Bayesian Optimization and Meta-Learning

Frank Hutter

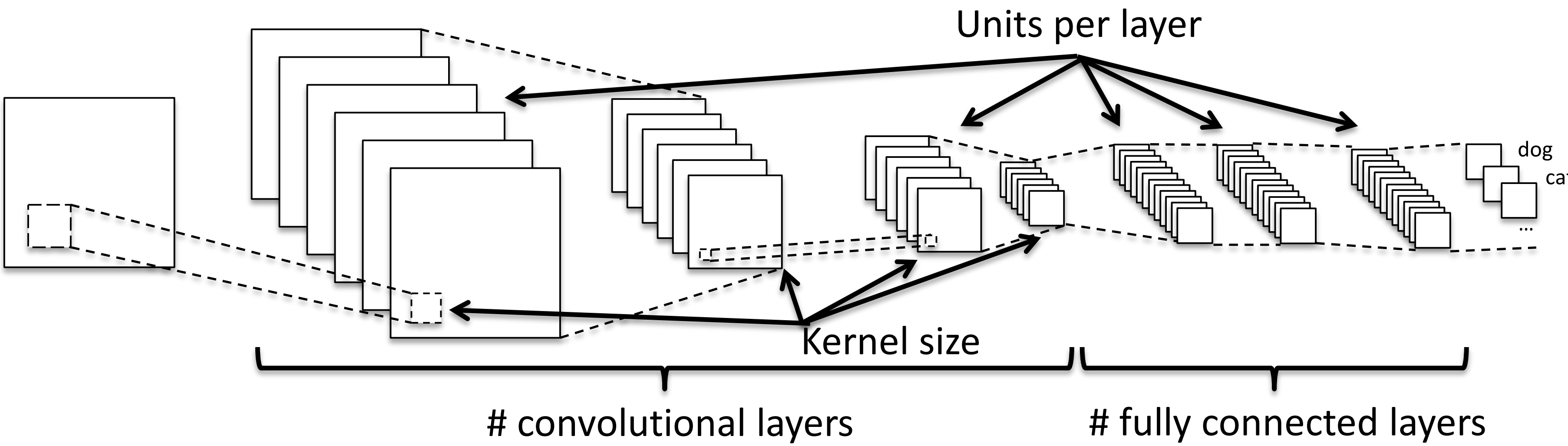
University of Freiburg &
Bosch Center for Artificial Intelligence
fh@cs.uni-freiburg.de

Slides available at http://bit.ly/metalearn;
all references are clickable hyperlinks

One Problem of Deep Learning

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

– Architectural hyperparameters



– **Optimization**: algorithm, learning rate initialization & schedule, momentum, batch sizes, batch normalization, …

– **Regularization**: dropout rates, weight decay, data augment., …

→ **Easily 20-50 design decisions**

Optimizing Hyperparameters Matters a Lot

Improvements in the state of the art in many applications:

- Auto-Augment [Cubuk et al, arXiv 2018]
  – Search space: Combinations of translation, rotation & shearing

Dataset	GPU hours	Best published results	Our results
CIFAR-10	5000	2.1	1.5
CIFAR-100	0	12.2	10.7
SVHN	1000	1.3	1.0
Stanford Cars	0	5.9	5.2
ImageNet	15000	3.9	3.5

- AlphaGO [Chen et al, 2018]
  – E.g., prior to the match with Lee Sedol: tuning increased win-rate from 50% to 66.5% in self-play games
  – Tuned version was deployed in the final match; many previous improvements during development based on tuning
- Neural language models [Melis et al, ICLR 2018]
- Deep RL is very sensitive to hyperparams [Henderson et al, AAAI 18]

Outline

1. Blackbox Bayesian Hyperparameter Optimization
2. Beyond Blackbox: Speeding up Bayesian Optimization
3. Hyperparameter Importance Analysis
4. Case Studies

Based on: Feuerer & Hutter: [Chapter 1 of the AutoML book: Hyperparameter Optimization]

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

- Continuous
  - Example: learning rate
- Integer
  - Example: #filters
- Categorical
  - Finite domain, unordered
    - Example 1: activation function ∈ {ReLU, Leaky ReLU, tanh}
    - Example 2: operator ∈ {conv3x3, separable conv3x3, max pool, …}
  - Special case: binary

Conditional hyperparameters

- Conditional hyperparameters B are only active if their “parent“ hyperparameters A are set a certain way

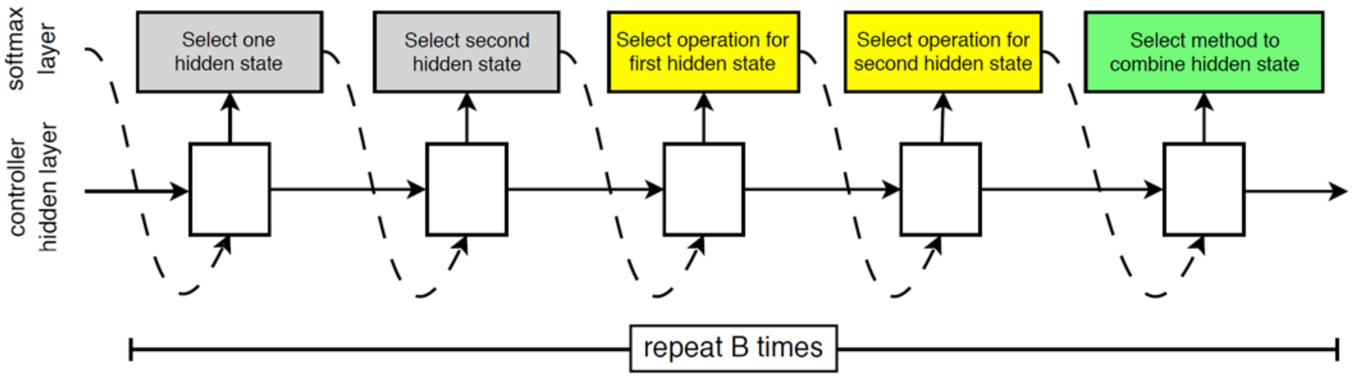
  – Example 1:
    - A = choice of optimizer (Adam or SGD)
    - B = Adam‘s second momentum hyperparameter
      – only active if A=Adam

  – Example 2:
    - A = # layers
    - B = operation in layer k (conv3x3, separable conv3x3, max pool, ...)
      – only active if A ≥ k

NAS as Hyperparameter Optimization

- We can rewrite many NAS problems as HPO problems

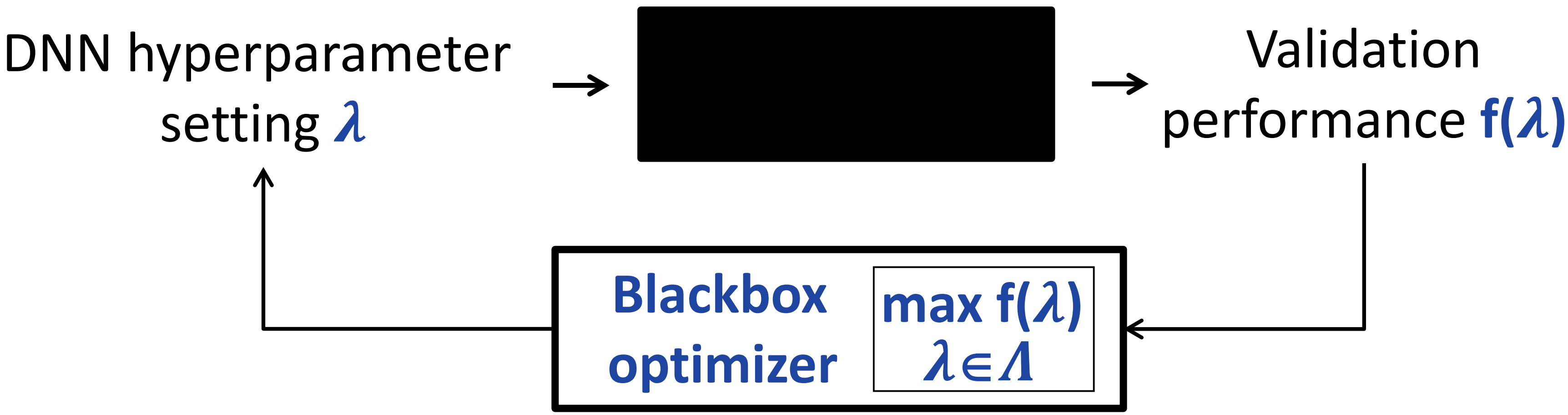
- E.g., cell search space by Zoph et al [CVPR 2018]



  – 5 categorical choices for Nth block:
    - 2 categorical choices of hidden states, each with domain {0, ..., N-1}
    - 2 categorical choices of operations
    - 1 categorical choice of combination method

    → Total number of hyperparameters for the cell: 5B (with B=5 by default)

Blackbox Hyperparameter Optimization

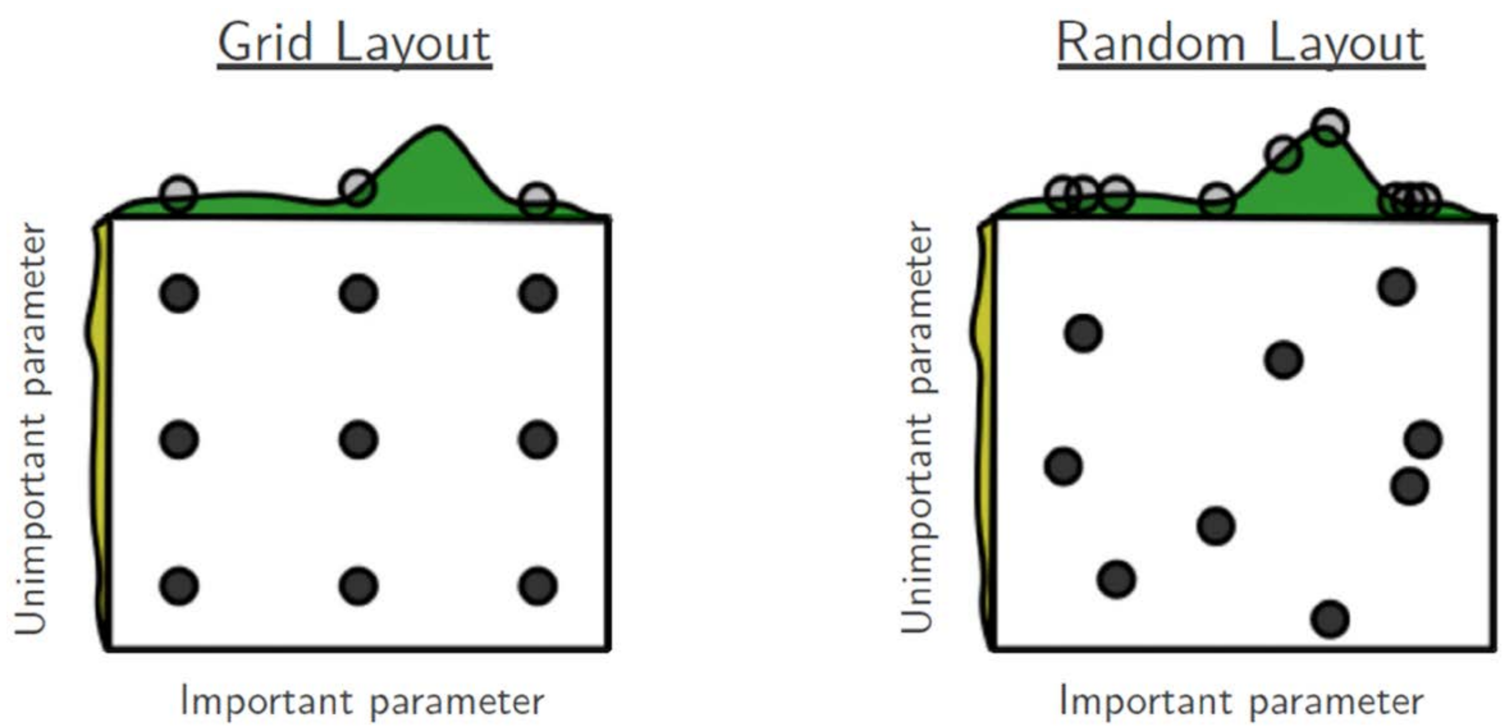


- The blackbox function is expensive to evaluate
  $\rightarrow$ sample efficiency is important

Grid Search and Random Search

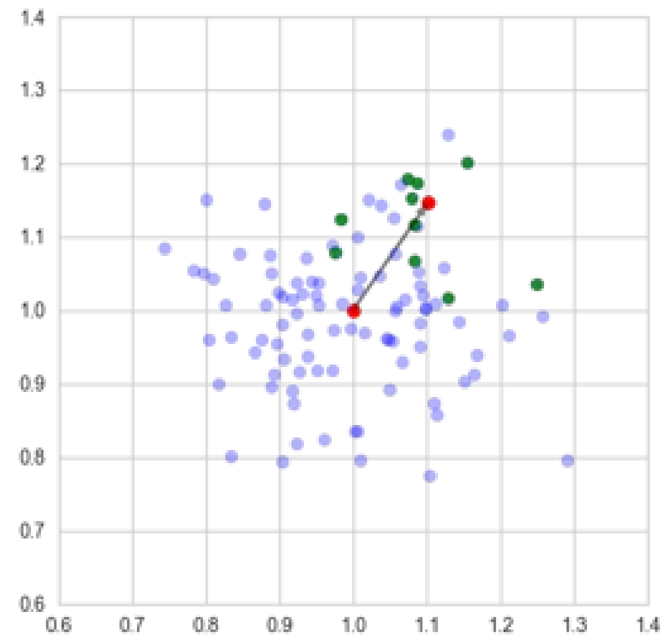
[Bergstra & Bengio, JMLR 2012]

- Both completely uninformed
- Random search handles unimportant dimensions better
- Random search is a useful baseline



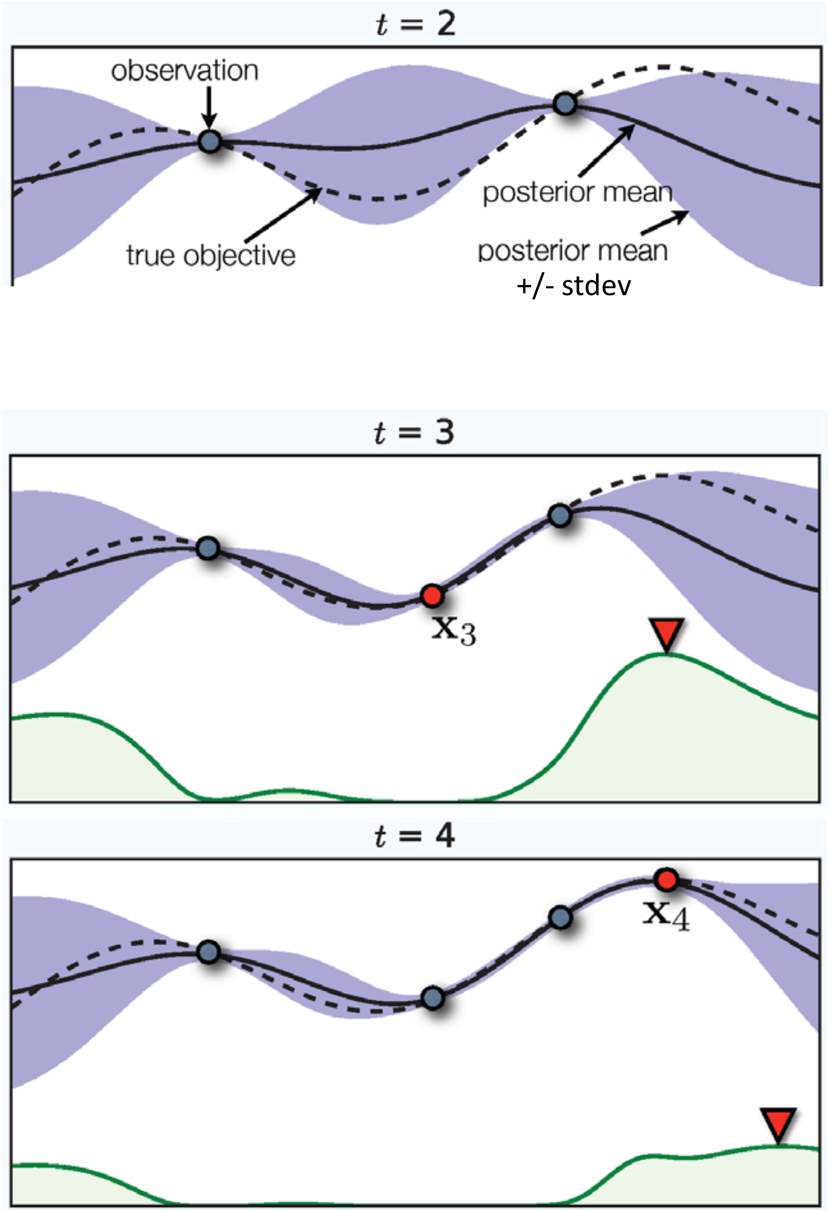
Population-based Methods

- Population of configurations
  - Maintain diversity
  - Improve fitness of population
- E.g, evolutionary strategies
  - Book: [Beyer & Schwefel [2002]]
  - Popular variant: CMA-ES [Hansen, 2016]
    - Very competitive for HPO of deep neural nets [Loshchilov & H., 2016]
    - Embarassingly parallel
    - Purely continuous



Bayesian Optimization

- Approach
  - Fit a proabilistic model to the function evaluations $\langle \lambda, f(\lambda) \rangle$
  - Use that model to trade off exploration vs. exploitation



- 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]

Image source: Brochu et al [arXiv, 2010]

Challenges in Bayesian Hyperparameter Optimization

- Problems for standard Gaussian Process (GP) approach:
  - Complex hyperparameter space
    - High-dimensional (low effective dimensionality) [Wang et al, 2013]
    - Mixed continuous/discrete hyperparameters [H. et al, 2011]
    - Conditional hyperparameters [Swersky et al, 2013; Levesque et al, 2017]
  - Non-standard noise
    - Non-Gaussian [Williams et al, 2000; Shah et al, 2018; Martinez-Cantinet al, 2018]
    - Sometimes heteroscedastic [Le et al, 2005; Wang & Neal, 2012]
  - Robustness of the model [Malkomes and Garnett, 2018]
  - Model overhead [Quiñonero-Candela & Rasmussen, 2005; Bui et al, 2018; H. et al, 2010]

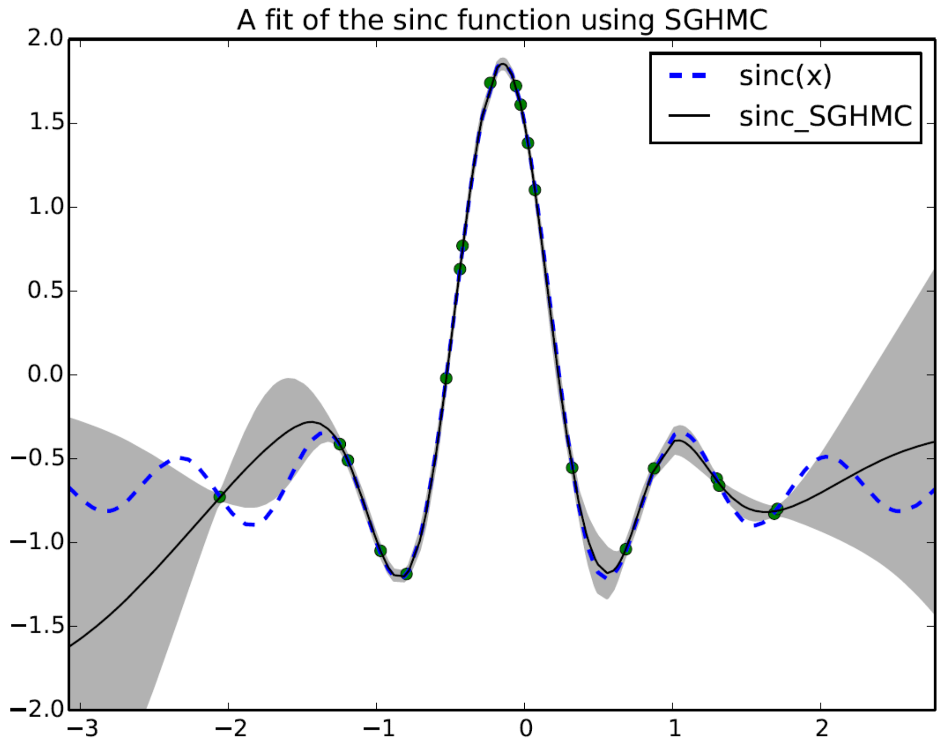
- Simple solution used in SMAC: random forests [Breiman, 2001]
  - Frequentist uncertainty estimate:
    variance across individual trees' predictions [H. et al, 2011]

Bayesian Optimization with Neural Networks

- 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]

- Strong performance on low-dimensional continuous tasks

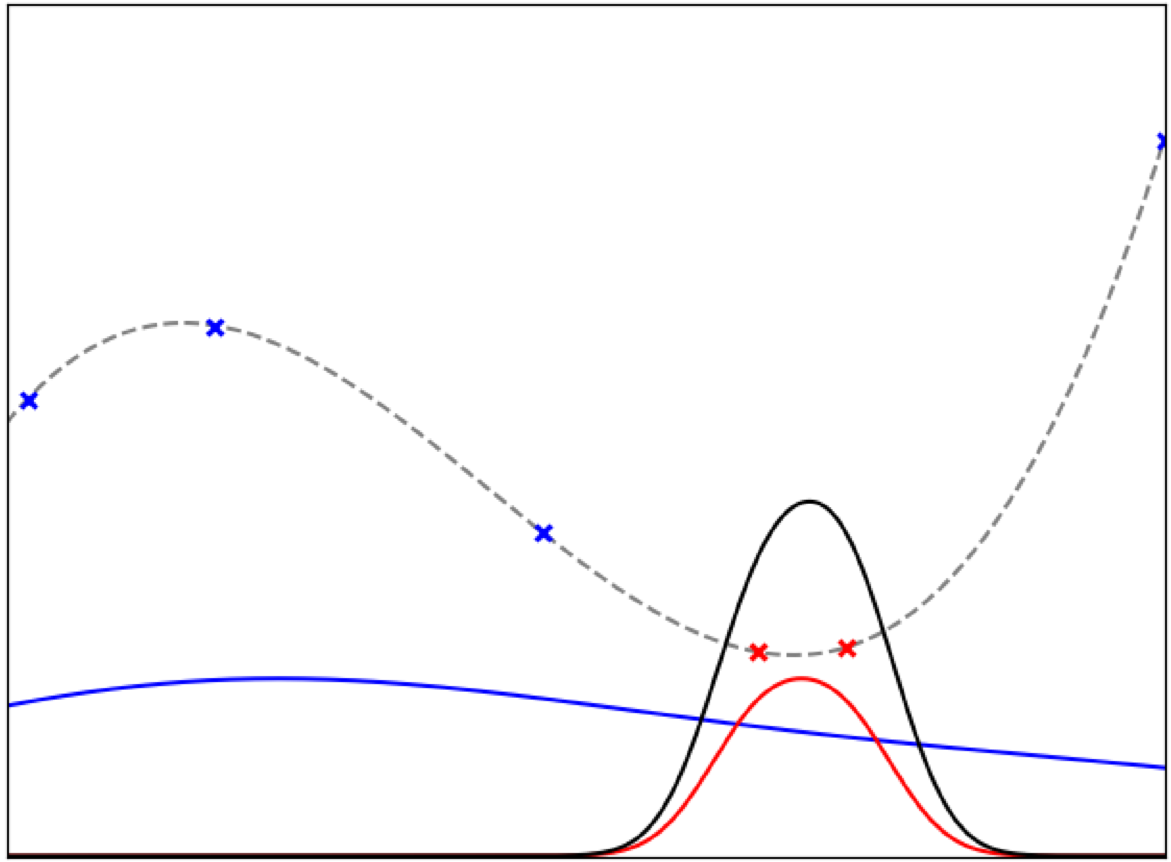
- So far not studied for:
  - High dimensionality
  - Discrete & conditional hyperparameters



Tree of Parzen Estimators (TPE)

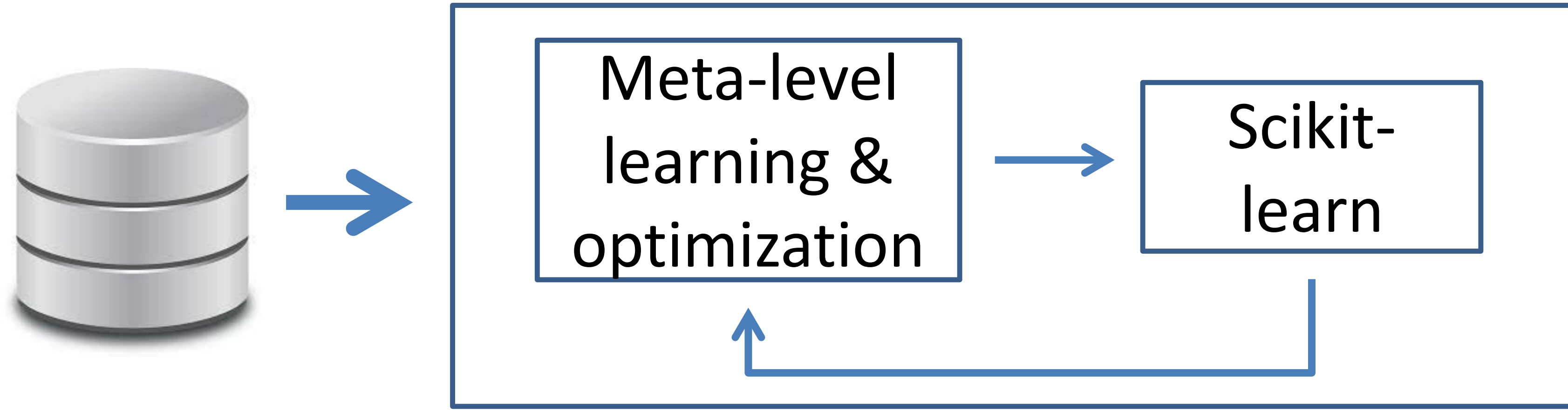
[Bergstra et al, NIPS 2011]

- Non-parametric KDEs for p(λ is good) and p(λ is bad), rather than p(y|λ)
- Equivalent to expected improvement
- Pros:
  - Efficient: O(N*d)
  - Parallelizable
  - Robust
- Cons:
  - Less sample-efficient than GPs



HPO enables AutoML Systems: e.g., Auto-sklearn

[Feurer et al, NIPS 2015]



- Optimize CV performance by SMAC

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

- Meta-learning to warmstart Bayesian optimization
  - Reasoning over different datasets
  - Dramatically speeds up the search (2 days $\rightarrow$ 1 hour)

- Automated posthoc ensemble construction to combine the models we already evaluated
  - Efficiently re-uses its data; improves robustness

HPO enables AutoML Systems: e.g., Auto-sklearn

Fork me on GitHub

- Winning approach in the AutoML challenge
  - Auto-track: overall winner, 1st place in 3 phases, 2nd in 1
  - Human track: always in top-3 vs. 150 teams of human experts
  - Final two rounds: won both tracks

https://github.com/automl/auto-sklearn

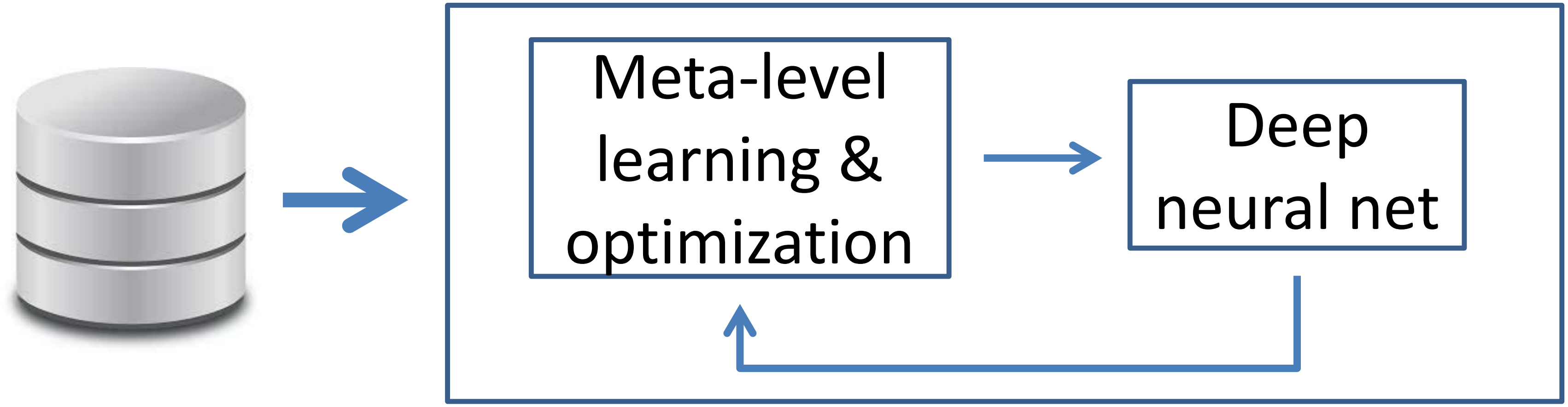
Used by 43 | Watch 196 | Star 3,477 | Fork 669

- Trivial to use, open source (BSD):

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

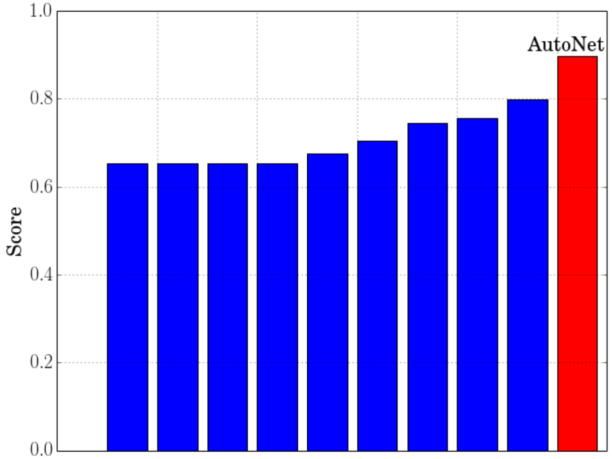
HPO enables AutoML Systems: e.g., Auto-Net

[Mendoza et al, AutoML 2016]



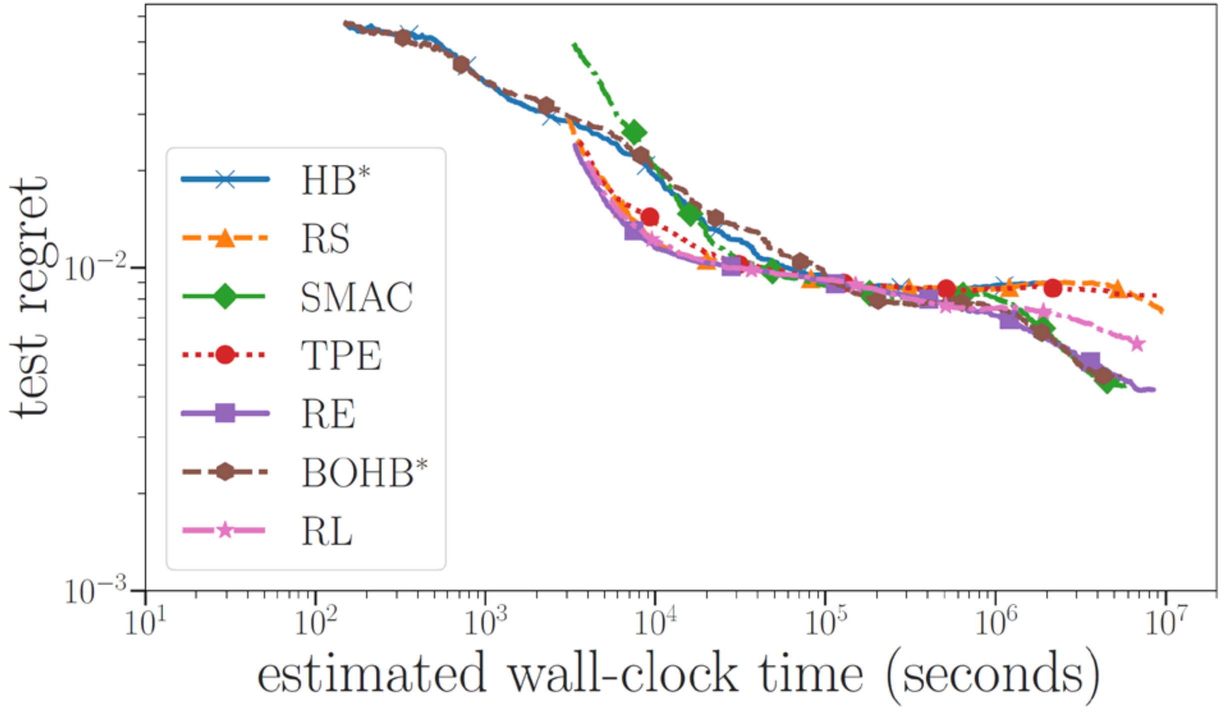
- Joint Architecture & Hyperparameter Optimization

- Auto-Net won several datasets against human experts
  – E.g., Alexis data set (2016)
    - 54491 data points,
      5000 features, 18 classes
  – First automated deep learning system to win a ML competition data set against human experts



Evaluation on NAS-Bench-101

- NAS-Bench-101 [Ying et al, ICML 2019]
  - Exhaustively evaluated, small, cell search space
  - Tabular benchmark for 423k possibilities
  - Allows for statistically sound & comparable experimentation
- Result:
  SMAC and regularized evolution outperform reinforcement learning



Other Extensions of Bayesian Optimization

- Joint optimization of a vision architecture with 238 hyperparameters with TPE [Bergstra et al, ICML 2013]

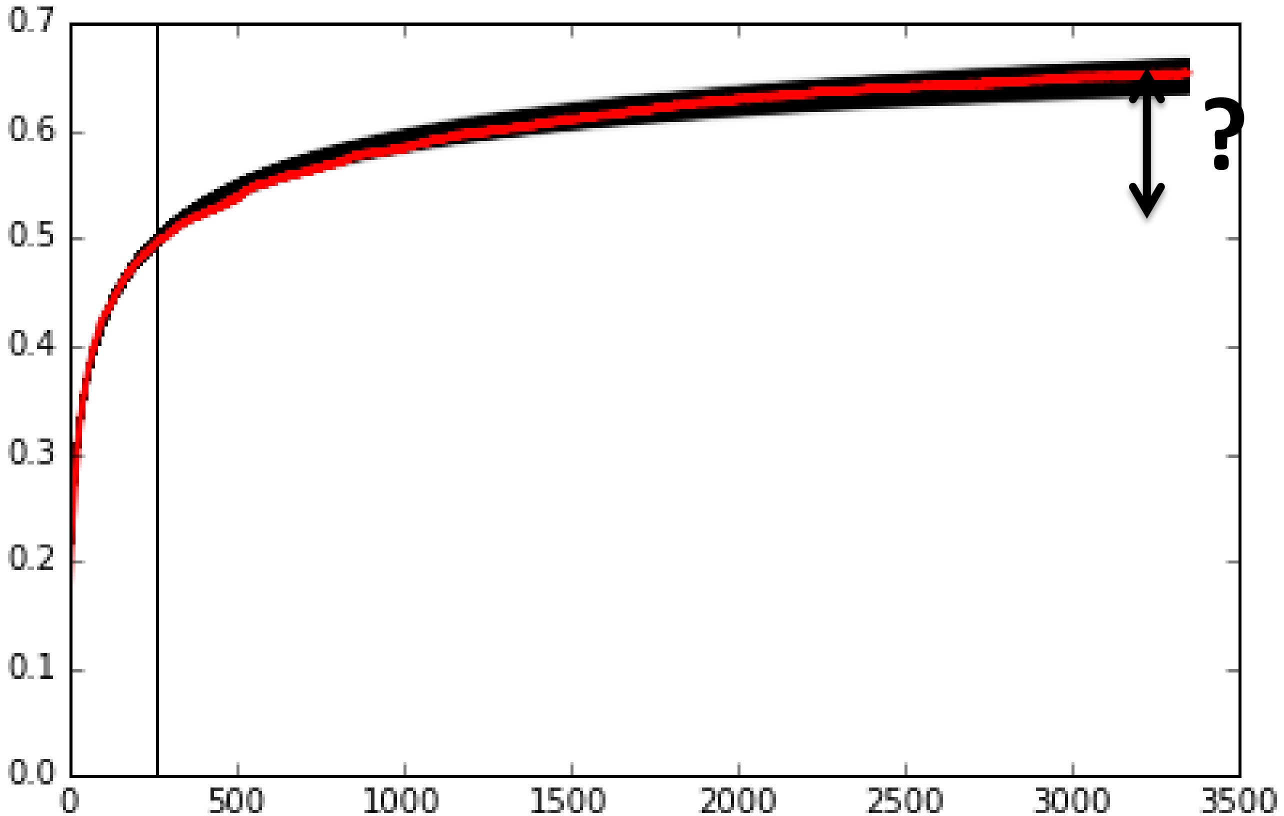
- Kernels for GP-based NAS
  - Arc kernel [Swersky et al, BayesOpt 2013]
  - NASBOT [Kandasamy et al, NIPS 2018]

- Sequential model-based optimization
  - PNAS [Liu et al, ECCV 2018]

Outline

1. Blackbox Bayesian Hyperparameter Optimization
2. Beyond Blackbox: Speeding up Bayesian Optimization
3. Hyperparameter Importance Analysis
4. Case Studies

Probabilistic Extrapolation of Learning Curves



- Parametric learning curve models [Domhan et al, IJCAI 2015]
- Gaussian process with special kernel [Swersky et al, arXiv 2014]
- Bayesian (recurrent) neural networks [Klein et al, ICLR 2017; Gargiani et al, AutoML 2019]

Learning Across Datasets

- Multitask Bayesian Optimization
  - Using Gaussian processes [Swersky et al, NIPS 2013]
  - Gaussian processes with special dataset kernel [Bardenet et al, ICML 2013; Yogatama & Mann, AISTATS 2014]
  - Using neural networks [Perrone et al, NIPS 2018]

- Warmstarting
  - Initialize with previous good models [Feurer et al, AAAI 2015]
  - Learn weights for each previous model [Feurer et al, arXiv 2018]

- Transfer acquisition functions
  - Mixture of experts approach [Wistuba et al, MLJ 2018]
  - Meta-learned acquisition function [Volpp et al, arXiv 2019]

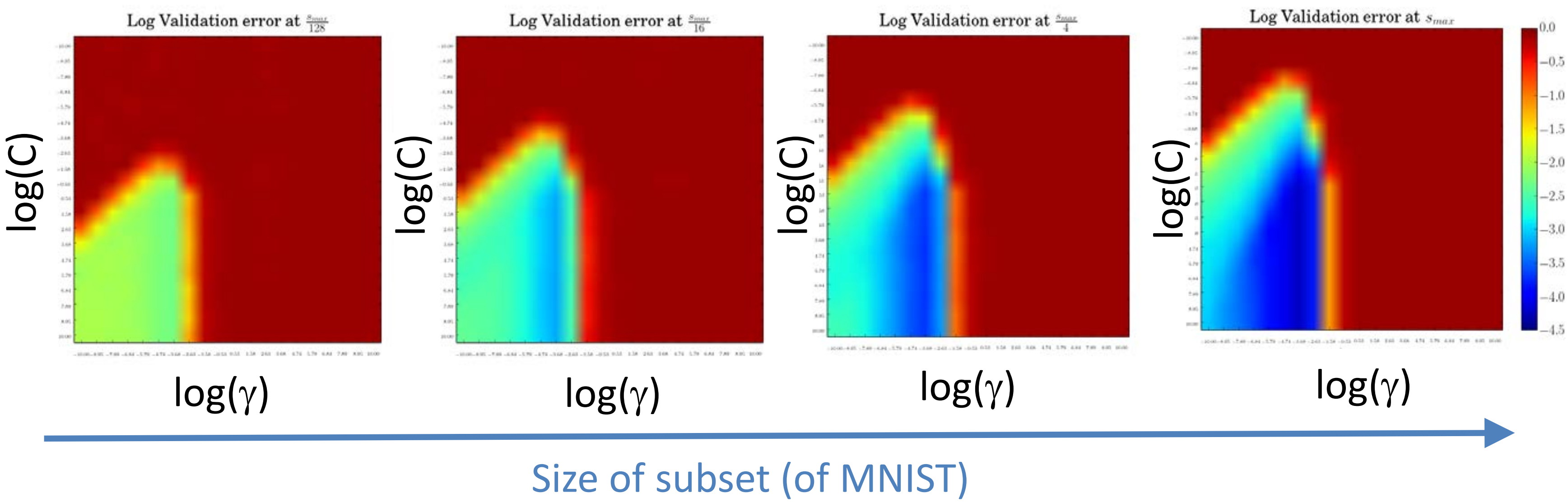
Multi-Fidelity Optimization

- 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)
  - Shorter MCMC chains in Bayesian deep learning
  - 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

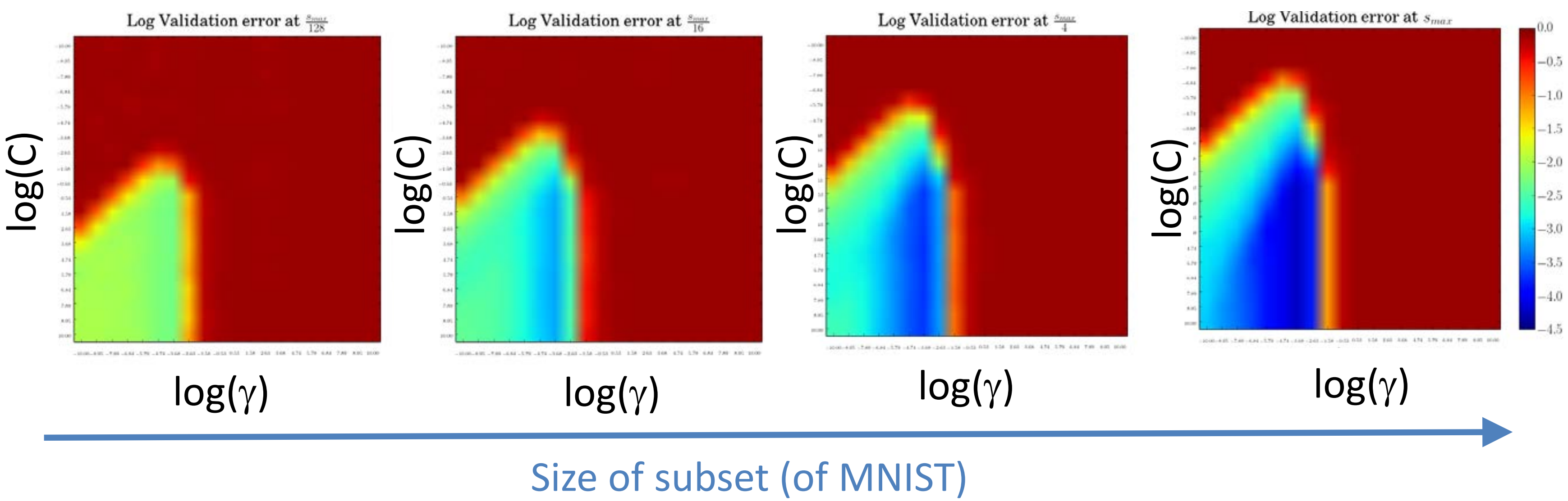
- **Make use of cheap low-fidelity evaluations**
  – E.g.: subsets of the data (here: SVM on MNIST)



  – 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

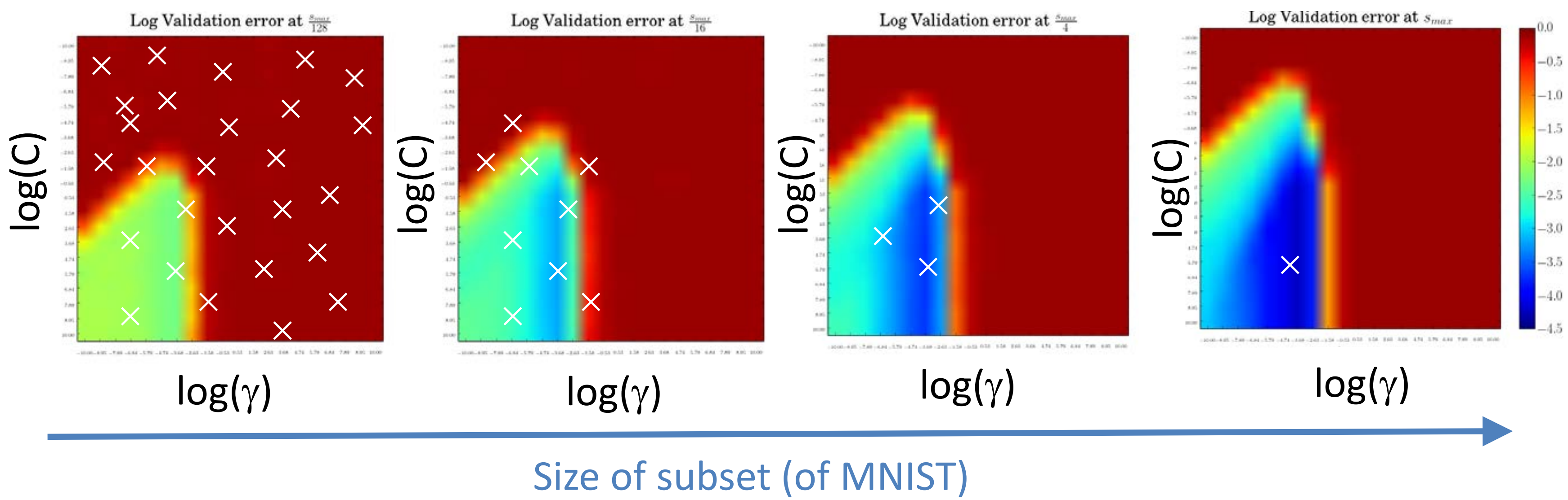
- **Make use of cheap low-fidelity evaluations**
  - E.g.: subsets of the data (here: SVM on MNIST)



  - 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, NIPS 2013; Swersky et al, arXiv 2014; Klein et al, AISTATS 2017; Kandasamy et al, ICML 2017]

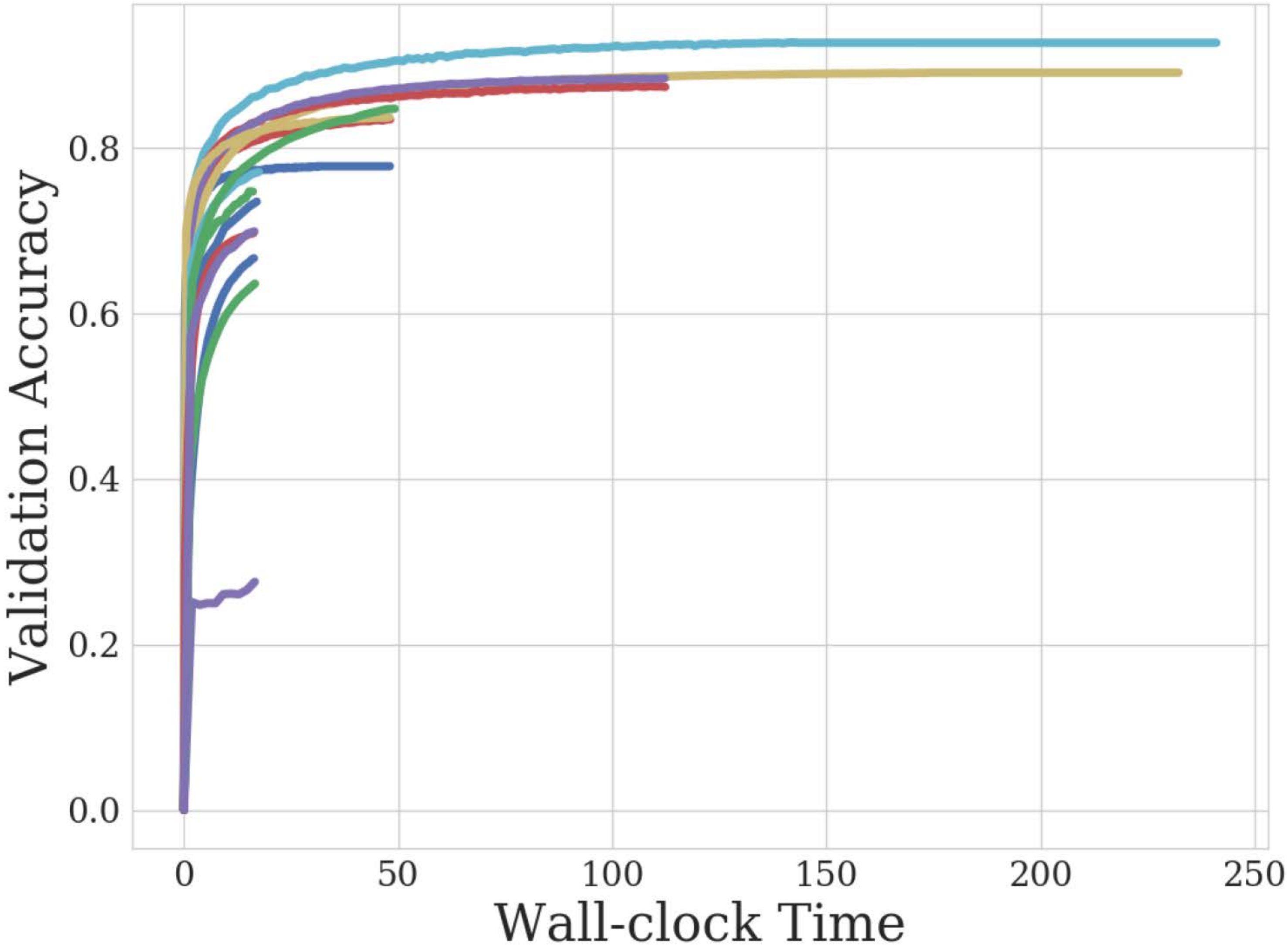
- **Make use of cheap low-fidelity evaluations**
  – E.g.: subsets of the data (here: SVM on MNIST)



  – A simpler approach: successive halving
  [Jamieson & Talwalkar, AISTATS 2016]
    - Initialize with lots of random configurations on the smallest budget
    - Top fraction survives to the next budget

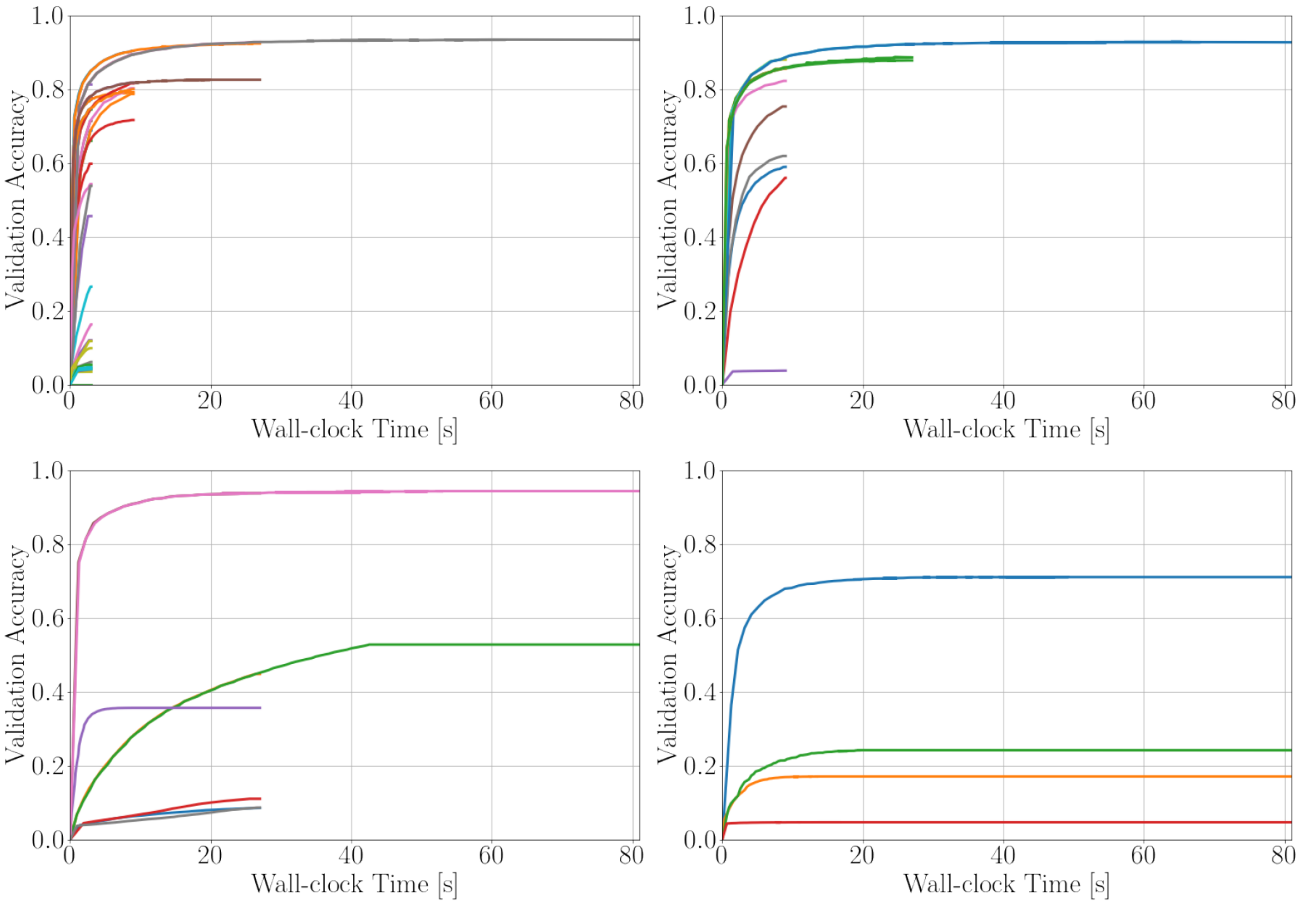
Successive Halving (SH) for Learning Curves

[Jamieson & Talwalkar, AISTATS 2016]



Hyperband (its first 4 calls to SH)

[Li et al, ICLR 2017]

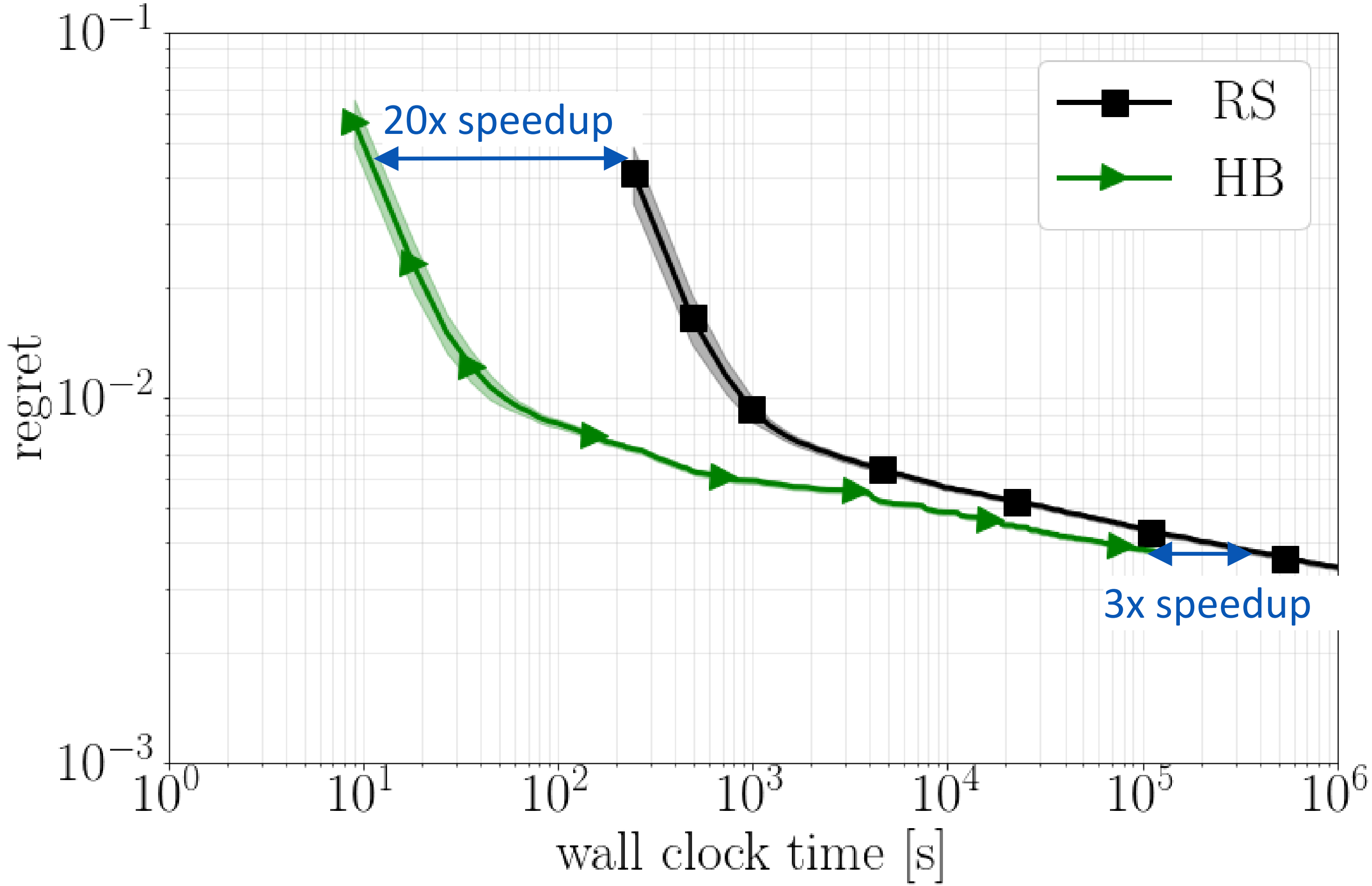


BOHB: Bayesian Optimization & Hyperband

[Falkner et al, 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 configuration to evaluate (using a TPE variant)
  - Hyperband
    - for deciding how to allocate budgets

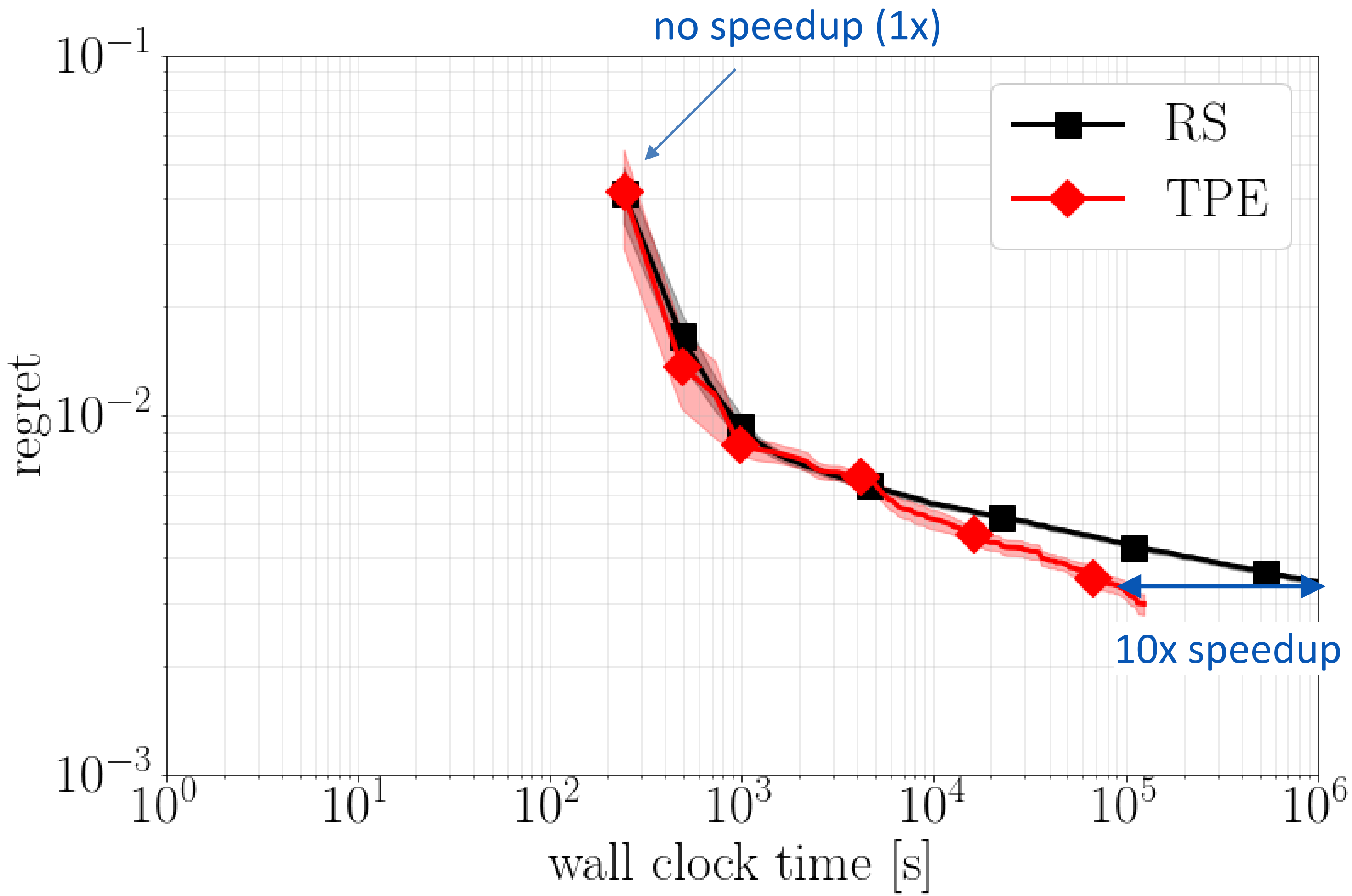
Hyperband vs. Random Search



Biggest advantage: much improved anytime performance

Auto-Net on dataset adult

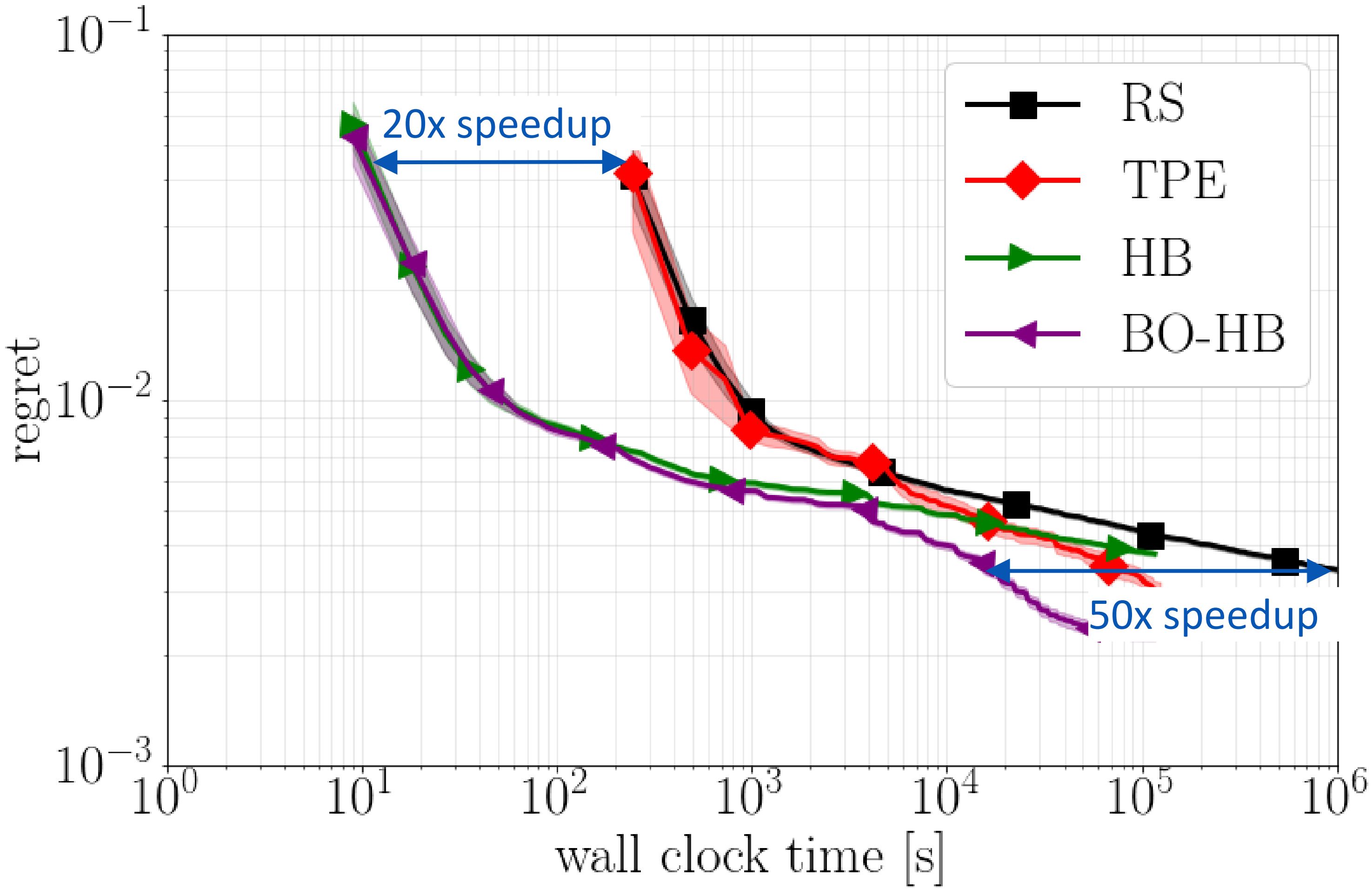
Bayesian Optimization vs Random Search



Biggest advantage: much improved **final performance**

Auto-Net on dataset adult

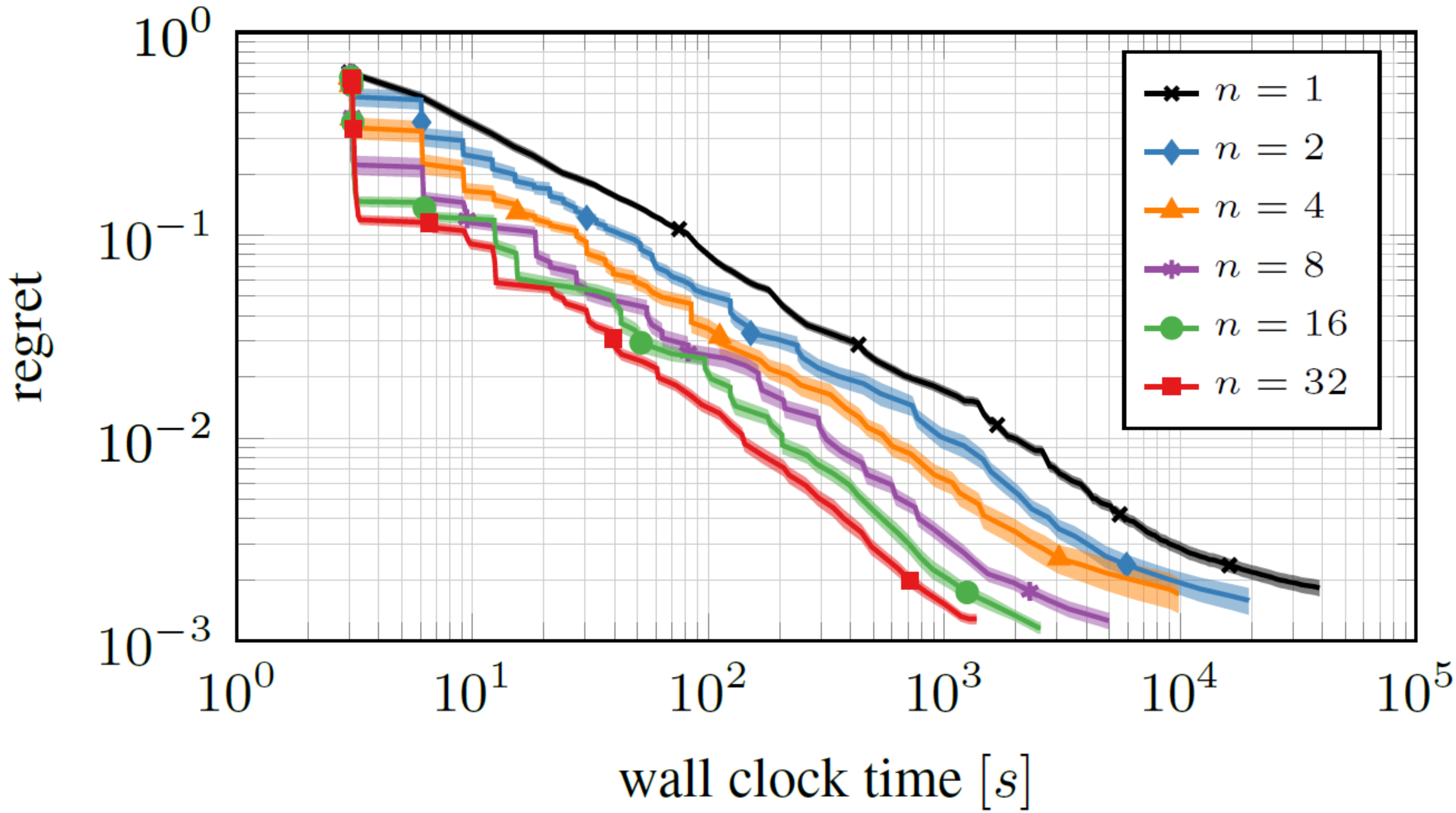
Combining Bayesian Optimization & Hyperband



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

Auto-Net on dataset adult

Almost linear speedups by parallelization



Auto-Net on dataset letter

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 TPE and Hyperband

- If you do not have access to multiple fidelities
  - Low-dim. continuous: GP-based BO (e.g., Spearmint)
  - High-dim, categorical, conditional: SMAC or TPE
  - Purely continuous, budget >10x dimensionality: CMA-ES

Outline

1. Blackbox Bayesian Hyperparameter Optimization
2. Beyond Blackbox: Speeding up Bayesian Optimization
3. Hyperparameter Importance Analysis
4. Case Studies

Hyperparameter Importance

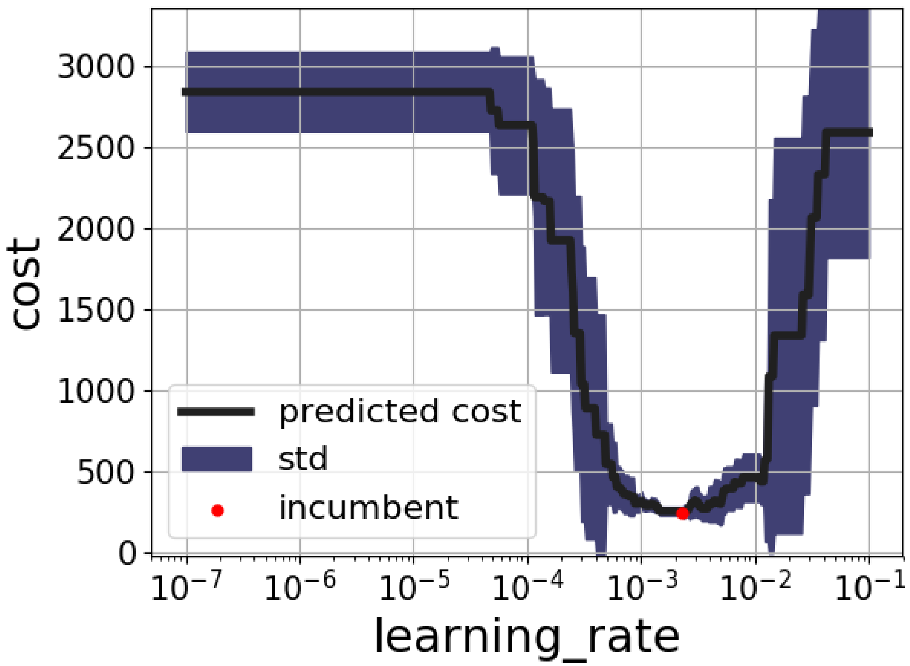
- Common question:
  Which hyperparameters actually matter?

- Hyperparameter space has low effective dimensionality
  – Only a few hyperparameters matter a lot
  – Many hyperparameters only have small effects
  – Some hyperparameters have robust defaults and never need to change

- Local importance (around best configuration) vs. global importance (on average across entire space)

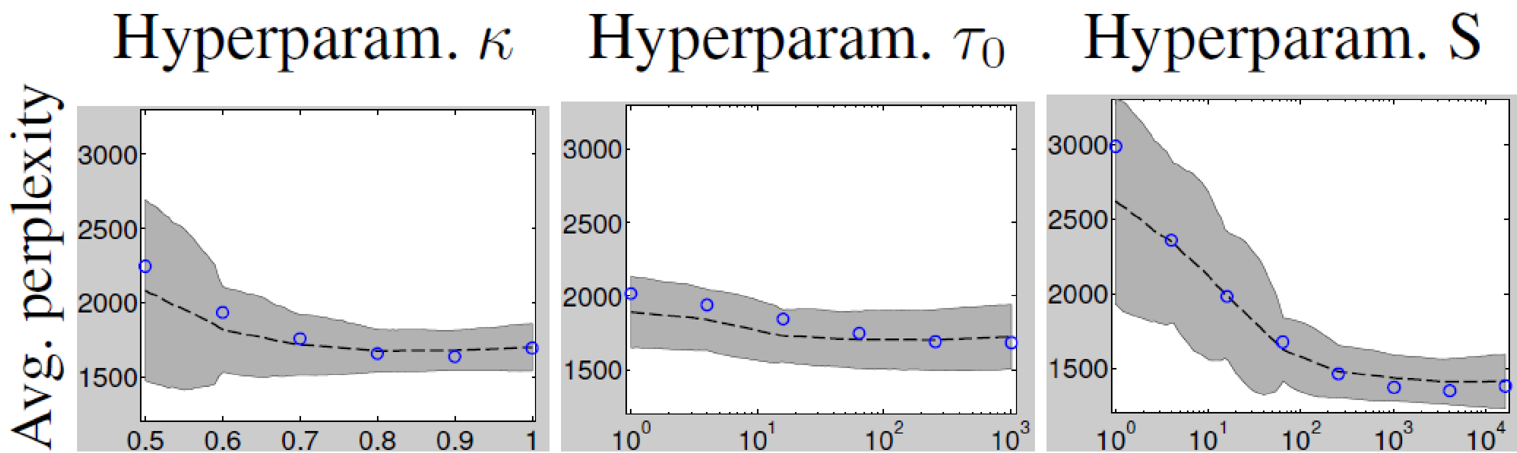
Local Hyperparameter Importance

- Starting from best known (incumbent) configuration:
  - Assess local effect of varying one parameter at a time
- Common analysis method
  - But typically used with additional runs around the incumbent $\rightarrow$ expensive
  - Can instead also use the model already built up during BO [Biedenkapp et al, 2018]
- Based on the BO model, this analysis takes milliseconds

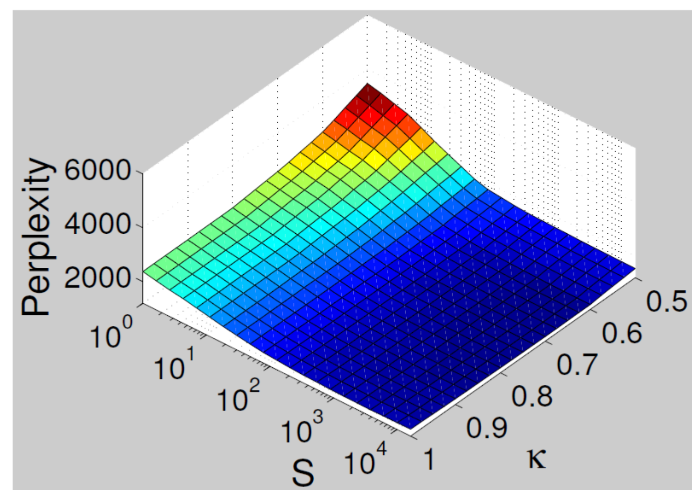


Global Hyperparameter Importance

Marginalize across all values of all other hyperparameters:



- Functional ANOVA [H. et al, 2014]
  - 65% of variance is due to S
  - Another 18% is due to interaction between S and κ
- These plots can be done as postprocessing for BOHB: link



Outline

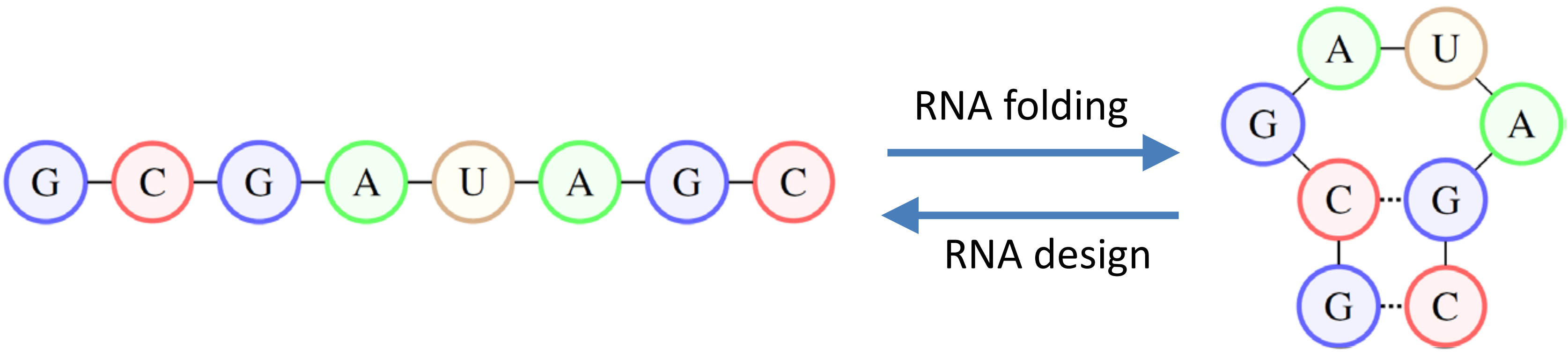
1. Blackbox Bayesian Hyperparameter Optimization
2. Beyond Blackbox: Speeding up Bayesian Optimization
3. Hyperparameter Importance Analysis
4. Case Studies
   a. BOHB for AutoRL
   b. Combining DARTS & BOHB for Auto-DispNet

Parallel work for AutoRL in robotics: [Chiang et al, ICRA 2019]

Application Domain: RNA Design

[Stoll et al, ICLR 2019]

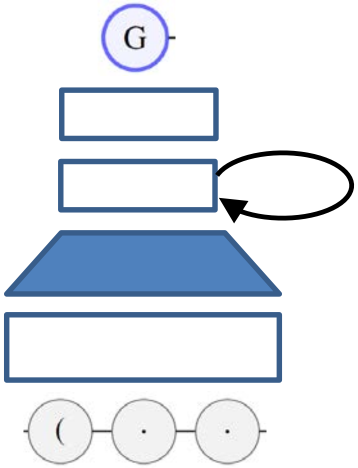
- Background on RNA:
  - Sequence of nucleotides (C, G, A, U)
  - Folds into a secondary structure, which determines its function
  - **RNA design**: find an RNA sequence that folds to a given structure



- RNA folding is $O(N^3)$ for sequences of length N
- RNA design is computationally hard
  - Typical approach: generate and test; local search
  - LEARNA: learns a policy network to sequentially design the sequence
  - Meta-LEARNA: meta-learn this policy across RNA sequences

AutoML for LEARNA and Meta-LEARNA ("Auto-RL")

- We optimize the policy network's neural architecture



Sampled action

Fully connected

Optional RNN (up to 2 layers)

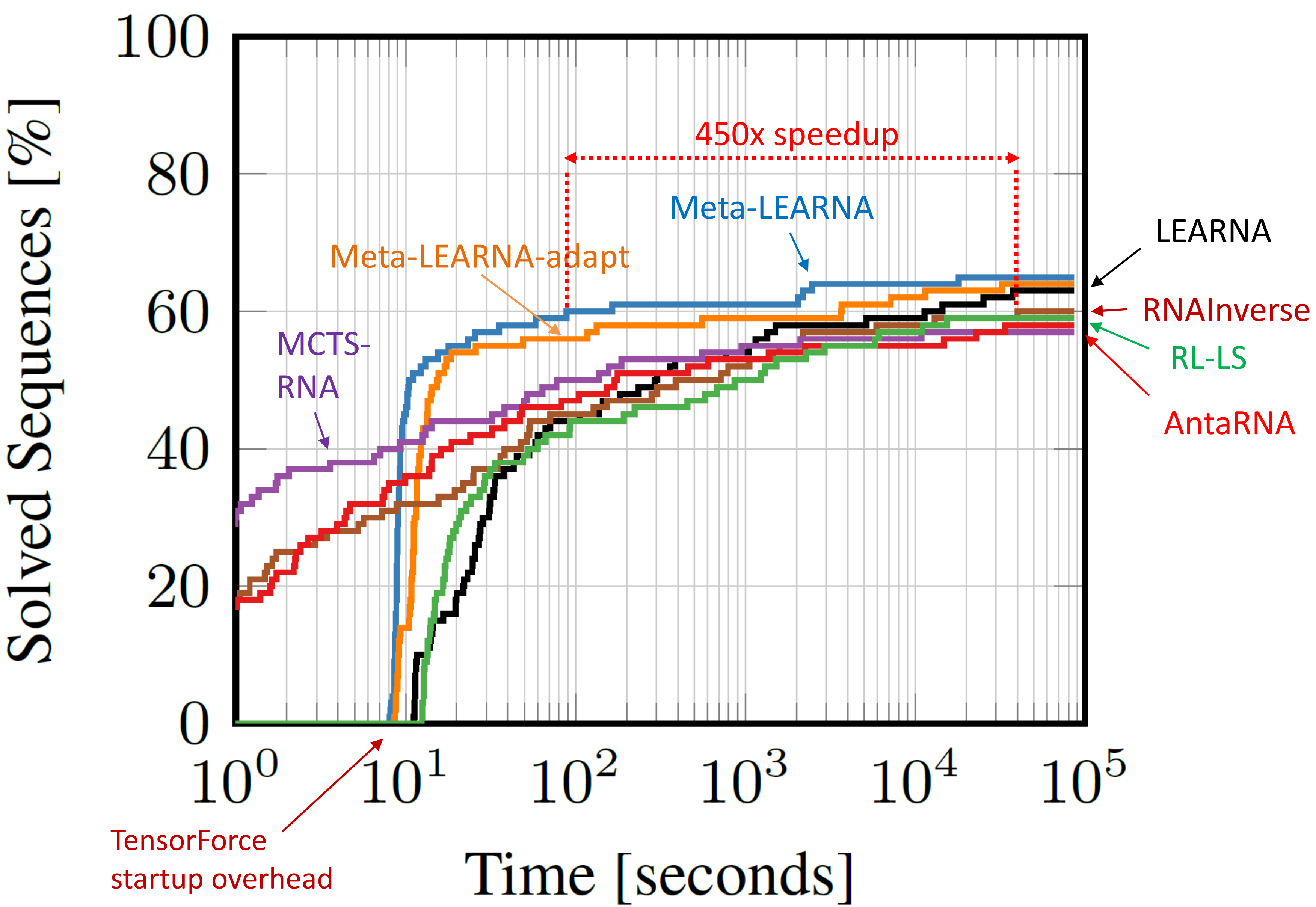
Optional CNN (up to 2 layers)

Optional embedding

State representation: n-grams

- At the same time, we jointly optimize further hyperparameters:
  - Length of n-grams (parameter of the decision process formulation)
  - Learning rate
  - Batch size
  - Strength of entropy regularization
  - Reward shaping

Results: 450x speedup over state-of-the-art



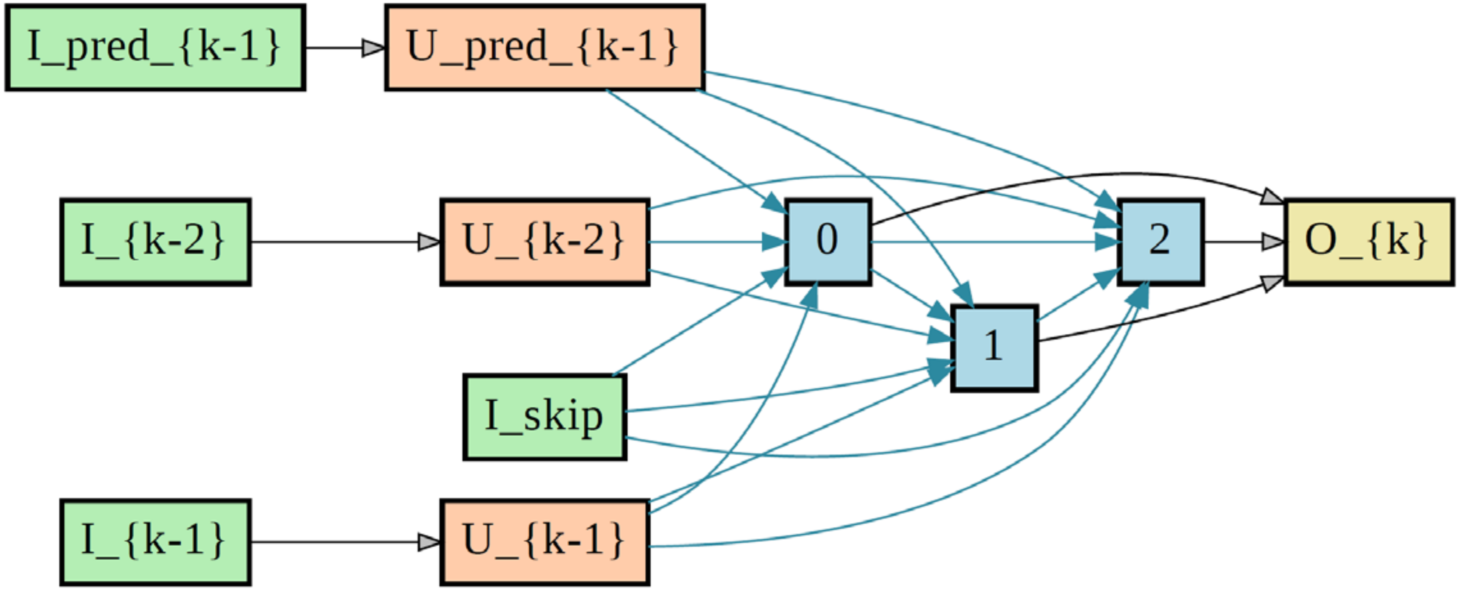
Outline

1. Blackbox Bayesian Hyperparameter Optimization
2. Beyond Blackbox: Speeding up Bayesian Optimization
3. Hyperparameter Importance Analysis
4. Case Studies
   a. BOHB for AutoRL
   b. Combining DARTS & BOHB for Auto-DispNet

Case Study 2: Auto-DispNet

[Saikia et al, arXiv 2019]

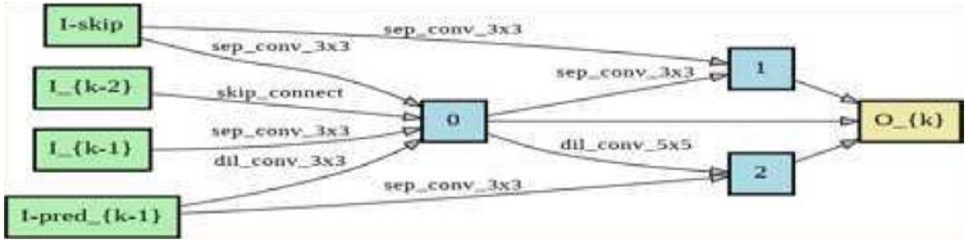
- Cell search space for a new upsampling cell with U-Net like skip connections



Case Study 2: Auto-DispNet

- **NAS: optimize neural architecture with DARTS**
  - Faster than BOHB
- **HPO: then optimize hyperparameters with BOHB**
  - DARTS does not apply
  - The weight sharing idea is restricted to the architecture space
- **Result:**
  - Both NAS and HPO yielded substantial improvements
  - E.g., EPE on Sintel: 2.36 -> 2.14 -> 1.94

Qualitative result



Selected upsampling cell

Conclusion

- Bayesian optimization (BO) allows joint neural architecture search & hyperparameter optimization
  - Yet, its vanilla blackbox optimization formulation is slow
- Going beyond blackbox BO leads to substantial speedups
  - Extrapolating learning curves
  - Reasoning across datasets
  - Multi-fidelity BO method BOHB is robust & efficient
- We can quantify the importance of hyperparameters
- Case studies: BOHB is versatile & practically useful
  - For “AutoRL“
  - For disparity estimation in combination with DARTS