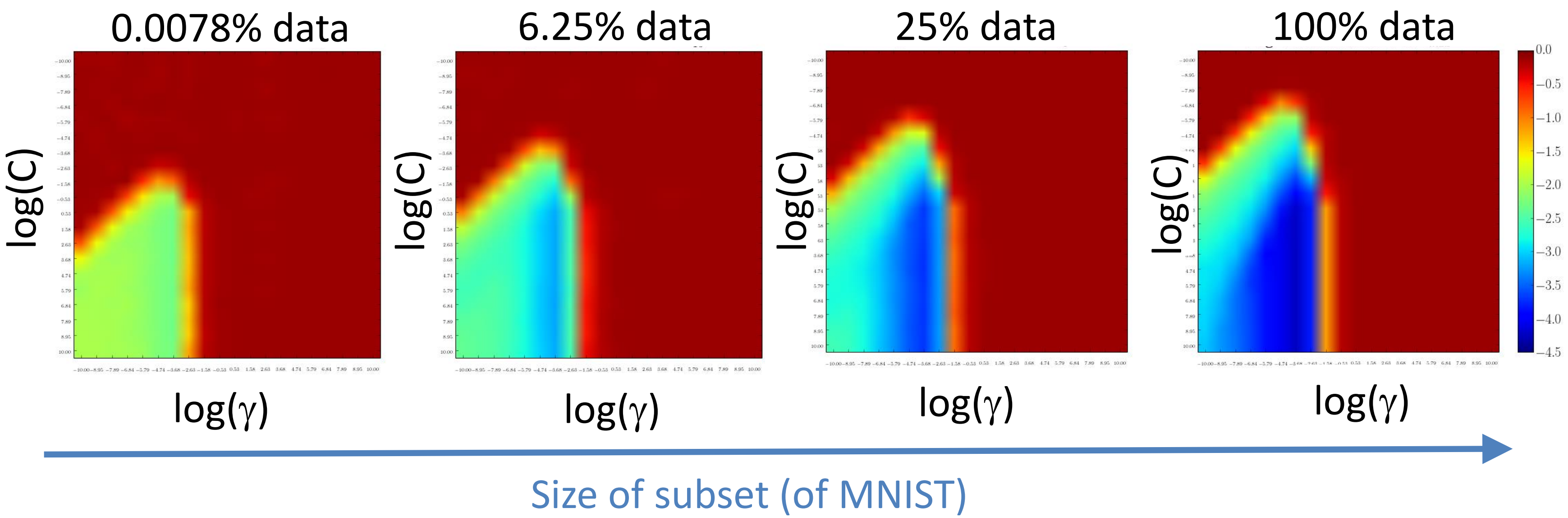
- **Make use of cheap low-fidelity evaluations**
  - E.g.: subsets of the data (here: SVM on MNIST)



  - Fit a Gaussian process model f(λ,b) to predict performance as a function of hyperparameters λ and budget b

Multi-fidelity Optimization

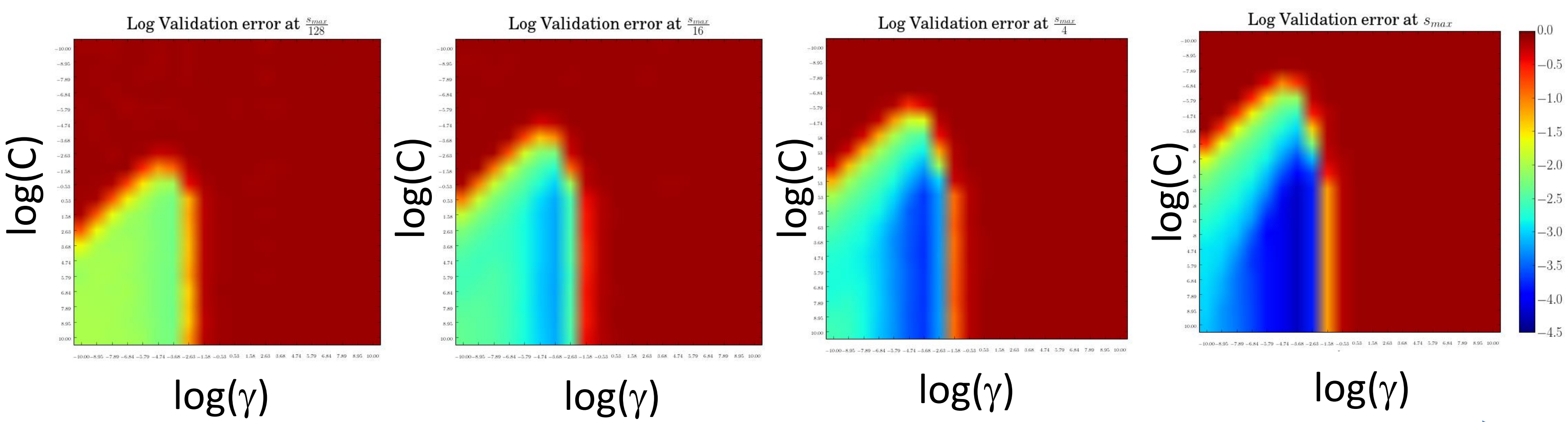
- **Make use of cheap low-fidelity evaluations**
  – E.g.: subsets of the data (here: SVM on MNIST)



  – Fit a Gaussian process model f(λ,b) to predict performance as a function of hyperparameters λ and budget b
  – Choose both λ and budget b to maximize “bang for the buck”

[Swersky et al, NeurIPS 2013; Swersky et al, arXiv 2014; Klein et al, AISTATS 2017; Kandasamy et al, ICML 2017]

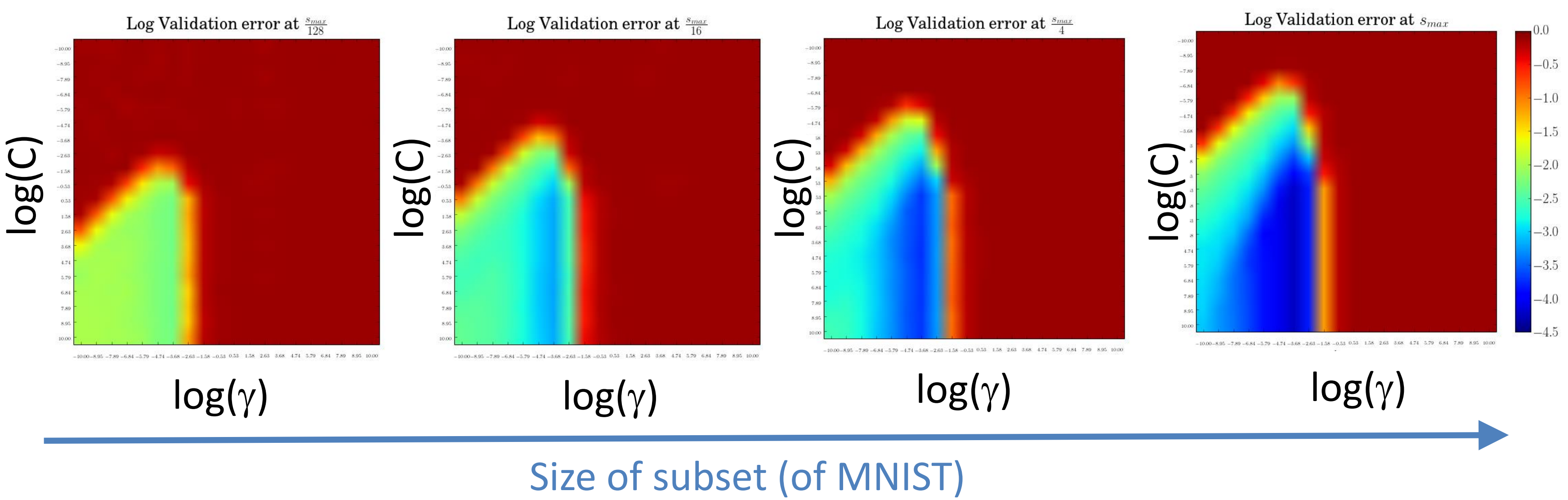
A Simpler Approach: Successive Halving (SH)



Size of subset (of MNIST)

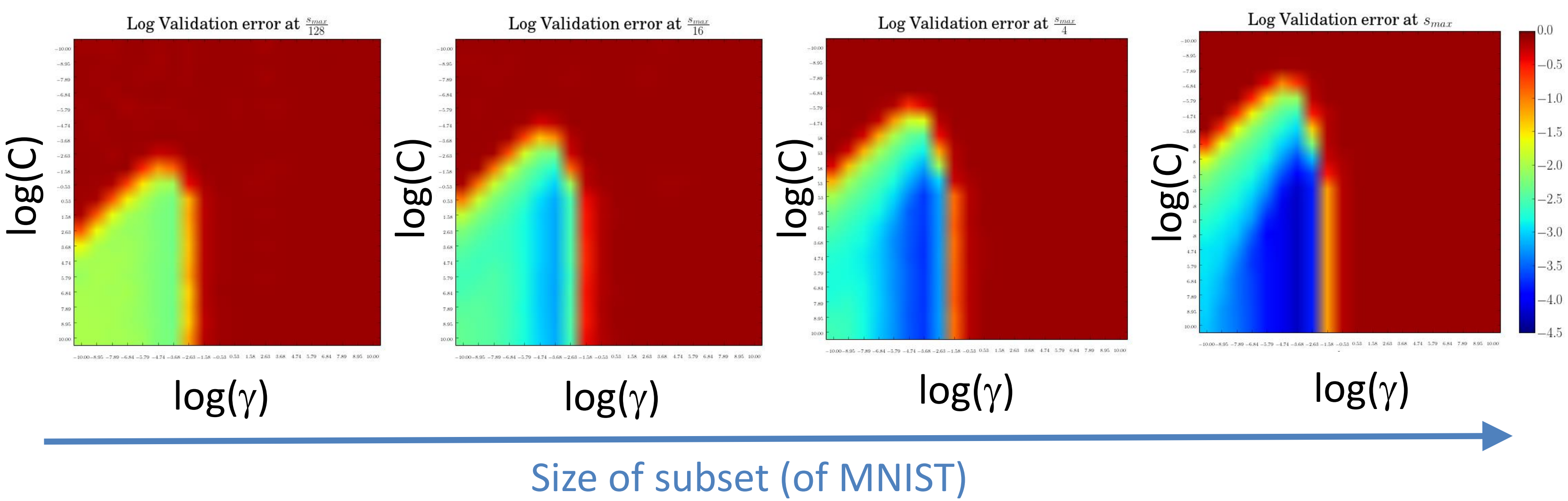
A Simpler Approach: Successive Halving (SH)

- **Idea:** Use a bandit to allocate more budget to promising configurations



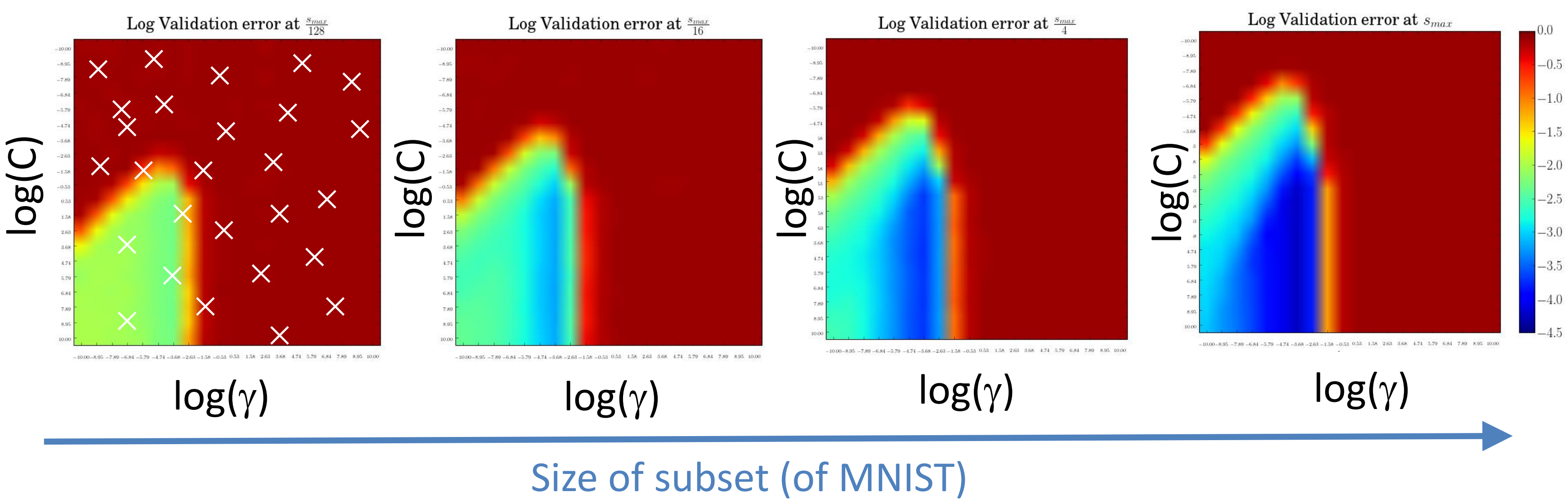
A Simpler Approach: Successive Halving (SH)

- **Idea:** Use a bandit to allocate more budget to promising configurations
- **Successive Halving** [Jamieson & Talwalkar, AISTATS 2016]
  - Randomly sample N configurations & evaluate on cheapest fidelity
  - Keep the top half, double its budget (or top third, triple budget)



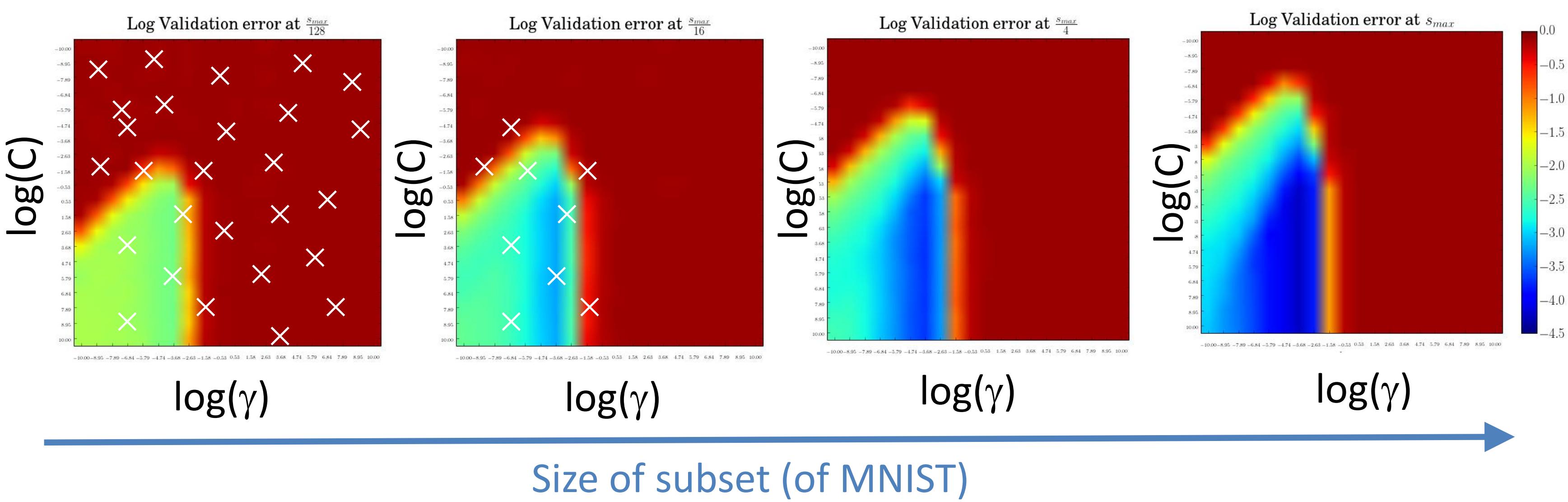
A Simpler Approach: Successive Halving (SH)

- **Idea:** Use a bandit to allocate more budget to promising configurations
- **Successive Halving** [Jamieson & Talwalkar, AISTATS 2016]
  - Randomly sample N configurations & evaluate on cheapest fidelity
  - Keep the top half, double its budget (or top third, triple budget)



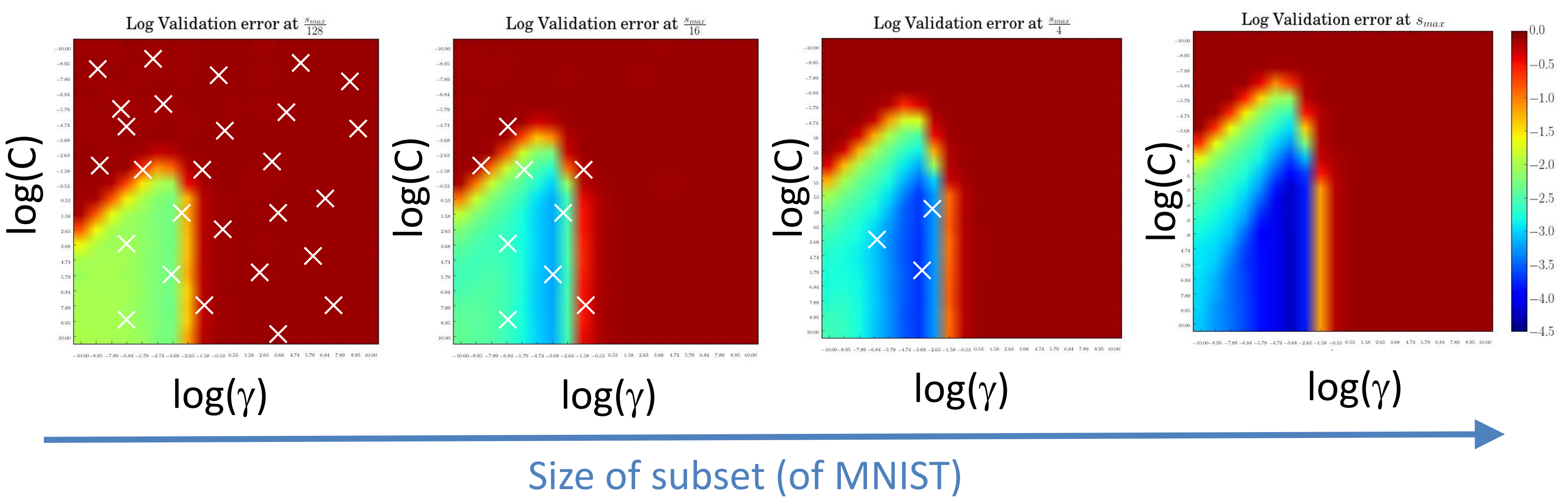
A Simpler Approach: Successive Halving (SH)

- **Idea:** Use a bandit to allocate more budget to promising configurations
- **Successive Halving** [Jamieson & Talwalkar, AISTATS 2016]
  - Randomly sample N configurations & evaluate on cheapest fidelity
  - Keep the top half, double its budget (or top third, triple budget)



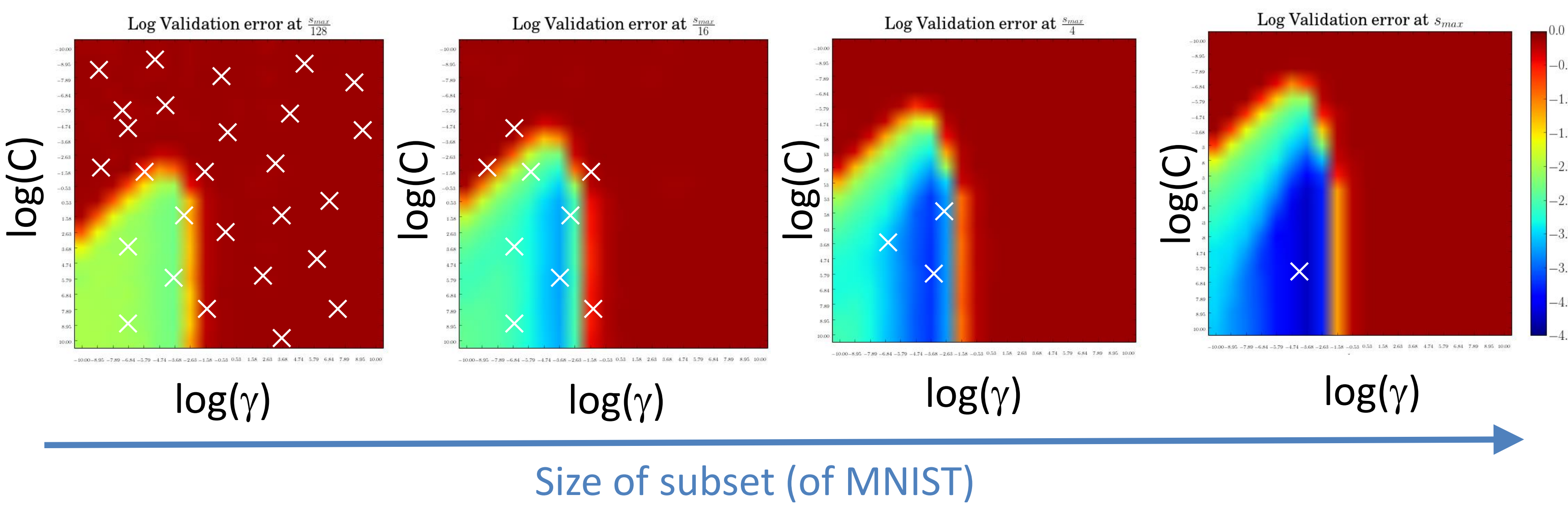
A Simpler Approach: Successive Halving (SH)

- **Idea:** Use a bandit to allocate more budget to promising configurations
- **Successive Halving** [Jamieson & Talwalkar, AISTATS 2016]
  - Randomly sample N configurations & evaluate on cheapest fidelity
  - Keep the top half, double its budget (or top third, triple budget)



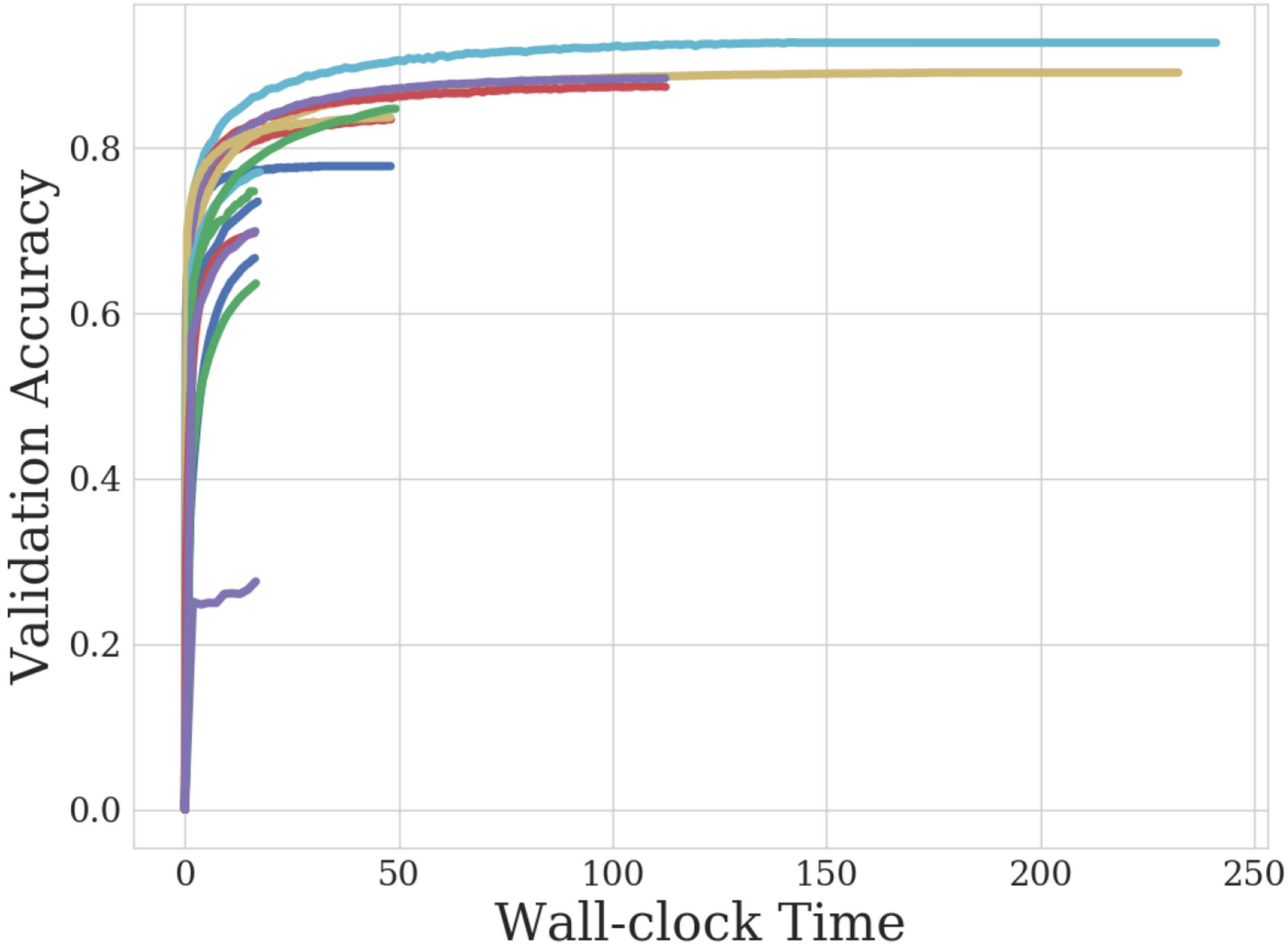
A Simpler Approach: Successive Halving (SH)

- **Idea:** Use a bandit to allocate more budget to promising configurations
- **Successive Halving** [Jamieson & Talwalkar, AISTATS 2016]
  - Randomly sample N configurations & evaluate on cheapest fidelity
  - Keep the top half, double its budget (or top third, triple budget)



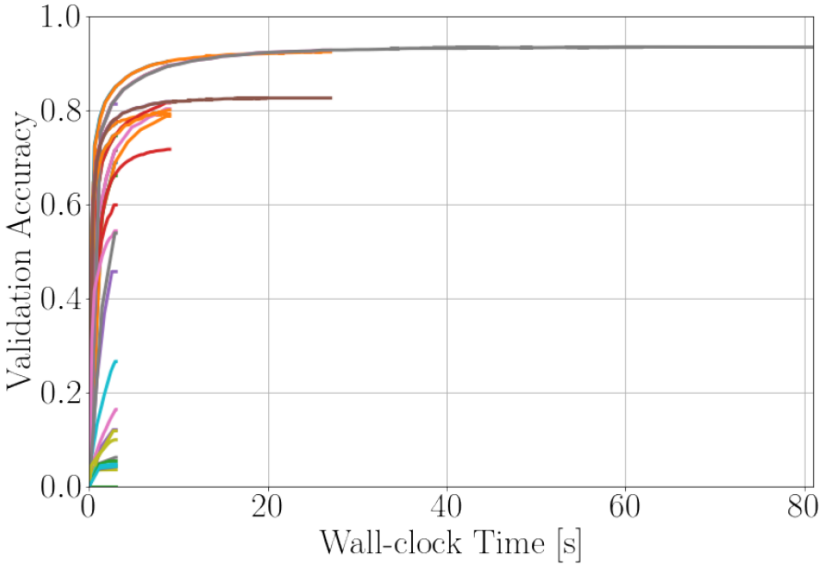
A Simpler Approach: Successive Halving (SH)

[Jamieson & Talwalkar, AISTATS 2016]



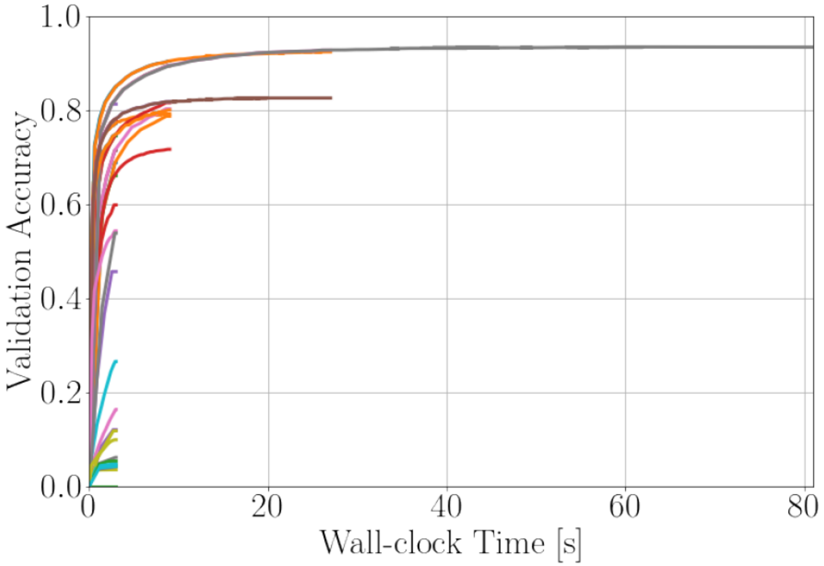
Hyperband (its first 4 calls to SH)

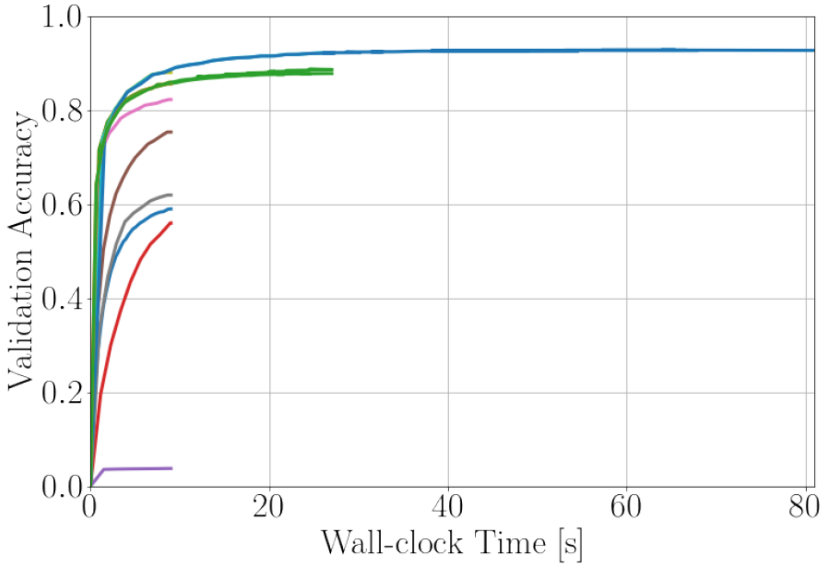
[Li et al, JMLR 2018]



Hyperband (its first 4 calls to SH)

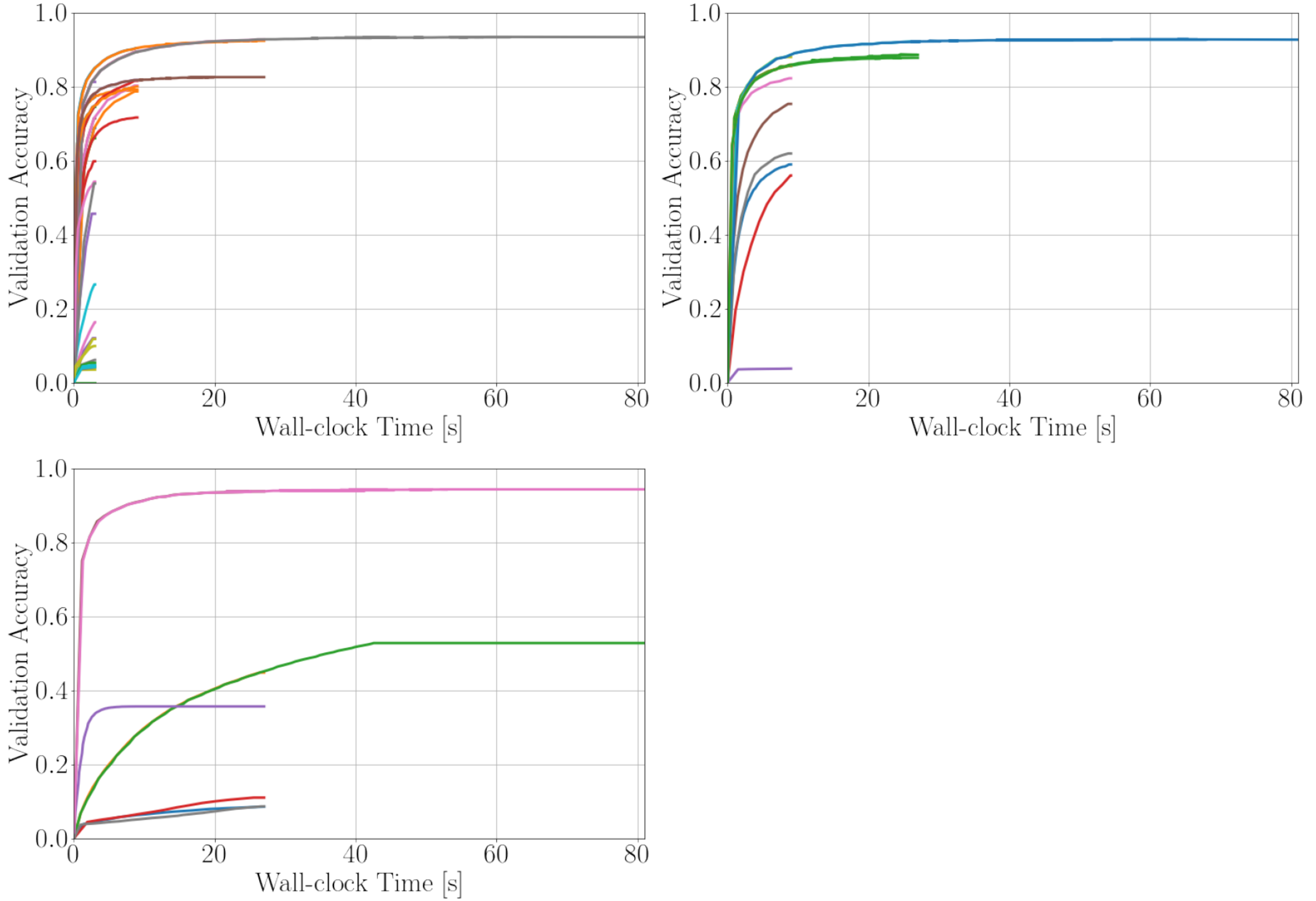
[Li et al, JMLR 2018]





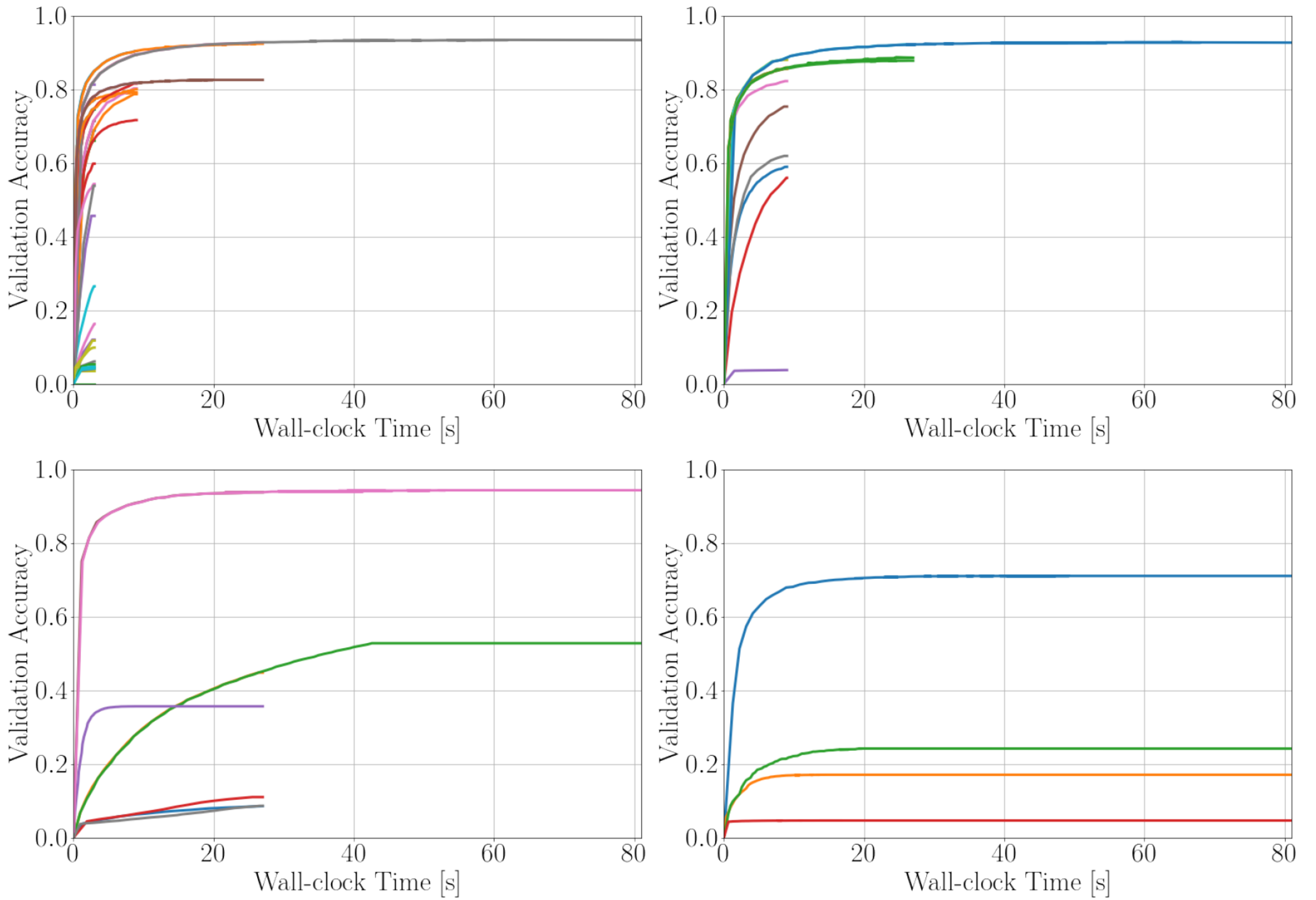
Hyperband (its first 4 calls to SH)

[Li et al, JMLR 2018]



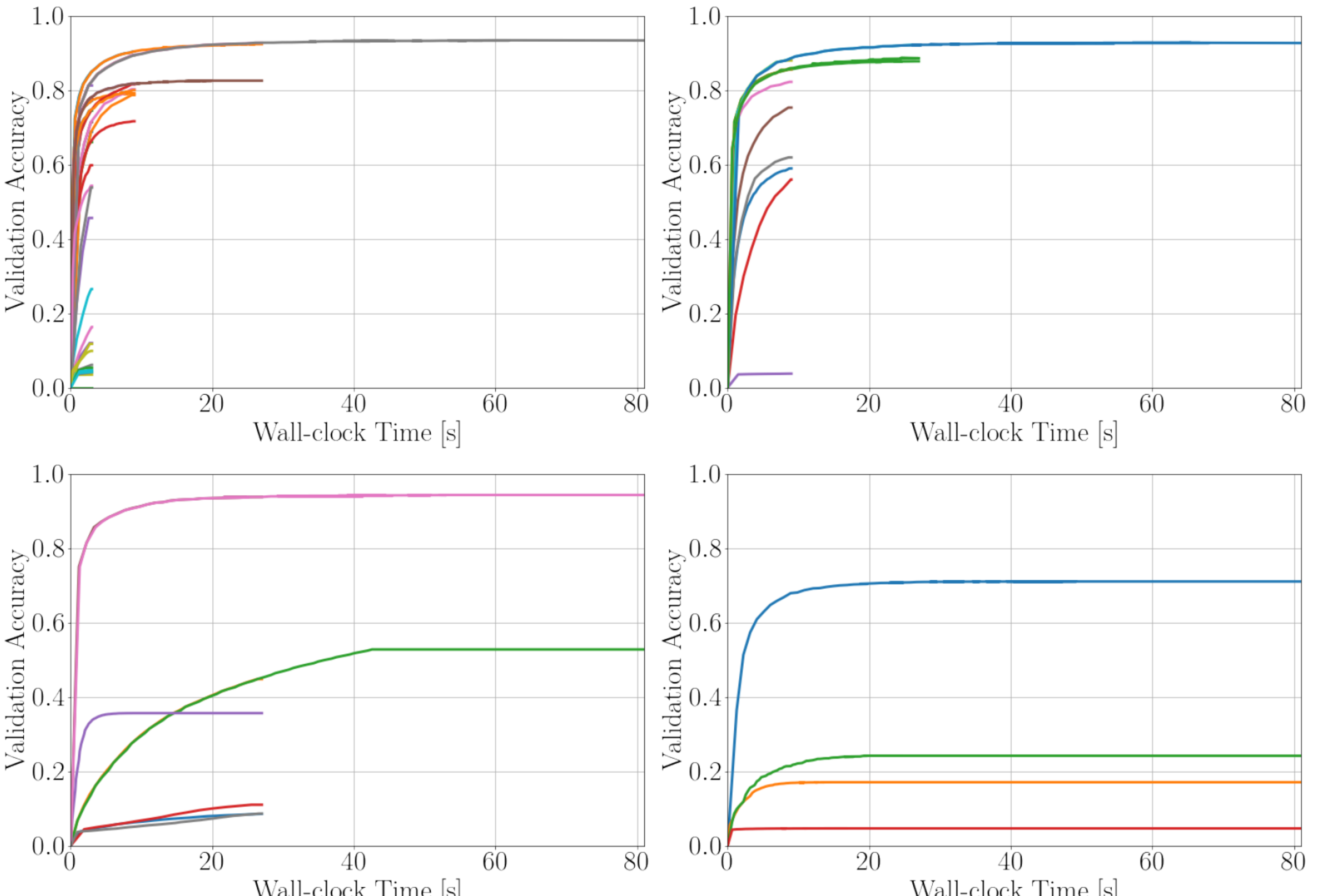
Hyperband (its first 4 calls to SH)

[Li et al, JMLR 2018]



Hyperband (its first 4 calls to SH)

[Li et al, JMLR 2018]



Each call to Successive Halving takes roughly the same amount of wallclock time!

BOHB: Bayesian Optimization & Hyperband

[Falkner, Klein & Hutter, ICML 2018]

- Advantages of Hyperband
  - Strong anytime performance
  - General-purpose
    - Low-dimensional continuous spaces
    - High-dimensional spaces with conditionality, categorical dimensions, etc
  - Easy to implement
  - Scalable
  - Easily parallelizable

BOHB: Bayesian Optimization & Hyperband

[Falkner, Klein & Hutter, ICML 2018]

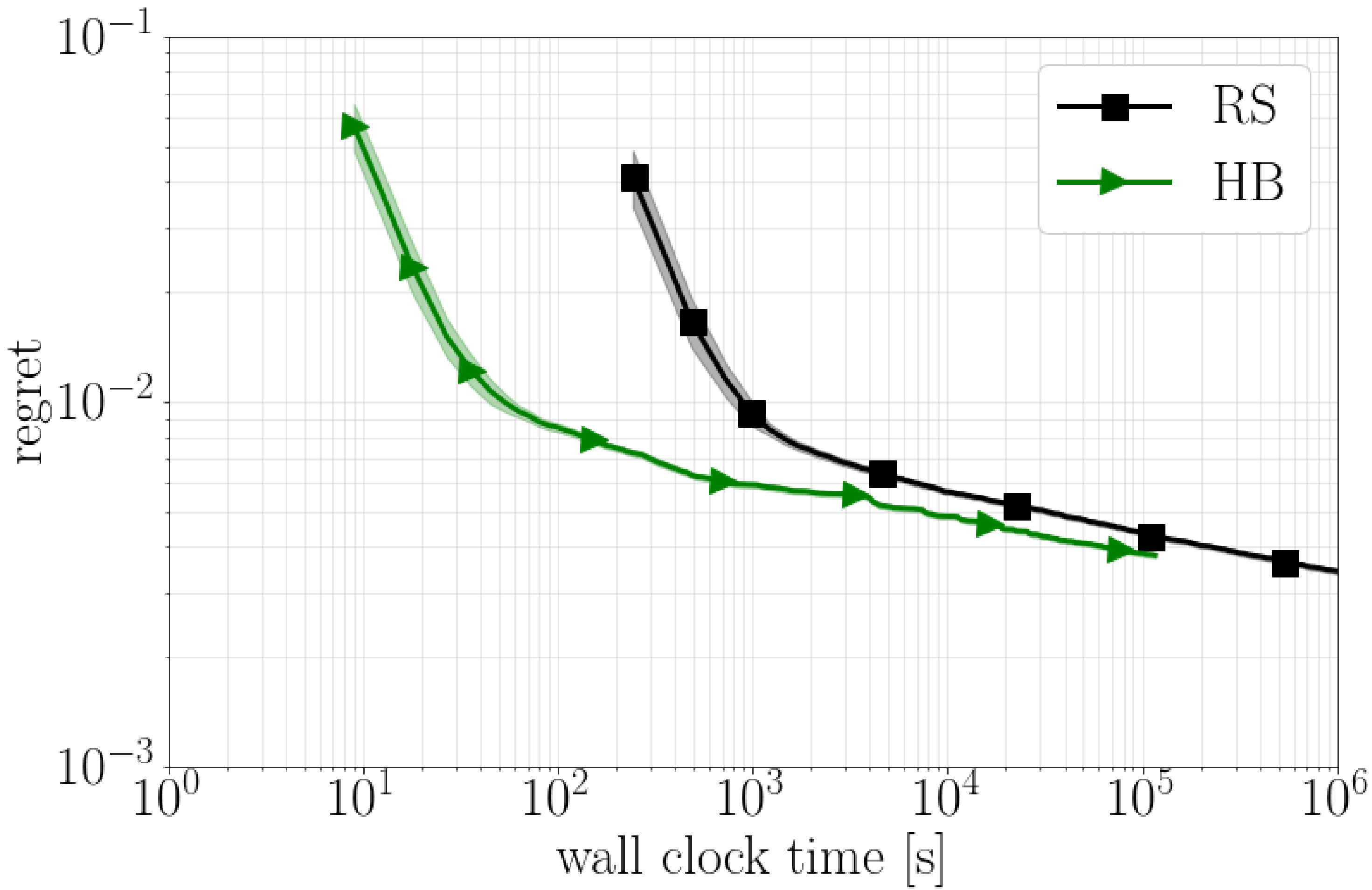
- Advantages of Hyperband
  - Strong anytime performance
  - General-purpose
    - Low-dimensional continuous spaces
    - High-dimensional spaces with conditionality, categorical dimensions, etc
  - Easy to implement
  - Scalable
  - Easily parallelizable
- Advantage of Bayesian optimization: strong final performance

BOHB: Bayesian Optimization & Hyperband

[Falkner, Klein & Hutter, ICML 2018]

- Advantages of Hyperband
  - Strong anytime performance
  - General-purpose
    - Low-dimensional continuous spaces
    - High-dimensional spaces with conditionality, categorical dimensions, etc
  - Easy to implement
  - Scalable
  - Easily parallelizable
- Advantage of Bayesian optimization: strong final performance
- Combining the best of both worlds in BOHB
  - Bayesian optimization
    - for choosing the configurations to evaluate (using a TPE variant)
  - Hyperband
    - for deciding how to allocate budgets

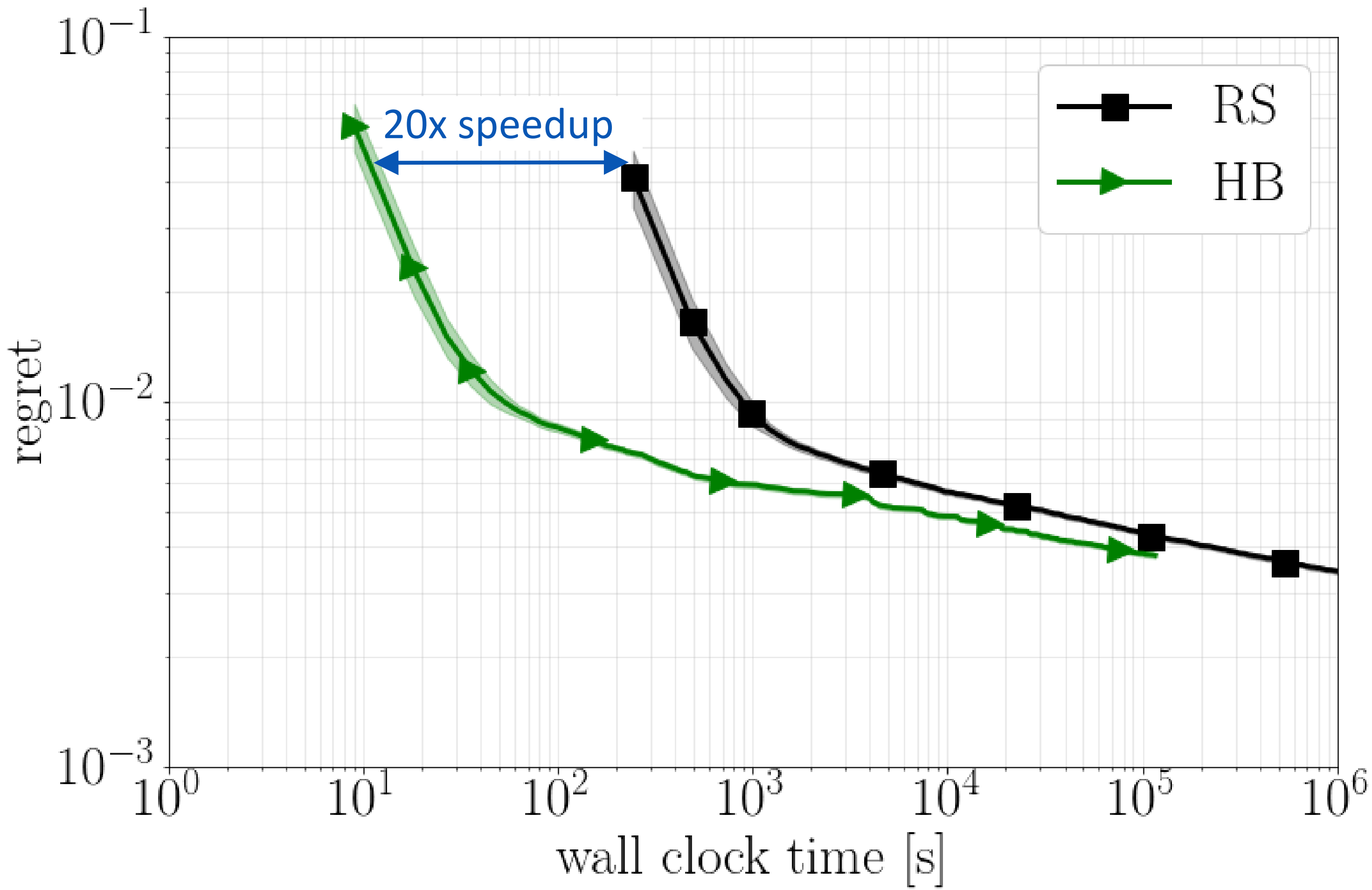
Hyperband vs. Random Search



Biggest advantage: much improved **anytime performance**

Auto-Net on dataset adult

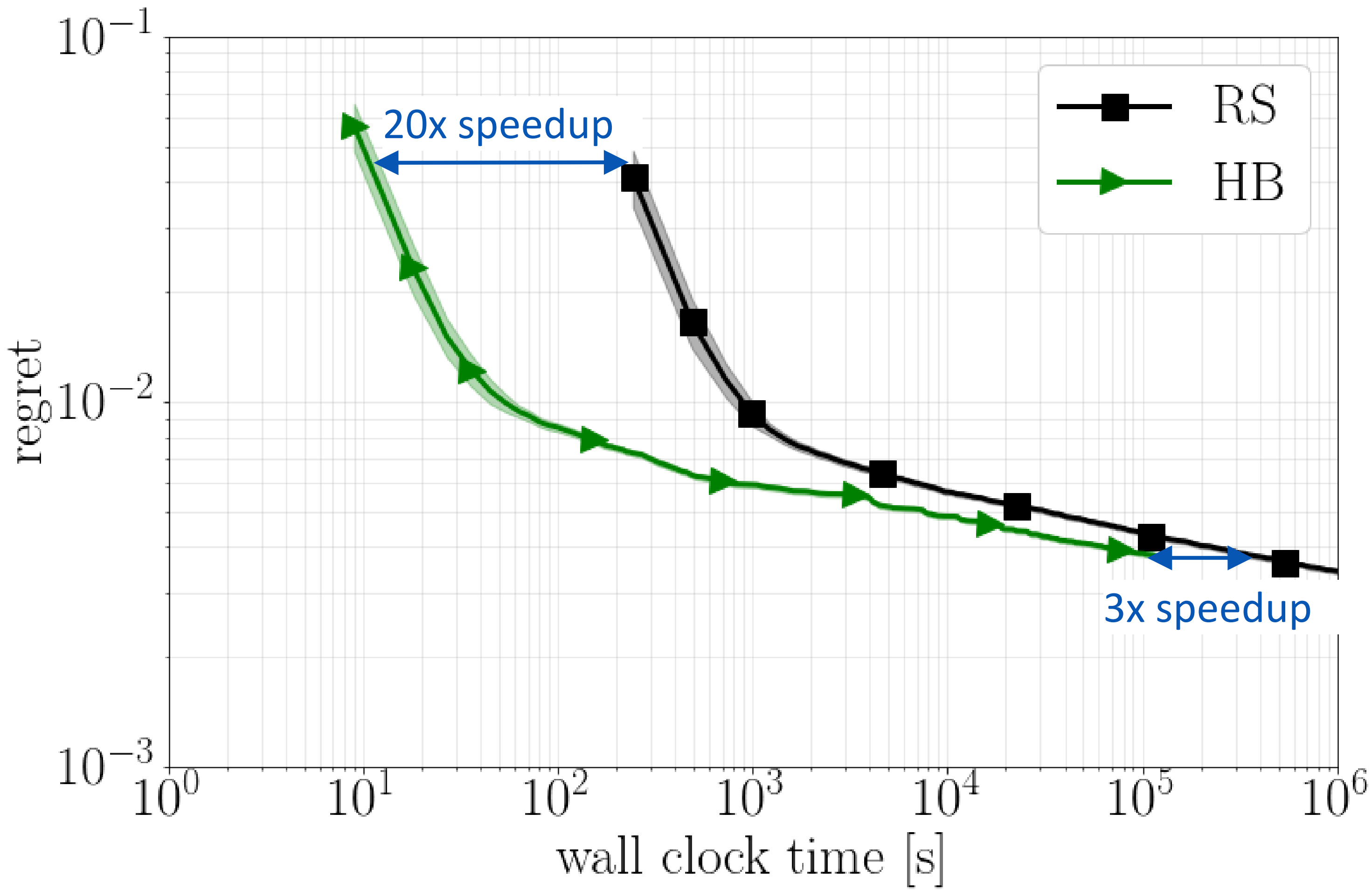
Hyperband vs. Random Search



Biggest advantage: much improved **anytime performance**

Auto-Net on dataset adult

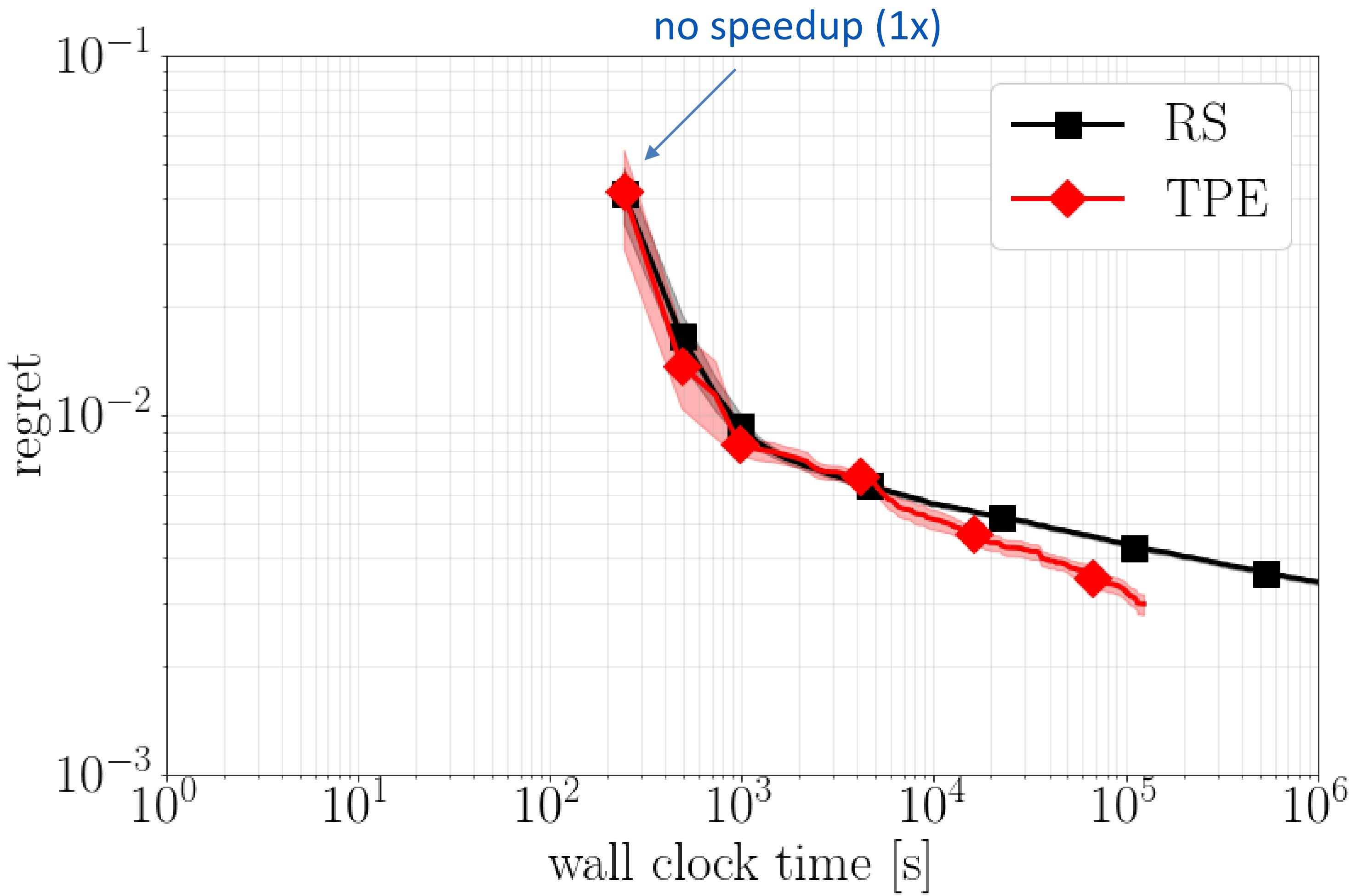
Hyperband vs. Random Search



Biggest advantage: much improved **anytime performance**

Auto-Net on dataset adult

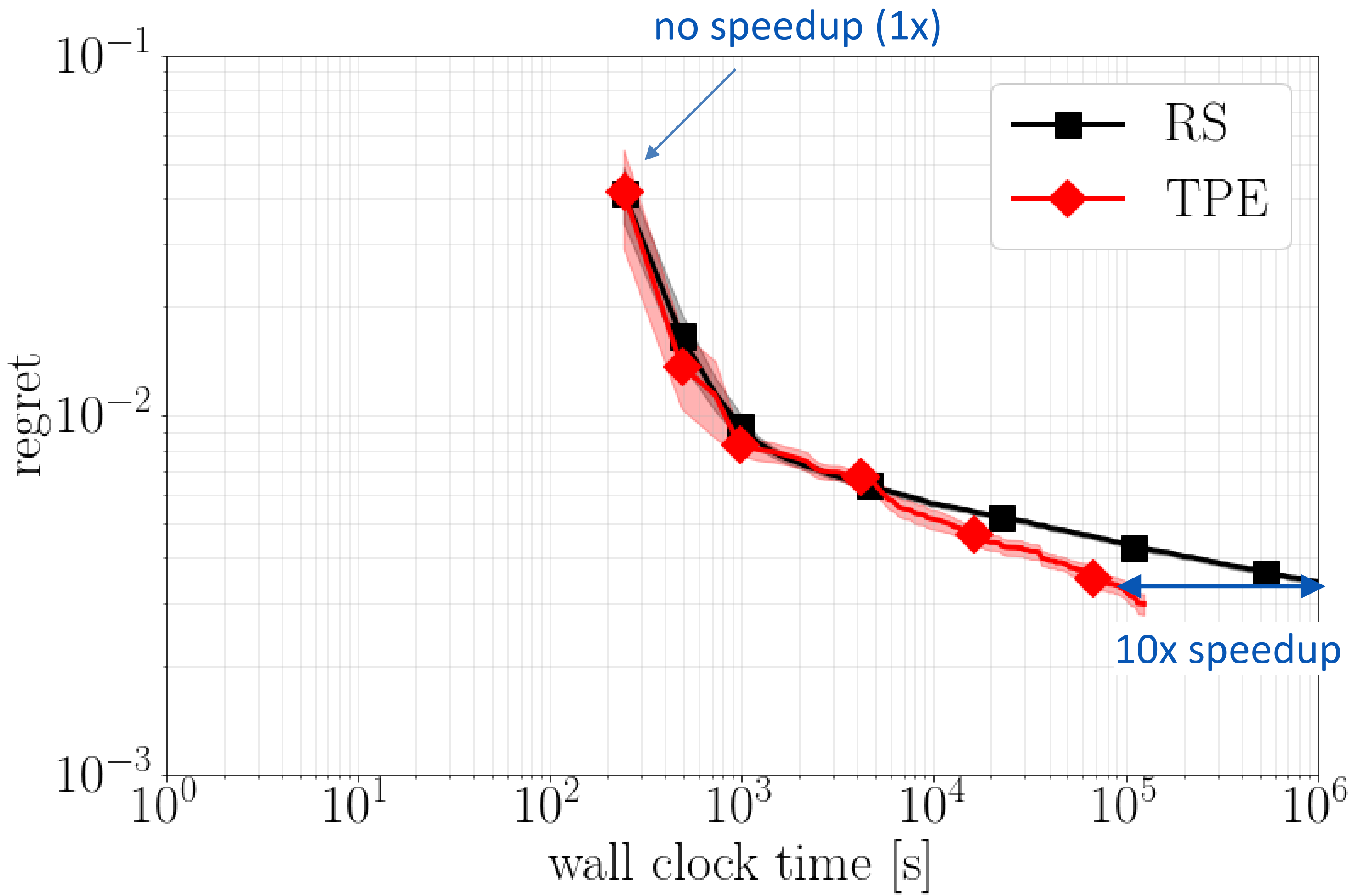
Bayesian Optimization vs Random Search



Biggest advantage: much improved **final performance**

Auto-Net on dataset adult

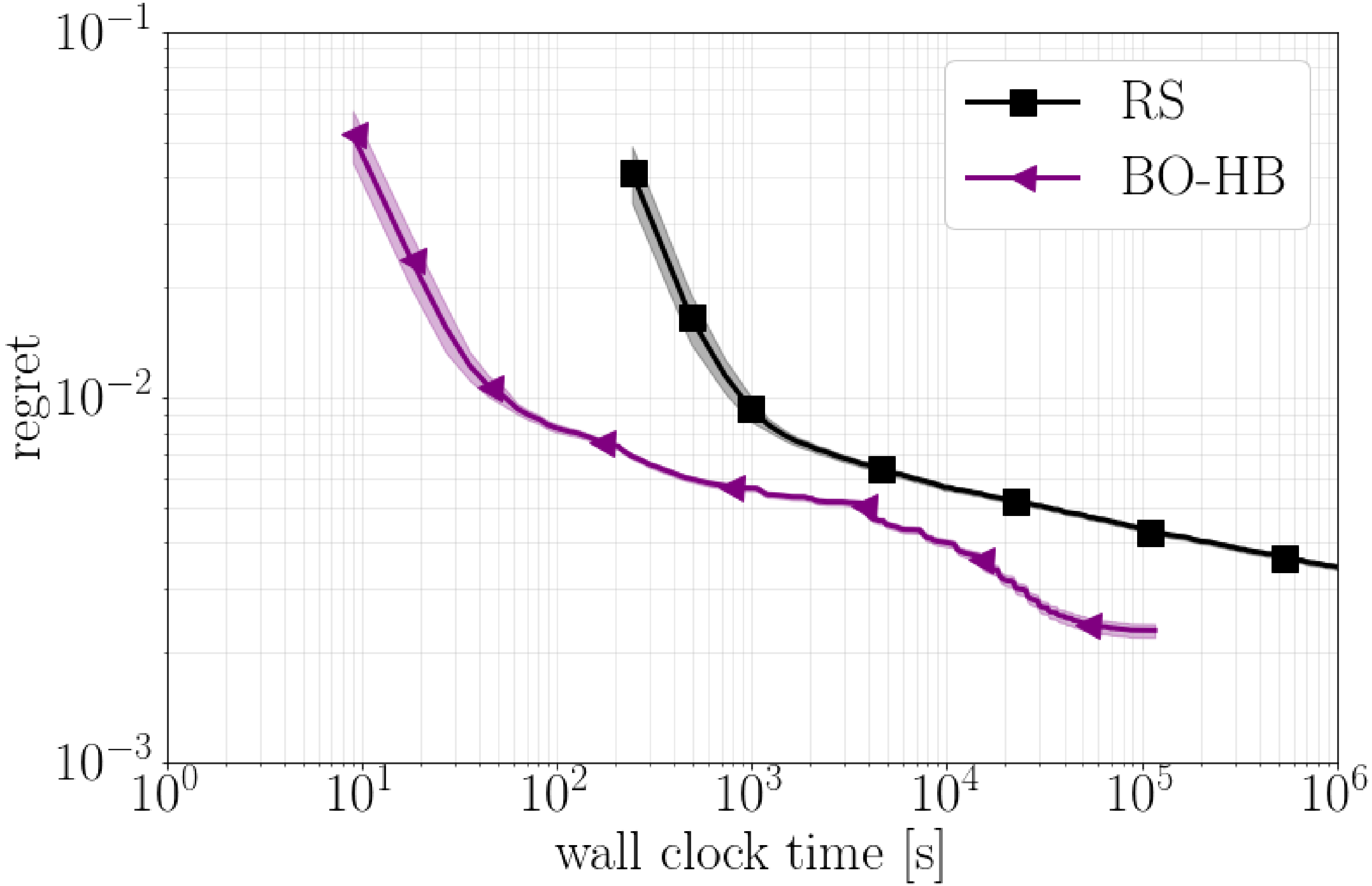
Bayesian Optimization vs Random Search



Biggest advantage: much improved **final performance**

Auto-Net on dataset adult

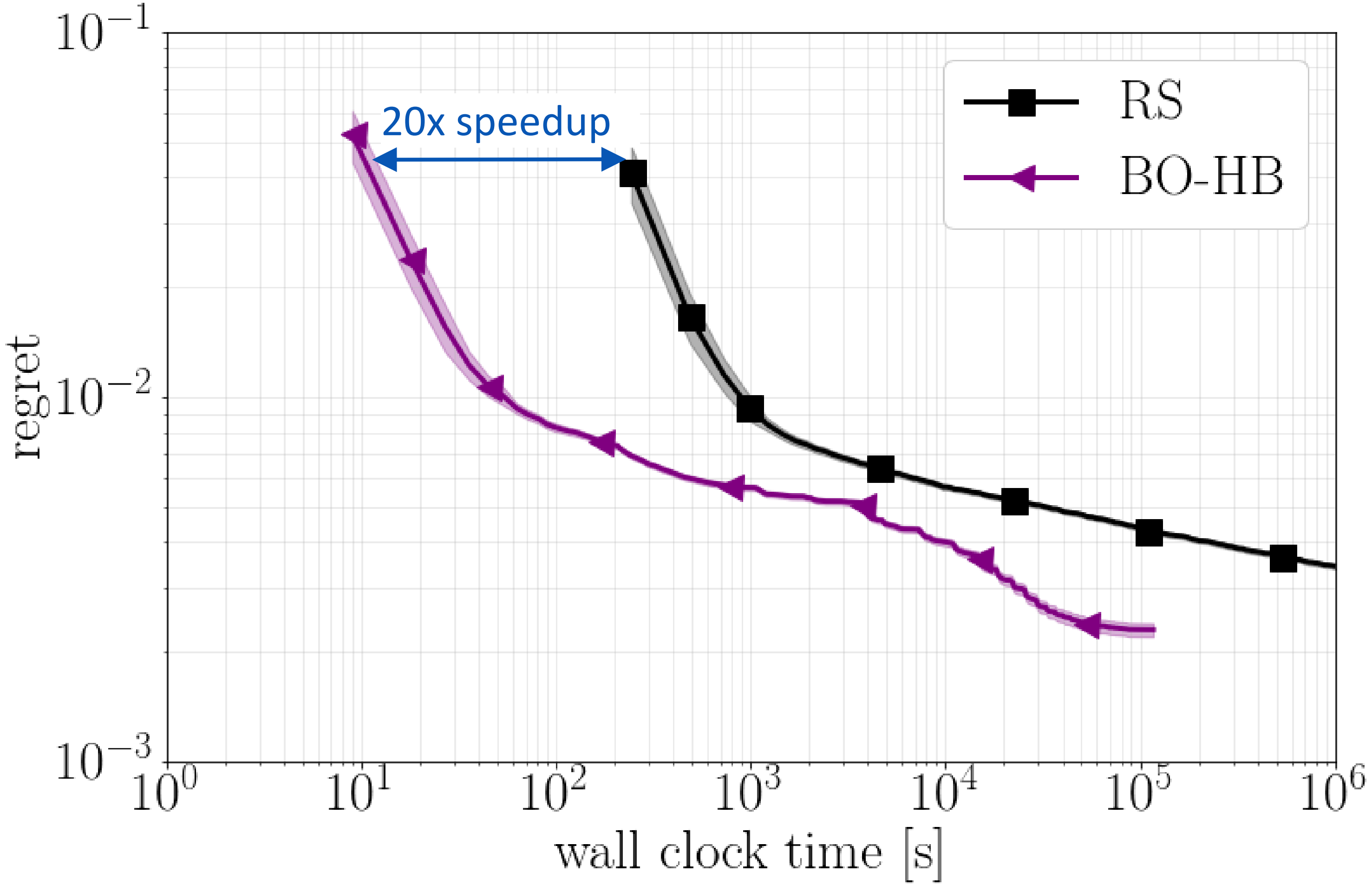
Combining Bayesian Optimization & Hyperband



Best of both worlds: strong **anytime and final performance**

Auto-Net on dataset adult

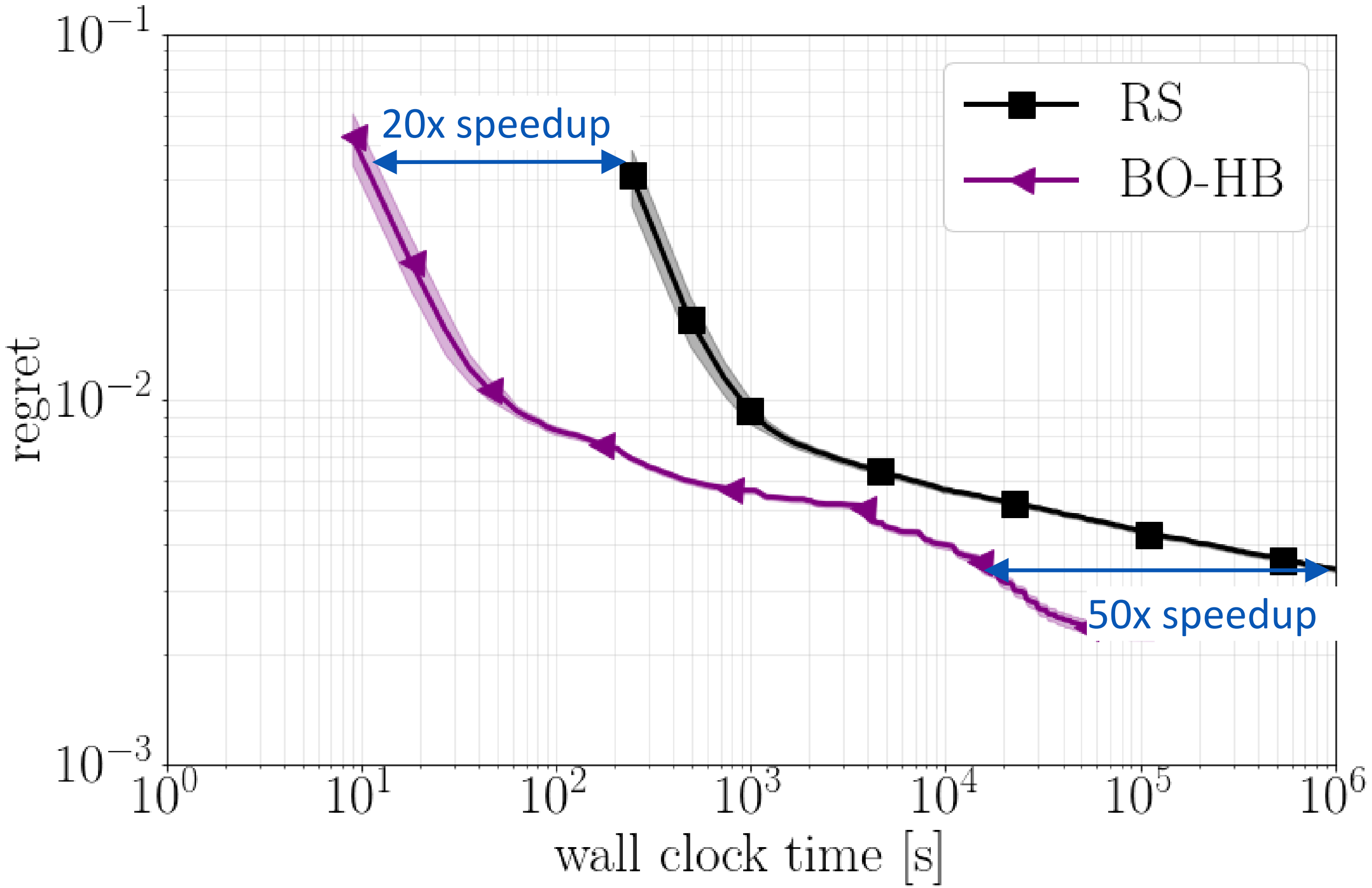
Combining Bayesian Optimization & Hyperband



Best of both worlds: strong **anytime and final performance**

Auto-Net on dataset adult

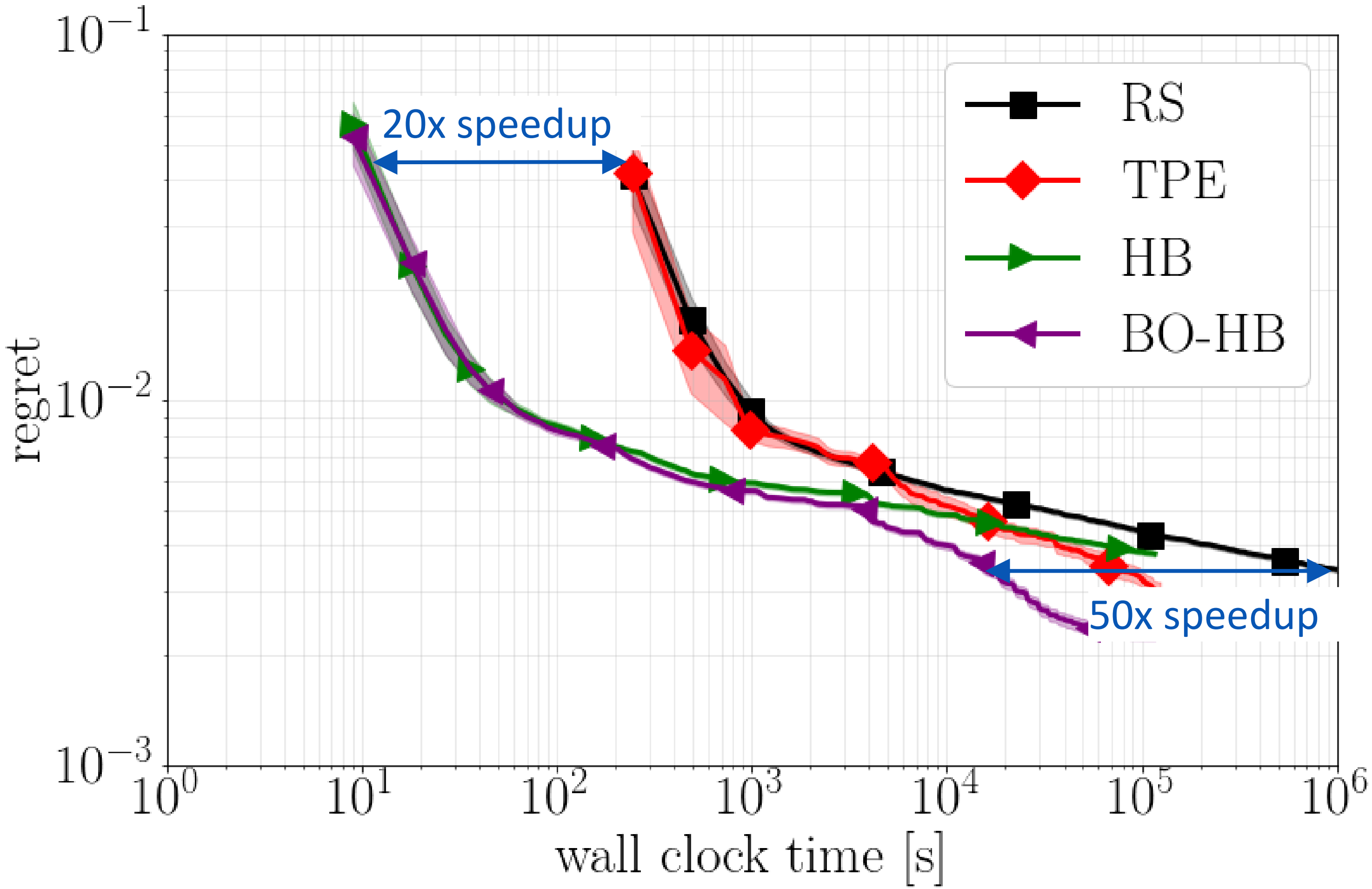
Combining Bayesian Optimization & Hyperband



Best of both worlds: strong **anytime and final performance**

Auto-Net on dataset adult

Combining Bayesian Optimization & Hyperband



Best of both worlds: strong **anytime and final performance**

Auto-Net on dataset adult

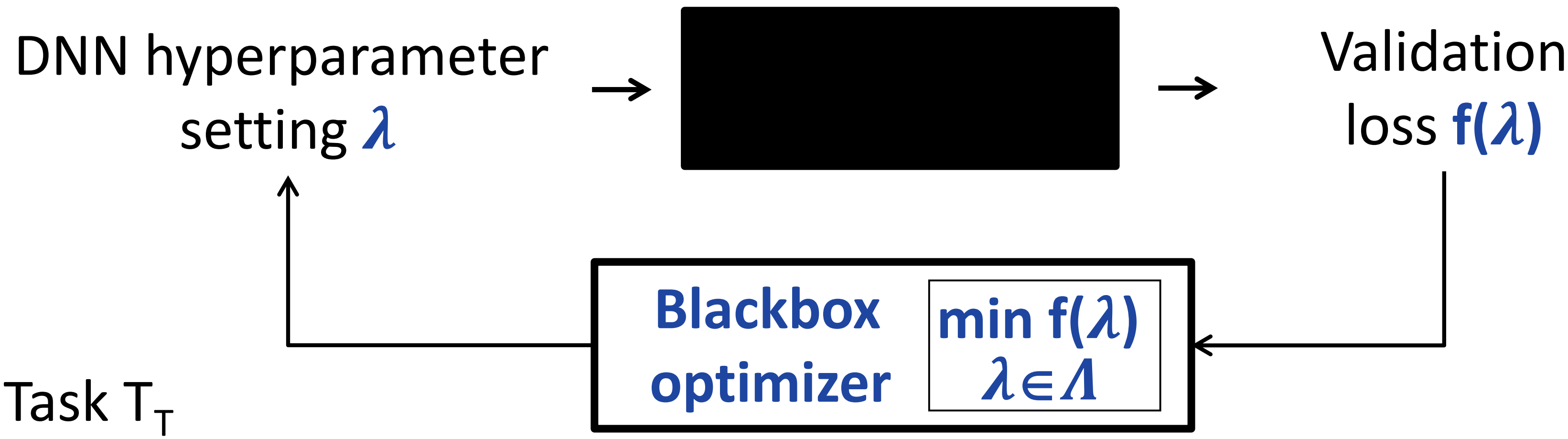
Outline

Part 1: General AutoML

1. AutoML by Hyperparameter Optimization
2. Black-box Hyperparameter Optimization
3. Beyond black-box optimization
4. Meta-learning
5. Examples of AutoML
6. Open issues and future work
7. Wrap-up & Conclusion

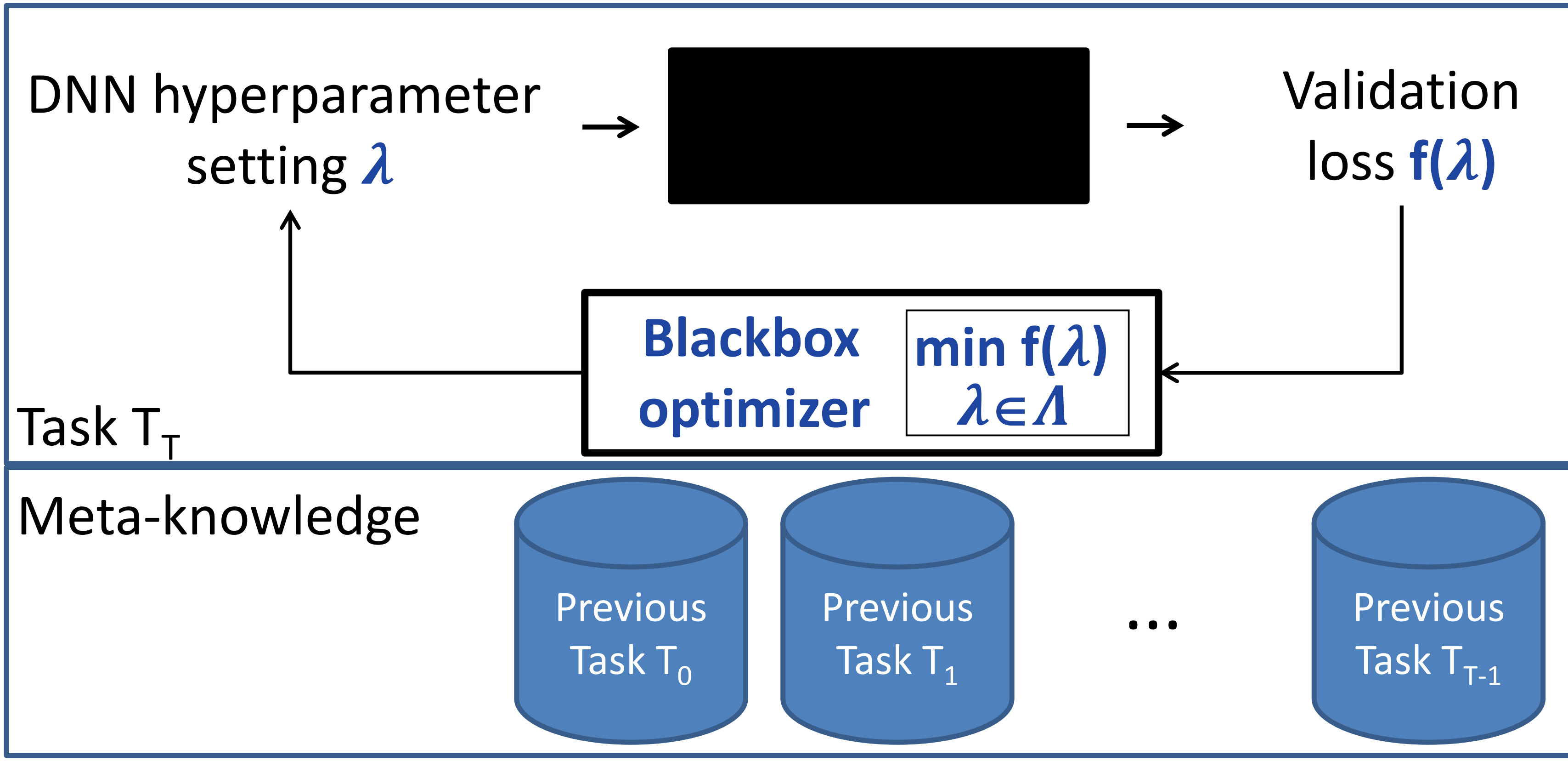
Part 2: Neural Architecture Search & Meta-Learning

Blackbox Hyperparameter Optimization

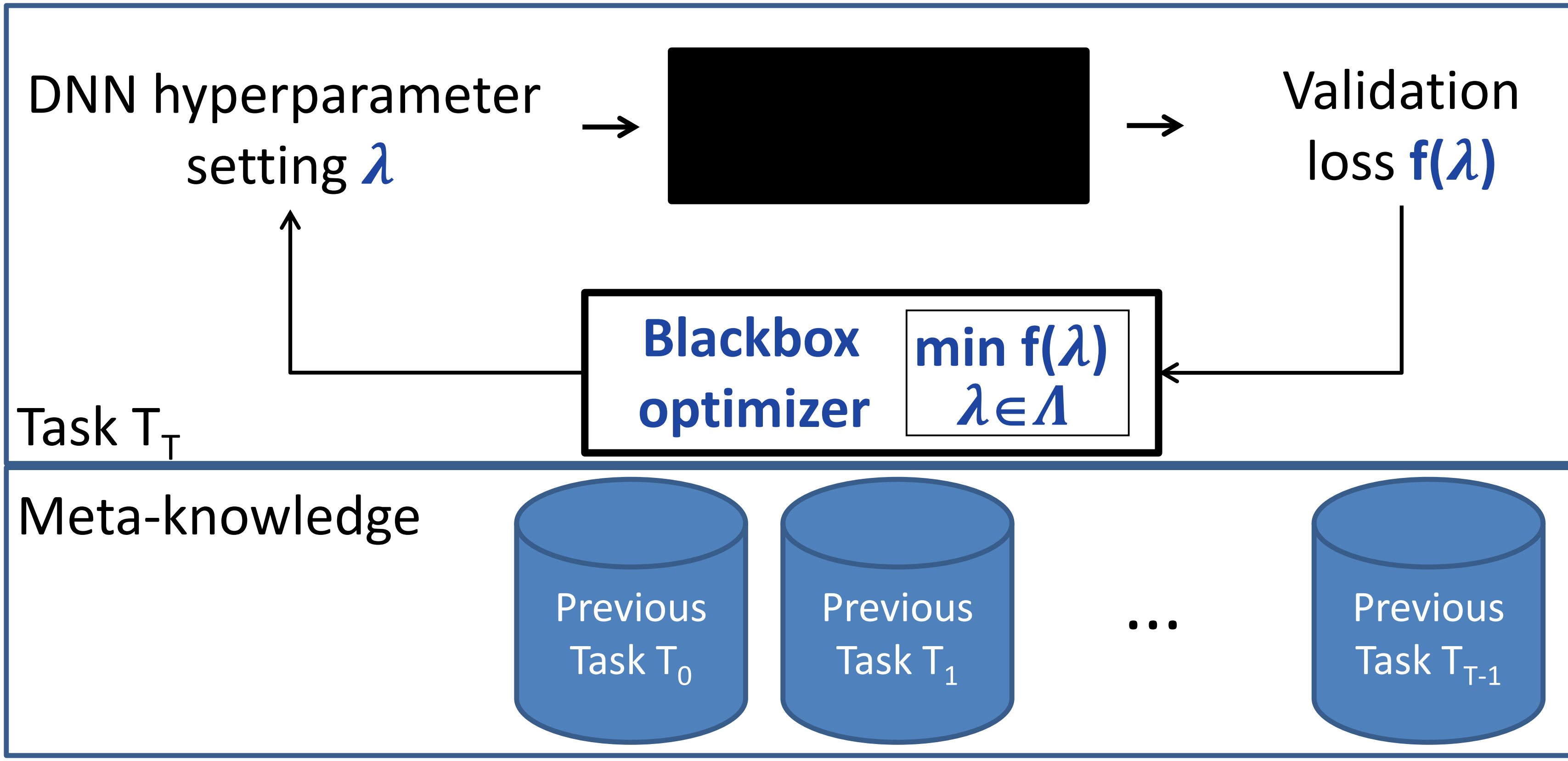


Task T_T

Blackbox Hyperparameter Optimization



Blackbox Hyperparameter Optimization



Analogy to manual hyperparameter optimization:

- Accumulate knowledge over time
- Use Knowledge when optimizing on a new dataset

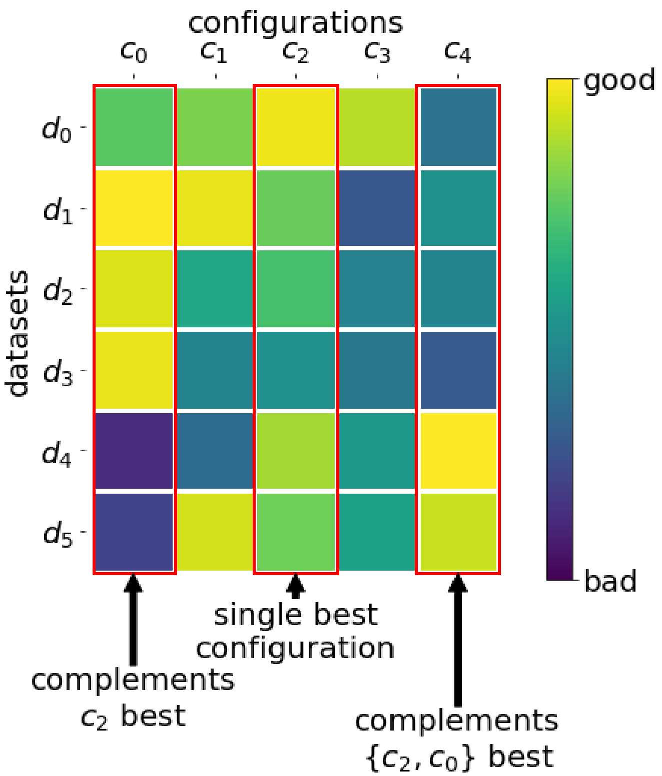
Task-independent recommendations

- Idea: learn a sorted list of defaults
- Advantages:
  - Easy to share and use
  - Strong anytime performance
  - Embarassingly parallel
- Disadvantages:
  - Not adaptive

[Wistuba et al., 2015a,&b, Feurer et al., 2018, Pfisterer et al., 2018]

Task-independent recommendations

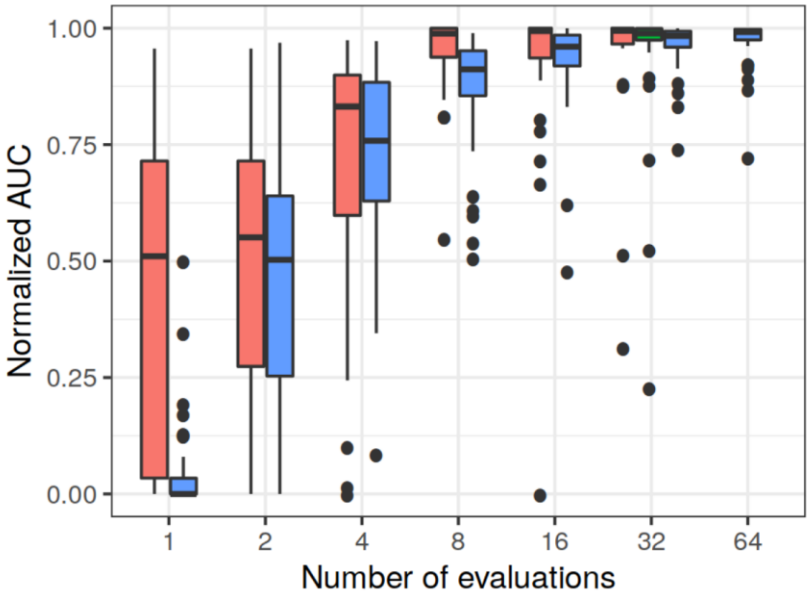
- Idea: learn a sorted list of defaults
- Advantages:
  - Easy to share and use
  - Strong anytime performance
  - Embarassingly parallel
- Disadvantages:
  - Not adaptive
- Method:
  - Mostly greedy



[Wistuba et al., 2015a,&b, Feurer et al., 2018, Pfisterer et al., 2018]

Task-independent recommendations

- Idea: learn a sorted list of defaults
- Advantages:
  - Easy to share and use
  - Strong anytime performance
  - Embarassingly parallel
- Disadvantages:
  - Not adaptive
- Method:
  - Mostly greedy
- Results
  - Improves over Random Search and Bayesian Optimization

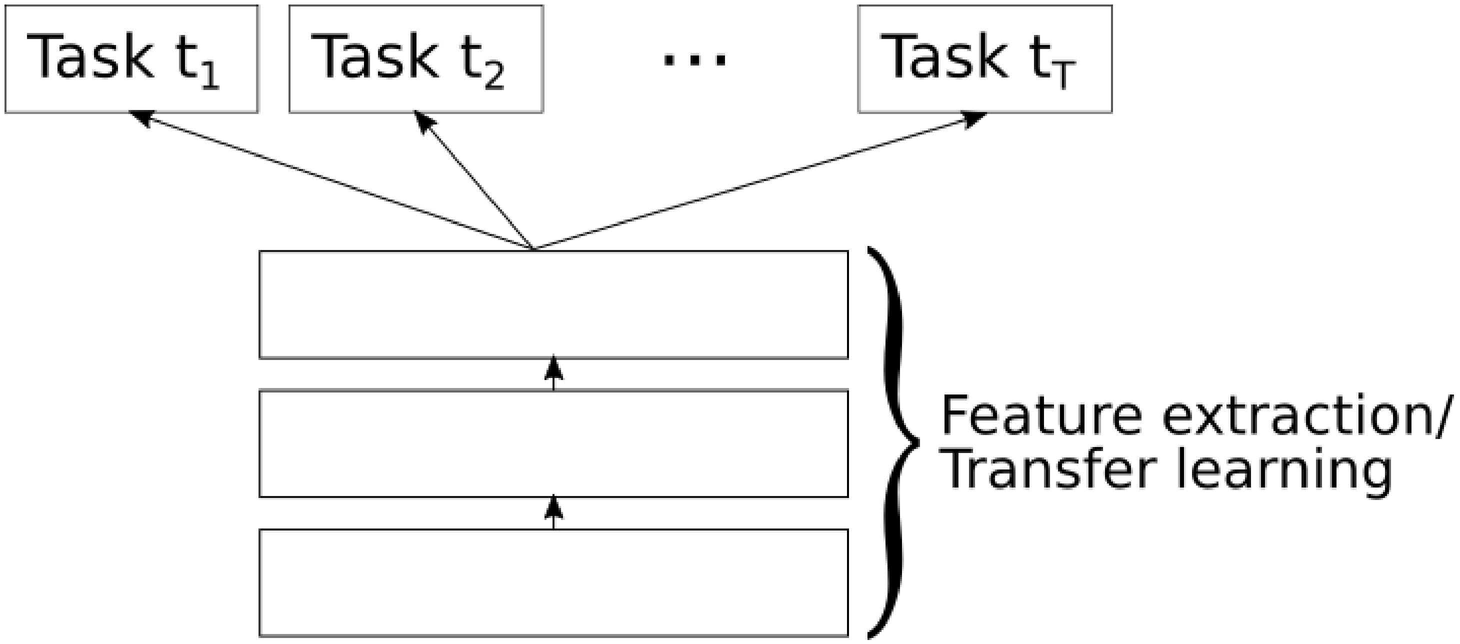


[Wistuba et al., 2015a,&b, Feurer et al., 2018, Pfisterer et al., 2018]

Joint model for Bayesian optimization

Joint model for Bayesian optimization

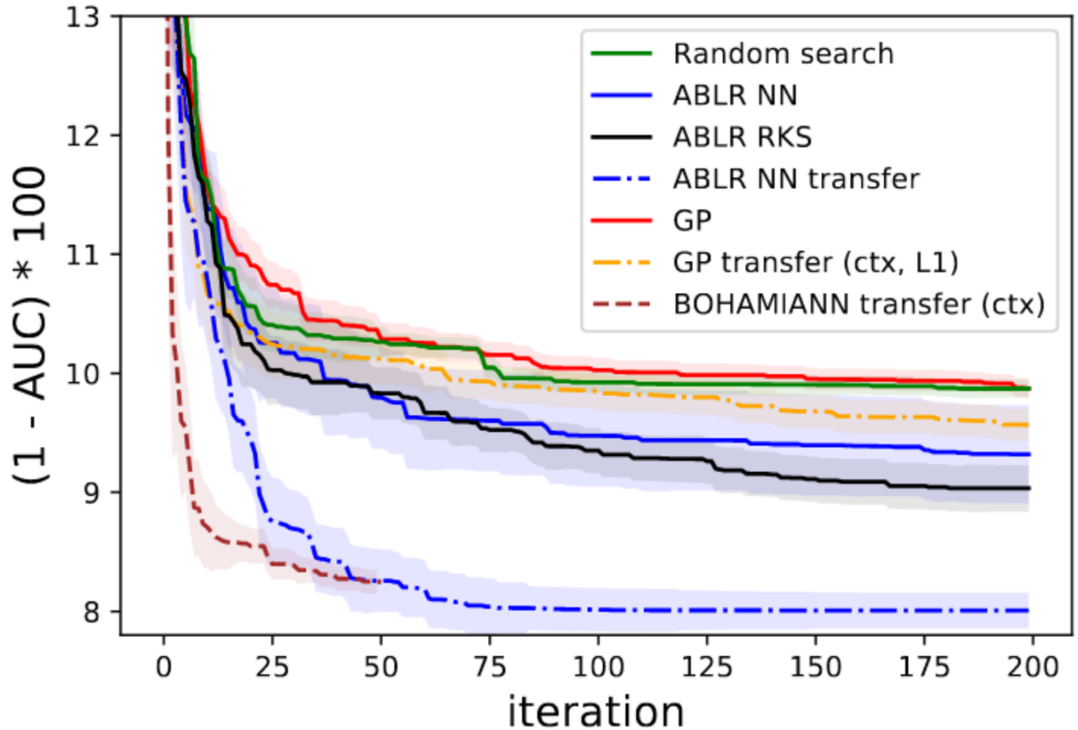
- Jointly train a „deep“ neural network on all tasks
- Have a separate output layer (head) for each tasks
- Each head is a Bayesian linear regression
- Feature extraction on hyperparameter configurations



[Perrone et al., NeurIPS 2018]

Joint model for Bayesian optimization

- Jointly train a „deep“ neural network on all tasks
- Have a separate output layer (head) for each tasks
- Each head is a Bayesian linear regression
- Feature extraction on hyperparameter configurations



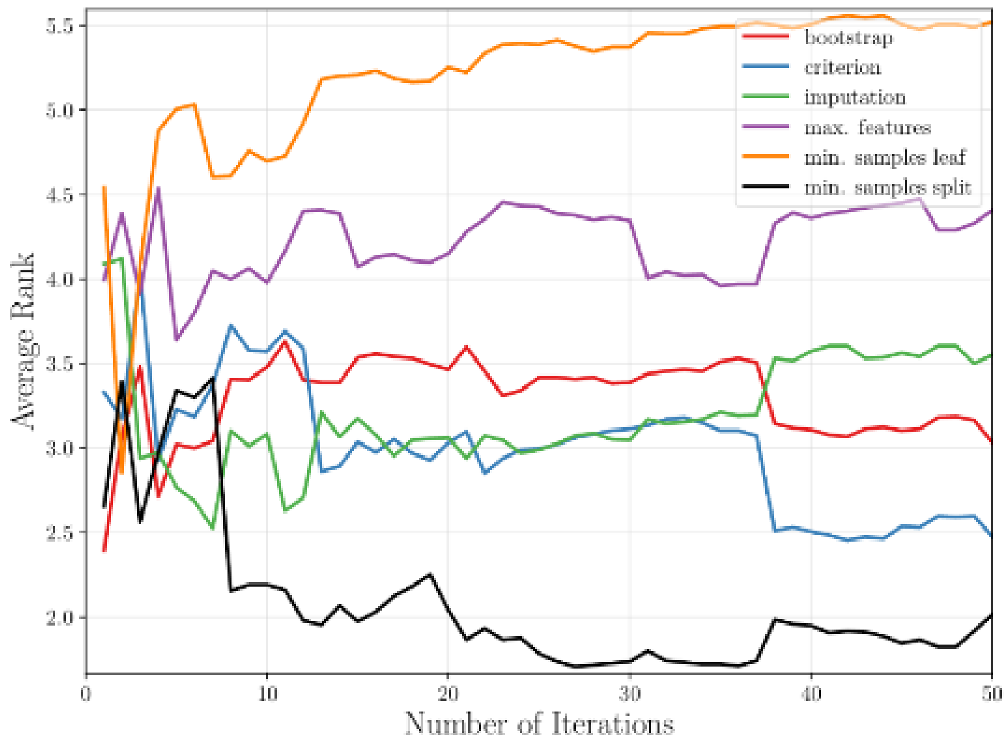
[Perrone et al., NeurIPS 2018]

Analyzing the effect of hyperparameters

- Search Space Pruning [Wistuba et al., ECMLPKDD 2015]
  - Rate all candidate configurations by their potential on past datasets
  - Drop the ones with low potential (plus some space around)

Analyzing the effect of hyperparameters

- Search Space Pruning [Wistuba et al., ECMLPKDD 2015]
  - Rate all candidate configurations by their potential on past datasets
  - Drop the ones with low potential (plus some space around)
- Hyperparameter importance [van Rijn and Hutter, KDD 2018]



Outline

Part 1: General AutoML

1. AutoML by Hyperparameter Optimization
2. Black-box Hyperparameter Optimization
3. Beyond black-box optimization
4. Meta-learning
5. Examples of AutoML
6. Open issues and future work
7. Wrap-up & Conclusion

Part 2: Neural Architecture Search & Meta-Learning

What can be automated?

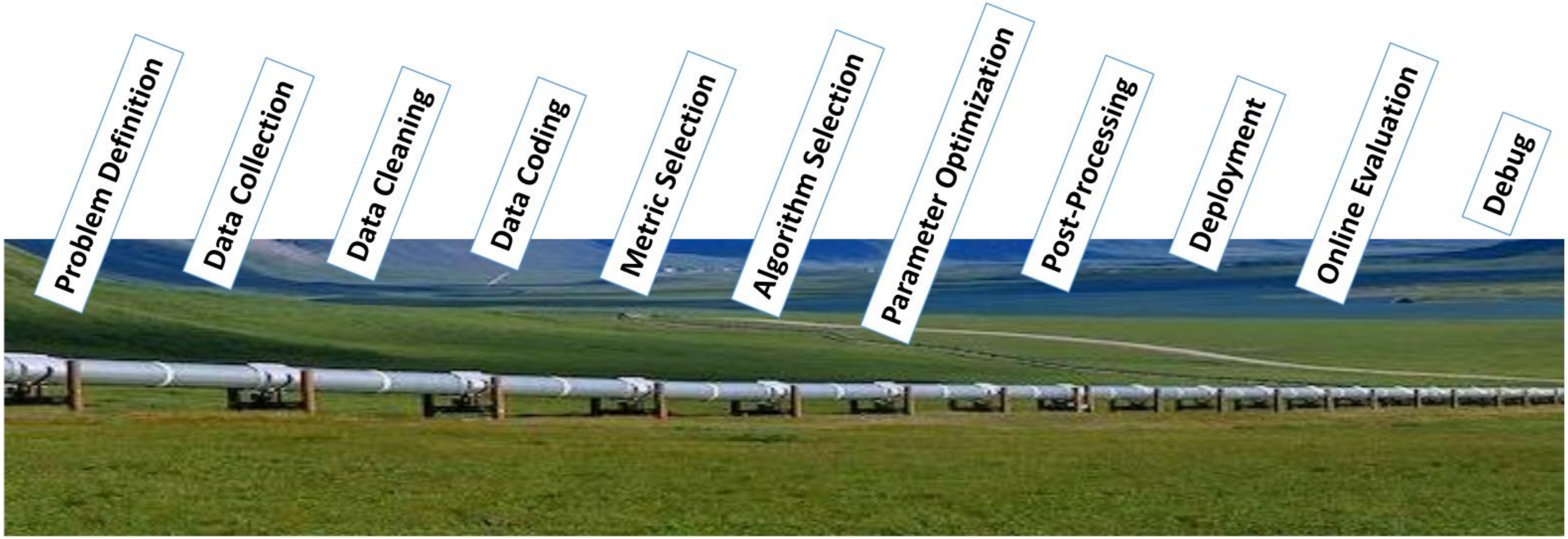


Image credit: Rich Caruana, AutoML 2015

Example I – Data cleaning and ingestion

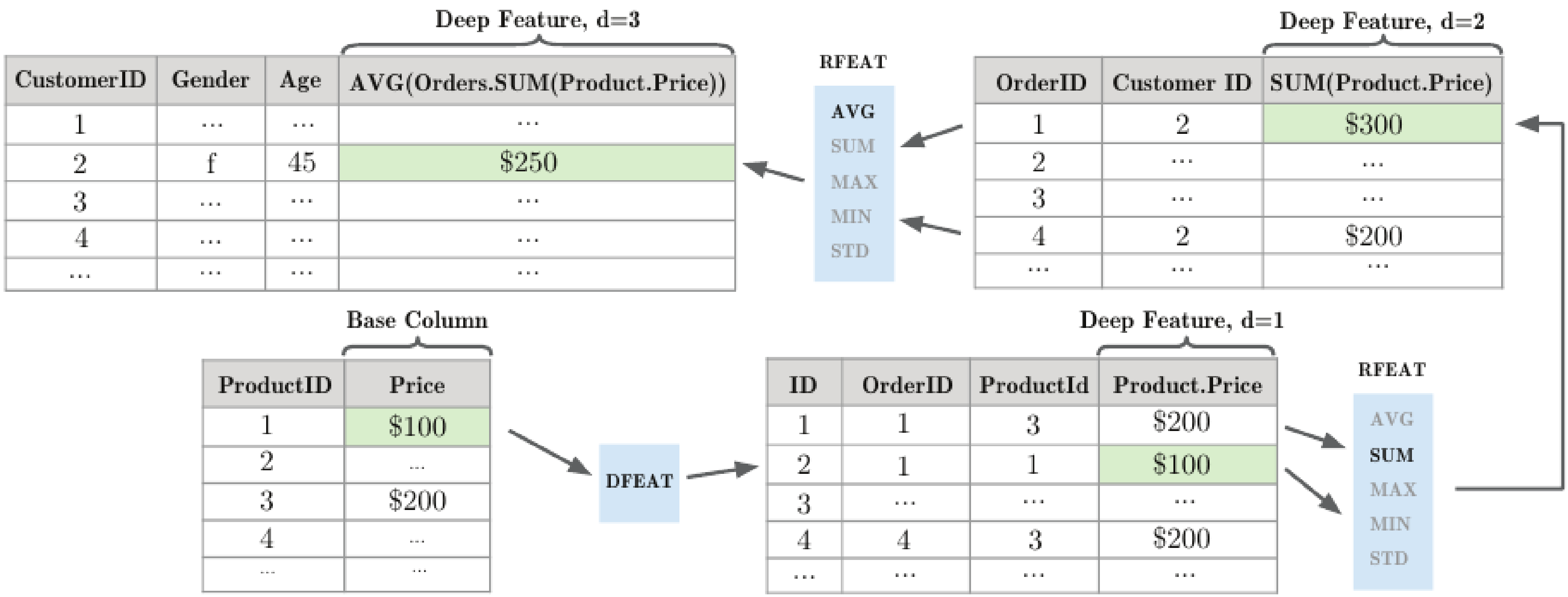
- Automatically detect the dialect of CSV files
  [van den Burg et al., arXiv:1811.11242]
- Automatically classify data types
  [Valera and Ghahramani, ICML 2017]
- Automatically detect mistakes in the data gathering process
  [Sutton et al., KDD 2018]
- Check out the talk of Charles Sutton@AutoML Workshop 2019

Example II – Feature Engineering

- From relational data bases:
  - Automatically aggregates information, can for example generate the *average sum of orders*
  - Requires post-hoc pruning of the features
  - [Kanter and Veeramachaneni, DSAA 2015]

Example II – Feature Engineering

- From relational data bases:
  - Automatically aggregates information, can for example generate the *average sum of orders*
  - Requires post-hoc pruning of the features
  - [Kanter and Veeramachaneni, DSAA 2015]



CustomerID	Gender	Age	AVG(Orders.SUM(Product.Price))
1	...	...	...
2	f	45	$250
3	...	...	...
4	...	...	...
...	...	...	...

OrderID	Customer ID	SUM(Product.Price)
1	2	$300
2	...	...
3	...	...
4	2	$200
...	...	...

ProductID	Price
1	$100
2	...
3	$200
4	...
...	...

ID	OrderID	ProductId	Product.Price
1	1	3	$200
2	1	1	$100
3	...	...	...
4	4	3	$200
...	...	...	...

Example II – Feature Engineering

- From relational data bases:
  - Automatically aggregates information, can for example generate the *average sum of orders*
  - Requires post-hoc pruning of the features
  - [Kanter and Veeramachaneni, DSAA 2015]
- From featurized data:
  - Genarate candidate features by applying
    - unary (normalization, discretization, sqrt, square, log etc.)
    - binary (+,-,*,/)
    - higher order (GroupByThen)
  - Use search mechanism to perform guided exploration
  - Use feature selection to remove unnecessray features again
  - [Smith and Bull, GP&EM 2005, Katz et al., ICDM 2016]

Example III: Off-the-shelf Algorithms

- Reduce the amount of tuning:
  - Random Forests are excellent default classifiers
  - Learning rate adaption
    - rProp
    - RMSProp
    - ...
    - Adam
    - ...
    - Ranger (look ahead + rectified Adam)
  - Pre-trained Neural Networks
  - Better defaults
  - ...

Outline

Part 1: General AutoML

1. AutoML by Hyperparameter Optimization
2. Black-box Hyperparameter Optimization
3. Beyond black-box optimization
4. Meta-learning
5. Examples of AutoML
6. Open issues and future work
7. Wrap-up & Conclusion

Part 2: Neural Architecture Search & Meta-Learning

Access to real-world large-scale datasets

While a commonly cited reason for the pressing need for effective and efficient data mining algorithms is the growing number of huge databases, the data mining research community almost never gets to see those databases. ***Most databases available for empirical studies are ridiculously small****. Unless a number of realistic and big databases become publically available, the only way to fill the gap seems to be the use of artificially generated databases.*

Johann Petrak, 2000

Access to real-world large-scale datasets

- The current state:

Access to real-world large-scale datasets

- The current state:
  - Many image datasets available -> good for NAS

Access to real-world large-scale datasets

- The current state:
  - Many image datasets available -> good for NAS
  - UCI is not structured/machine readable -> bad for general AutoML

Access to real-world large-scale datasets

- The current state:
  - Many image datasets available -> good for NAS
  - UCI is not structured/machine readable -> bad for general AutoML
- What do we need:
  - Large real-world datasets
  - Many of them
  - Machine readable description

Access to real-world large-scale datasets

- The current state:
  - Many image datasets available -> good for NAS
  - UCI is not structured/machine readable -> bad for general AutoML
- What do we need:
  - Large real-world datasets
  - Many of them
  - Machine readable description
- Call for contribution:
  - If you have a paper which introduces a new dataset
  - or if you have a paper which uses large datasets
  - or if you have large datasets at hand

  ⇨ upload them to OpenML.org

OpenML.org

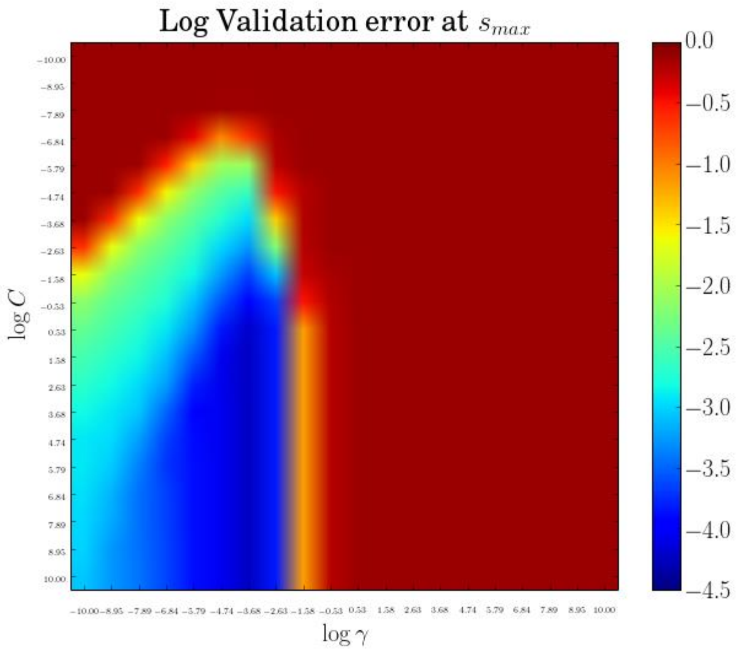
- Collaborative machine learning
- Share:
  - Datasets
  - Tasks
  - Runs
- Download:
  - >20.000 datasets
  - >90.000 tasks
  - >9.985.000 runs
- APIs in Python, R and Java
- Learn more on OpenML.org & get involved today!

[Vanschoren et al., SIGKDD 2014]

Search space representation

Bounded representation

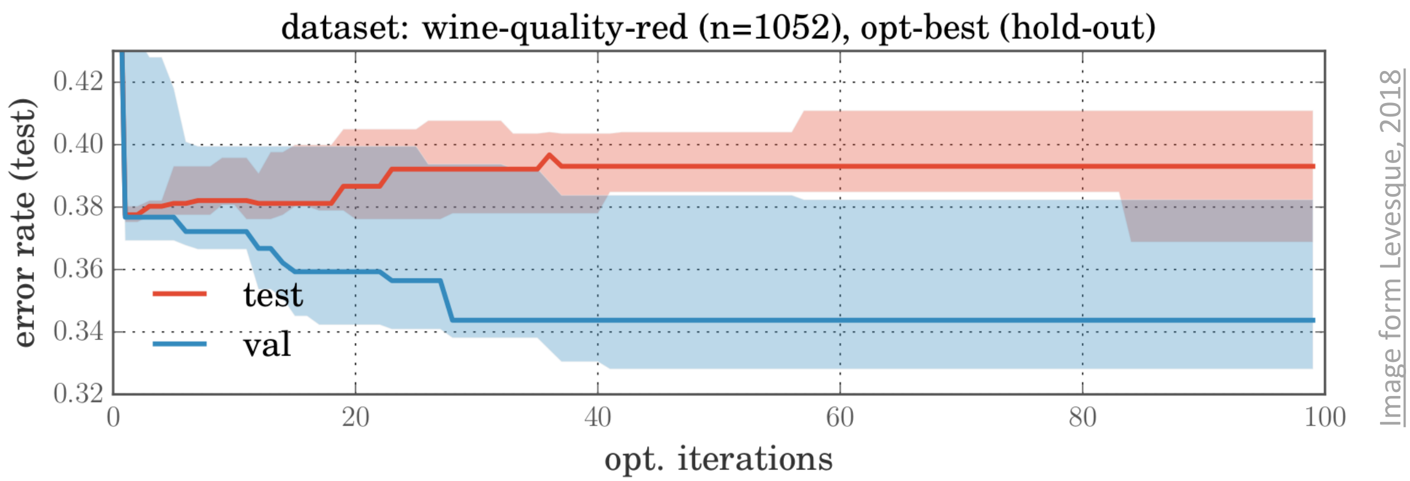
1. Creation of bounds still requires expert knowledge
2. Dynamic extension possible, but not widely used [Bergstra et al., NeurIPS 2011, Shahriari et al., AISTATS 2015]
3. AutoML tools ship with search spaces
4. If you release an algorithm, also release the search space and make magic constants tunable, too [Hoos, 2012]



Pipeline construction?

- See https://www.slideshare.net/JoaquinVanschoren/automl-lectures-acdl-2019

Overfitting



- Overfitting on HPO level possible
- Alleviate by:
  - More data [Levesque, 2018]
  - Reshuffle the train-valid split each iteration [Levesque, 2018]
  - Separate selection split [Zeng and Luo, Hiss 2017, Mohr et al., ML 2018, Levesque, 2018]
  - Stable optima [Nguyen et al., PAKDD 2017]
  - Ensembling [Momma and Bennett, 2002; Escalante et al., 2009; Bürger and Pauli, 2015; Feurer et al., 2015]

What can be automated?

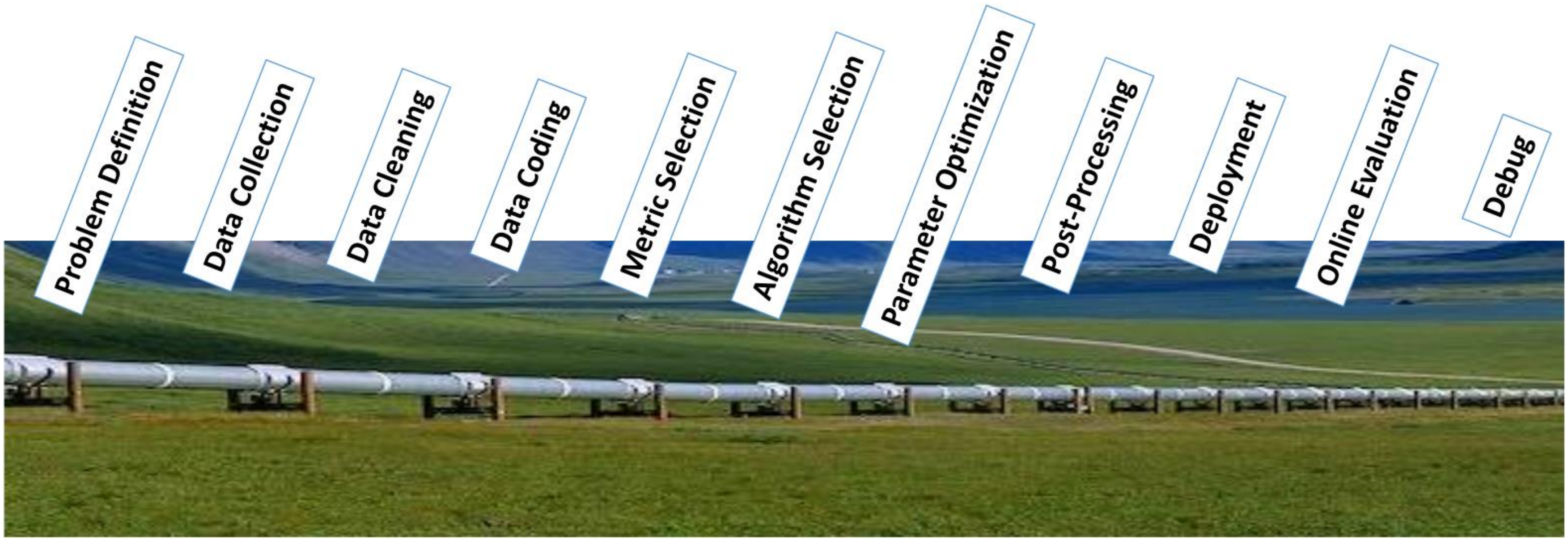


Image credit: Rich Caruana, AutoML 2015

What can be automated?

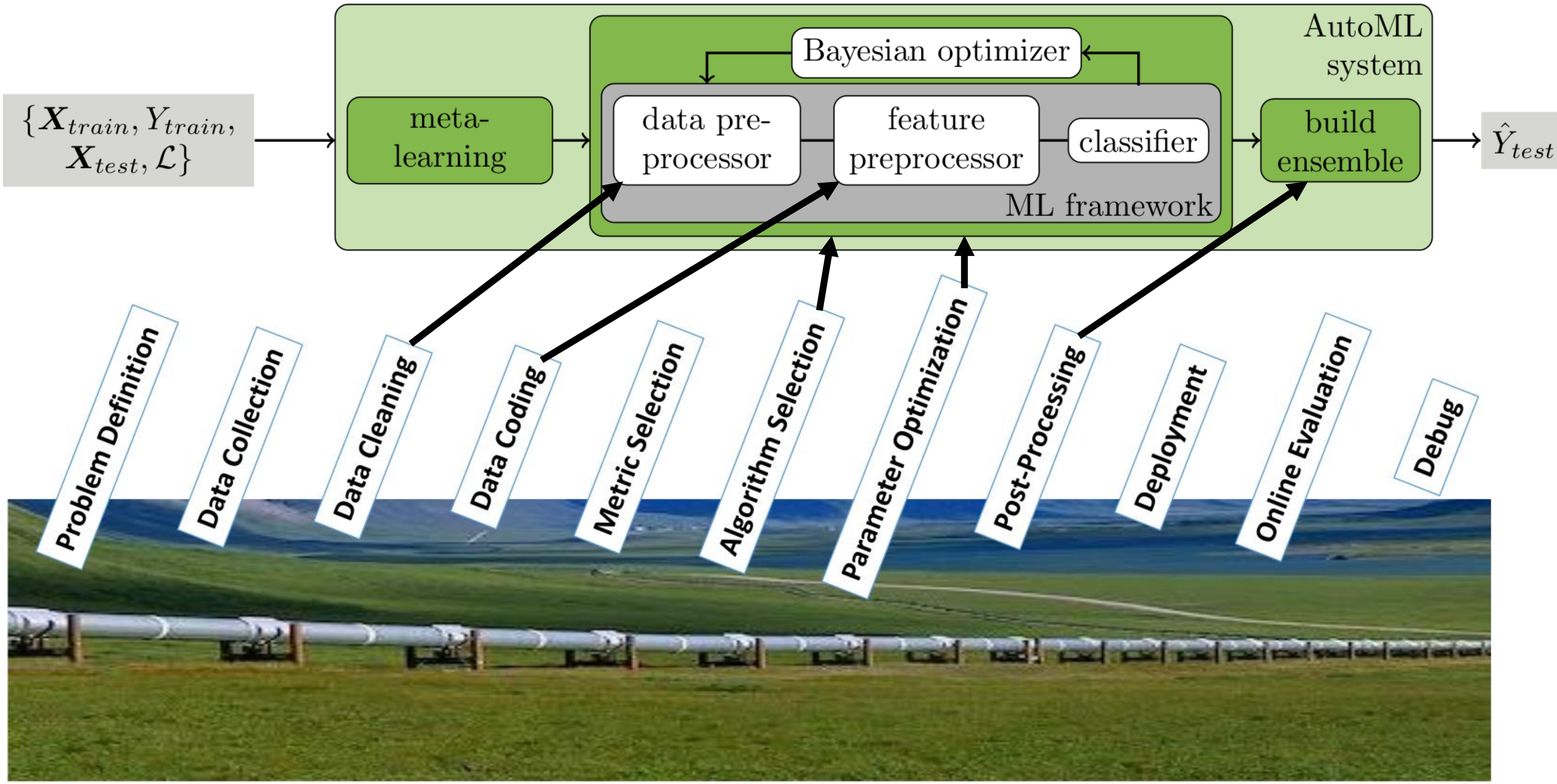


Image credit: Rich Caruana, AutoML 2015

What can be automated?

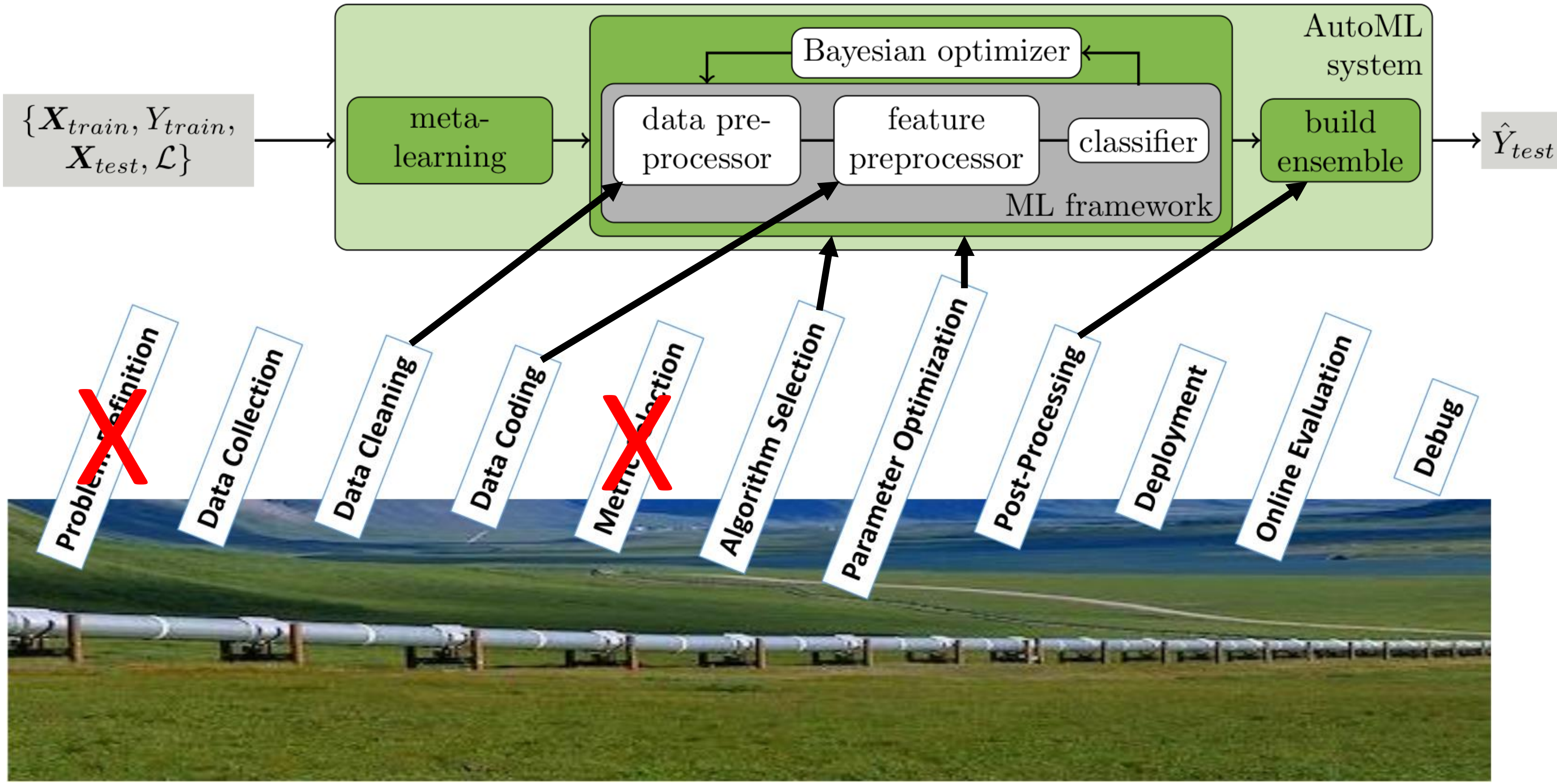


Image credit: Rich Caruana, AutoML 2015

Outline

Part 1: General AutoML

1. AutoML by Hyperparameter Optimization
2. Black-box Hyperparameter Optimization
3. Beyond black-box optimization
4. Meta-learning
5. Examples of AutoML
6. Open issues and future work
7. Wrap-up & Conclusion

Part 2: Neural Architecture Search & Meta-Learning

HPO for Practitioners: Which Tool to Use?

If you have access to multiple fidelities

- We recommend BOHB [Falkner et al, ICML 2018]
- https://github.com/automl/HpBandSter
- Combines the advantages of Bayesian optimization and Hyperband

If you do not have access to multiple fidelities

- Low-dim. continuous: GP-based BO (e.g., BoTorch, MLRMBO, Sigopt, GP version of SMACv3)
- High-dim, categorical, conditional: SMAC or Hyperopt
- Purely continuous, budget >10x dimensionality: CMA-ES

Open-source AutoML Tools based on HPO

- Auto-WEKA [Thornton et al, KDD 2013]
  - 768 hyperparameters, 4 levels of conditionality
  - Based on WEKA and SMAC
- Hyperopt-sklearn [Komer et al, SciPy 2014]
  - Based on scikit-learn & TPE
- Auto-sklearn [Feurer al, NeurIPS 2015]
  - Based on scikit-learn & SMAC
  - Uses meta-learning and posthoc ensembling
  - Won AutoML competitions 2015-2016 & 2017-2018
- H2O AutoML [no reference]
  - Uses implementations from H2O.ai
  - Based on random search and stacking
- TPOT [Olson et al, EvoApplications 2016]
  - Based on scikit-learn and evolutionary algorithms
- ML-PLAN [Mohr et al., Machine Learning 2018]
  - Based on WEKA and Hierarchical Task Networks

AutoML: Democratization of Machine Learning

Auto-sklearn also won the last two phases of the AutoML challenge human track (!)

- It performed better than up to 130 teams of human experts
- It is open-source (BSD) and trivial to use:

AutoML: Democratization of Machine Learning

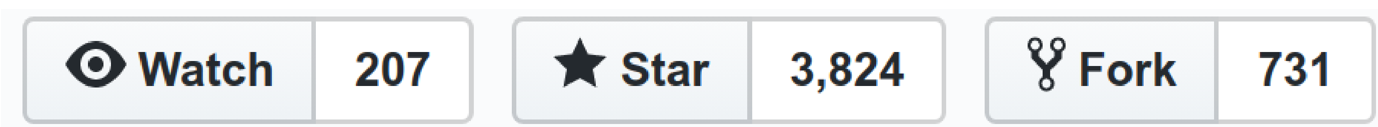
Fork me on GitHub

Auto-sklearn also won the last two phases
of the AutoML challenge human track (!)

- It performed better than up to 130 teams of human experts
- It is open-source (BSD) and trivial to use:

```
import autosklearn.classification as cls
automl = cls.AutoSklearnClassifier()
automl.fit(X_train, y_train)
y_hat = automl.predict(X_test)
```

automl.github.io/auto-sklearn



What have we learned?

1. AutoML by Hyperparameter Optimization

   AutoML can be phrased as an HPO problem

2. Black-box Hyperparameter Optimization

   We reviewed Bayesian optimization

3. Beyond black-box optimization

   Practically applicable by using domain knowledge

4. Meta-learning

   Increase practicality by using previous data

5. Examples

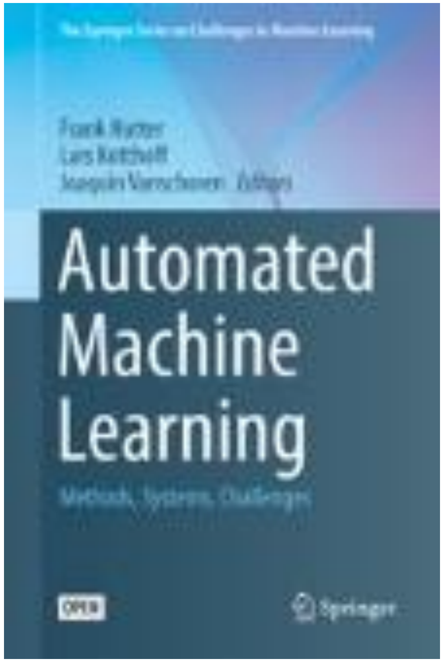
   AutoML can be used in almost every step of the ML pipeline

6. Open issues and future work

   Datasets, search space representation & overfitting

Further reading

- Automated Machine Learning: Methods, Systems, Challenges
  - Edited by Frank Hutter, Lars Kotthoff and Joaquin Vanschoren
  - Contains introductions to HPO, Meta-Learning and NAS
  - https://www.springer.com/de/book/9783030053178
- Various literature reviews on arXiv:
  - 1908.05557: Focus on open source software
  - 1810.13306: General and comprehensive
  - 1908.00709: Focuses mostly on NAS
  - 1905.01392: NAS survey
- AutoML workshop video recordings
  - icml2019.automl.org



The end

Thank you for your attention!

Special thanks to Frank Hutter and Joaquin Vanschoren for providing me with the slides this presentation is based on.

Contact:

feurerm@cs.uni-freiburg.de

@__mfeurer__ @automlfreiburg