Automatic Machine Learning (AutoML): A Tutorial

Frank Hutter

University of Freiburg
fh@cs.uni-freiburg.de

Joaquin Vanschoren

Eindhoven University of Technology
j.vanschoren@tue.nl

Slides available at automl.org/events -> AutoML Tutorial
(all references are clickable links)

Motivation: Successes of Deep Learning

Speech recognition

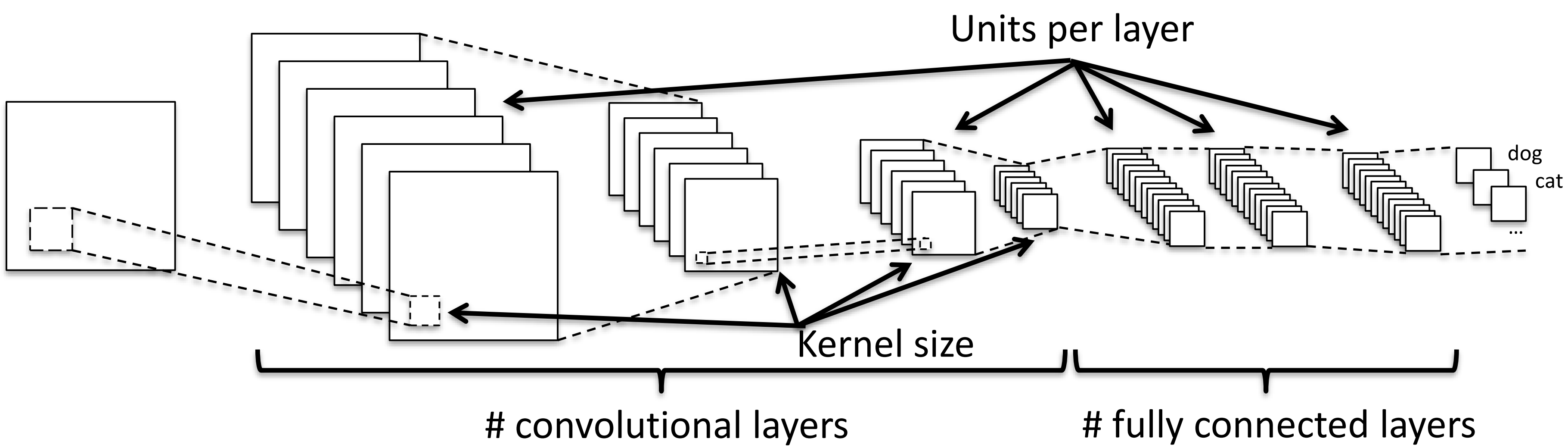
Computer vision in self-driving cars

Reasoning in games

One Problem of Deep Learning

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

- Architectural hyperparameters

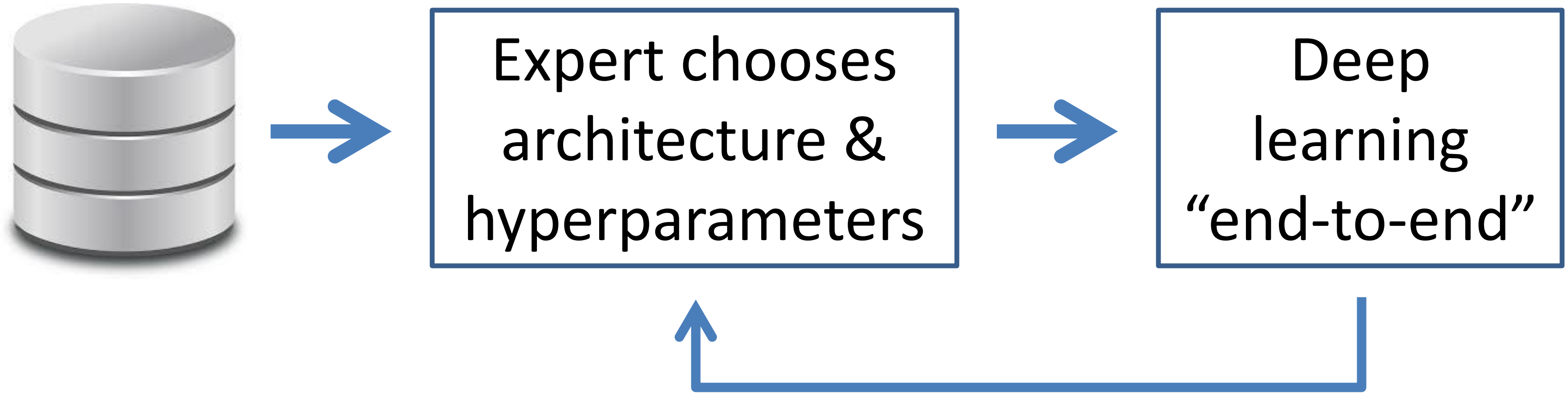


- Optimization algorithm, learning rates, momentum, batch normalization, batch sizes, dropout rates, weight decay, data augmentation, …

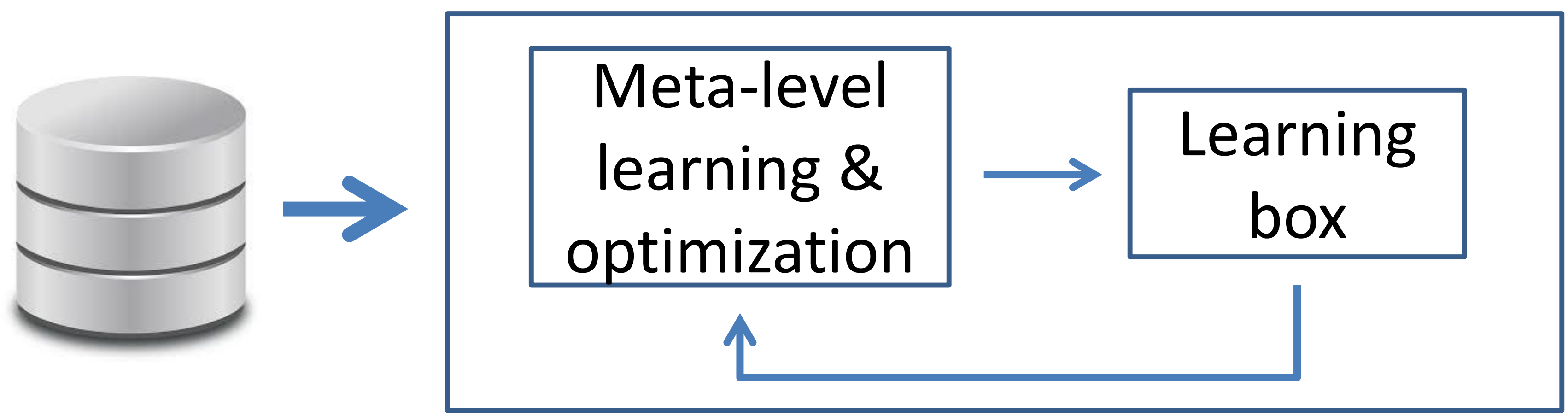
→ **Easily 20-50 design decisions**

Deep Learning and AutoML

Current deep learning practice



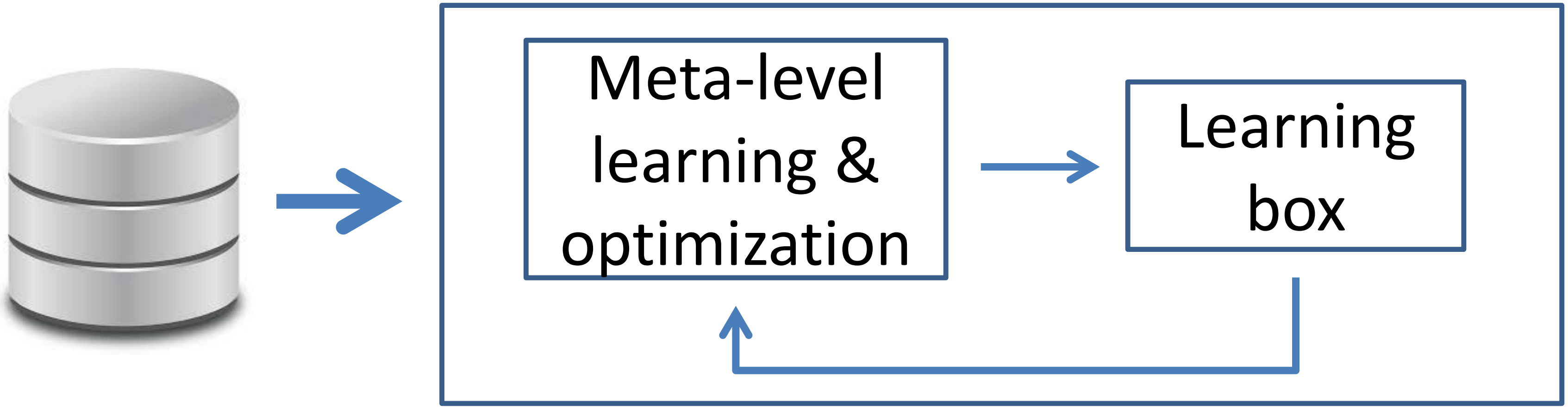
AutoML: true end-to-end learning



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

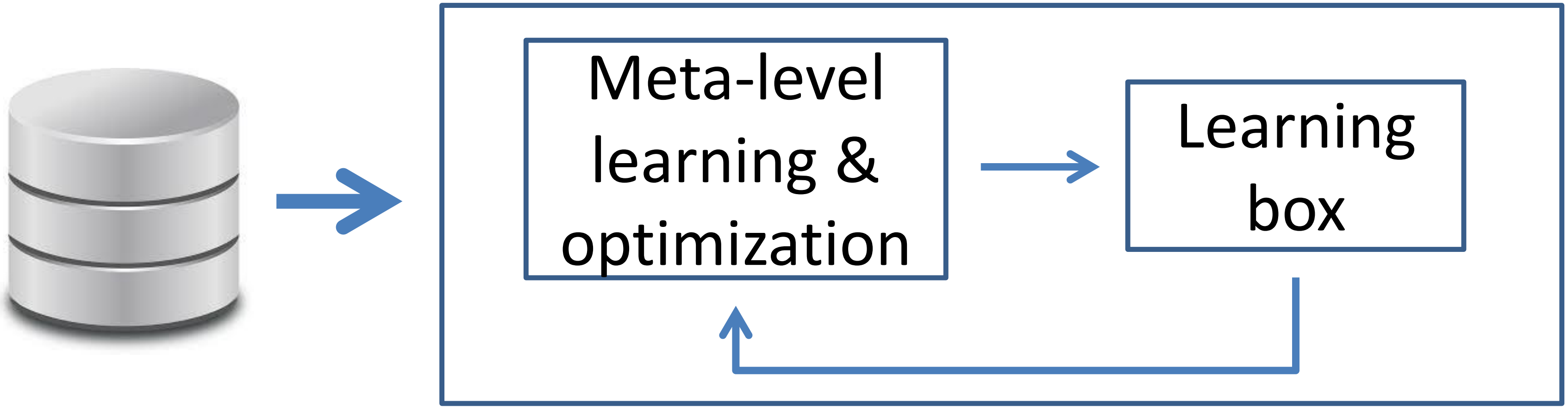


Outline

1. Modern Hyperparameter Optimization
2. Neural Architecture Search
3. Meta Learning

For more details, see: automl.org/book

AutoML: true end-to-end learning



Outline

1. Modern Hyperparameter Optimization
   - AutoML as Hyperparameter Optimization
   - Blackbox Optimization
   - Beyond Blackbox Optimization

Based on: Feuer & Hutter: Chapter 1 of the AutoML book: Hyperparameter Optimization

2. Neural Architecture Search
   - Search Space Design
   - Blackbox Optimization
   - Beyond Blackbox 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:

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

Types of Hyperparameters

- Continuous
  - Example: learning rate
- Integer
  - Example: #units
- Categorical
  - Finite domain, unordered
    - Example 1: algo ∈ {SVM, RF, NN}
    - 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
  - Example 1:
    - A = choice of optimizer (Adam or SGD)
    - B = Adam's second momentum hyperparameter (only active if A=Adam)
  - Example 2:
    - A = type of layer k (convolution, max pooling, fully connected, ...)
    - B = conv. kernel size of that layer (only active if A = convolution)
  - Example 3:
    - A = choice of classifier (RF or SVM)
    - B = SVM's kernel parameter (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:

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

→ Simply a HPO problem with a top-level hyperparameter (choice of algorithm) that all other hyperparameters are conditional on

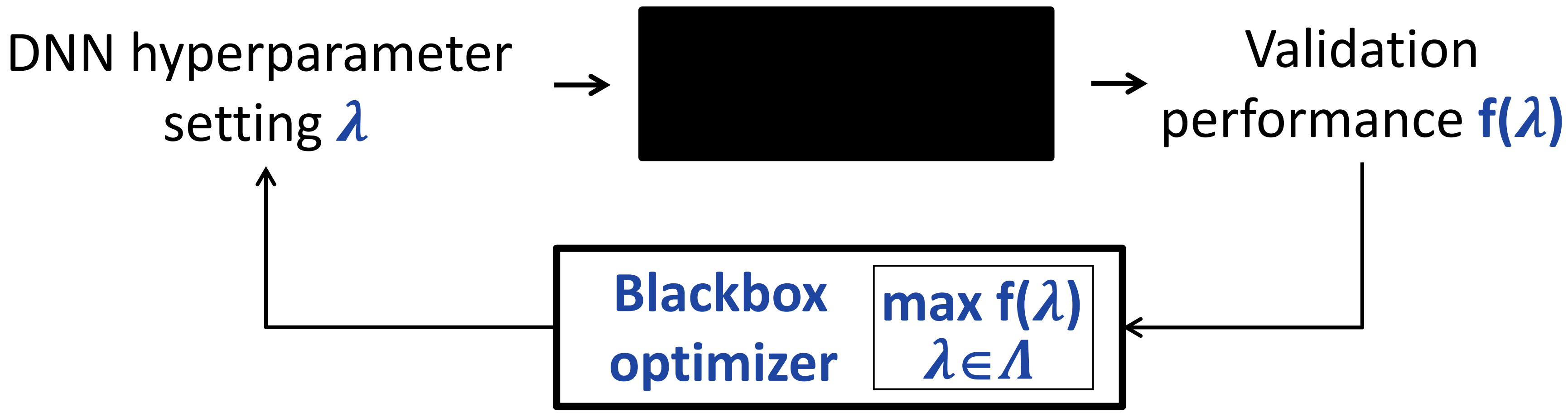
- E.g., Auto-WEKA: 768 hyperparameters, 4 levels of conditionality

Outline

1. Modern Hyperparameter Optimization
   - AutoML as Hyperparameter Optimization
   - Blackbox Optimization
   - Beyond Blackbox Optimization

2. Neural Architecture Search
   - Search Space Design
   - Blackbox Optimization
   - Beyond Blackbox Optimization

Blackbox Hyperparameter Optimization



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

Grid Search and Random Search

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

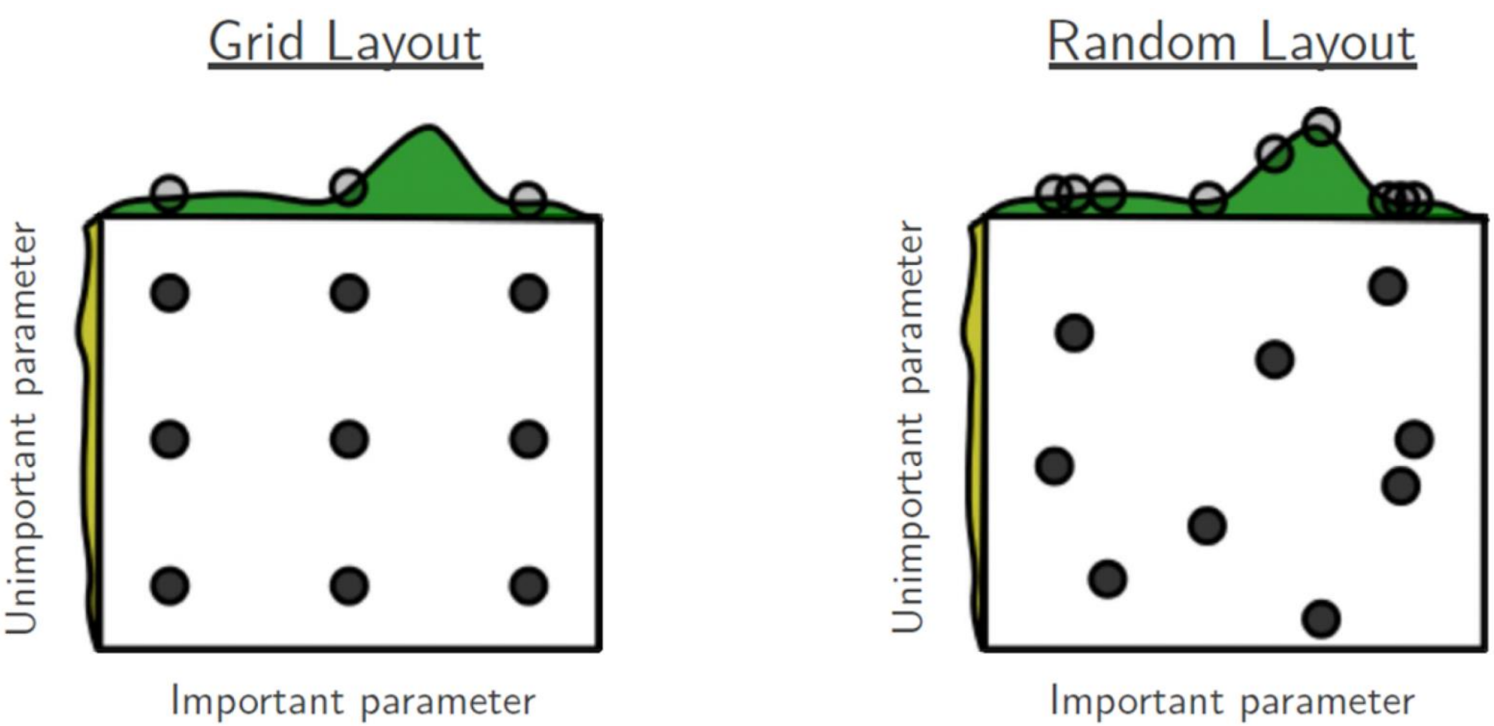


Image source: Bergstra & Bengio, JMLR 2012

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]

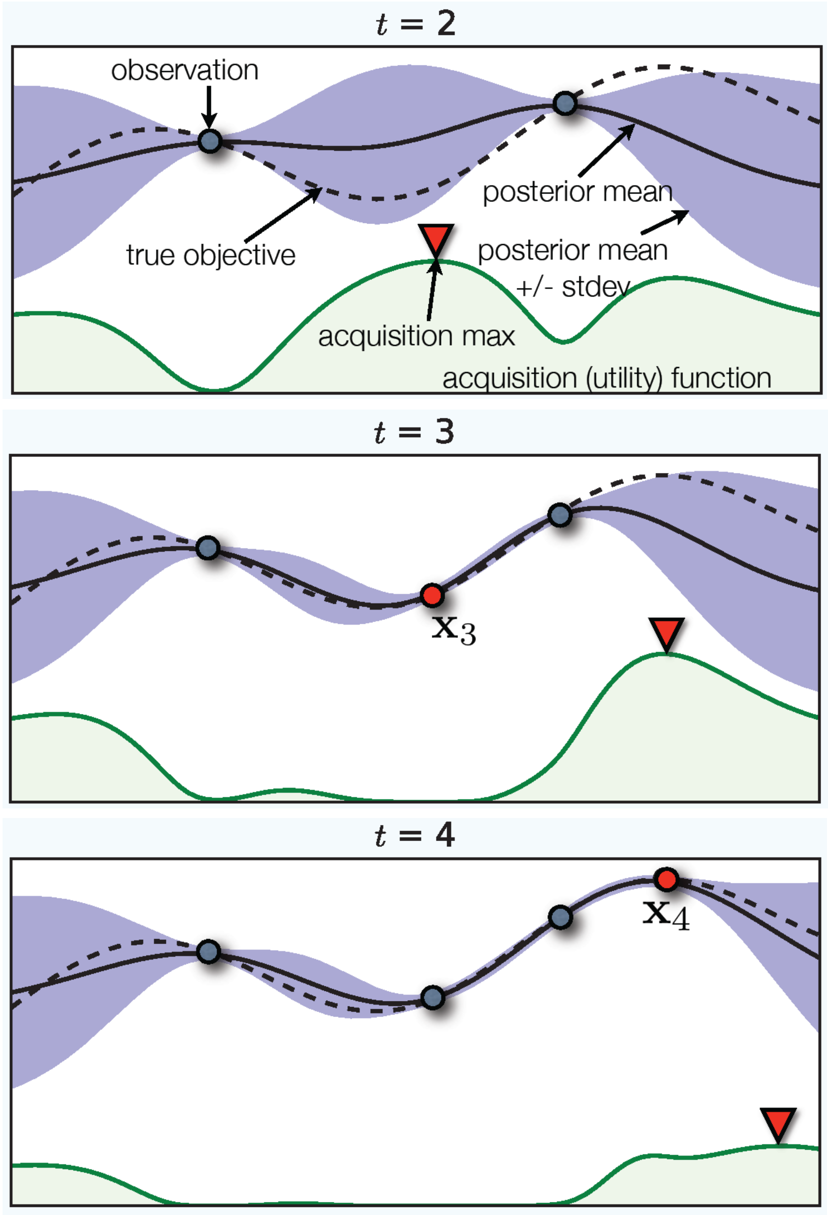


Image source: Brochu et al, 2010

Example: Bayesian Optimization in AlphaGo

[Source: email from Nando de Freitas, today; quotes from Chen et al, forthcoming]

- 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.

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., Swersky et al, 2013]
  - Noise: sometimes heteroscedastic, large, non-Gaussian
  - Robustness (usability out of the box)
  - Model overhead (budget is runtime, not #function evaluations)

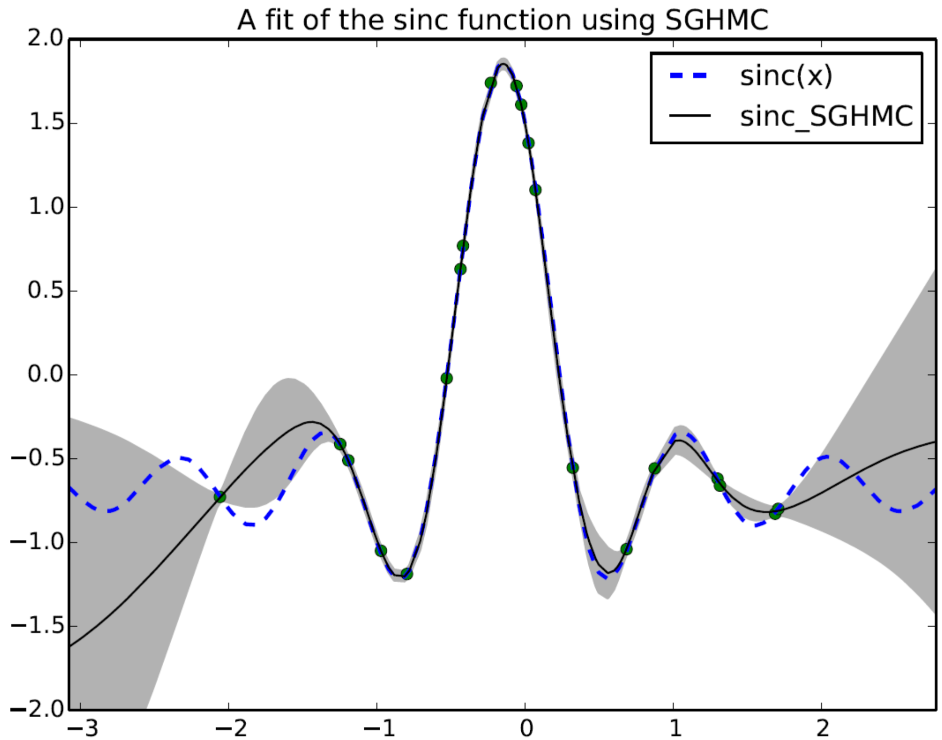
- Simple solution used in SMAC: random forests [Breiman, 2001]
  - Frequentist uncertainty estimate:
    variance across individual trees' predictions [Hutter et al, 2011]

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 HPOlib tasks

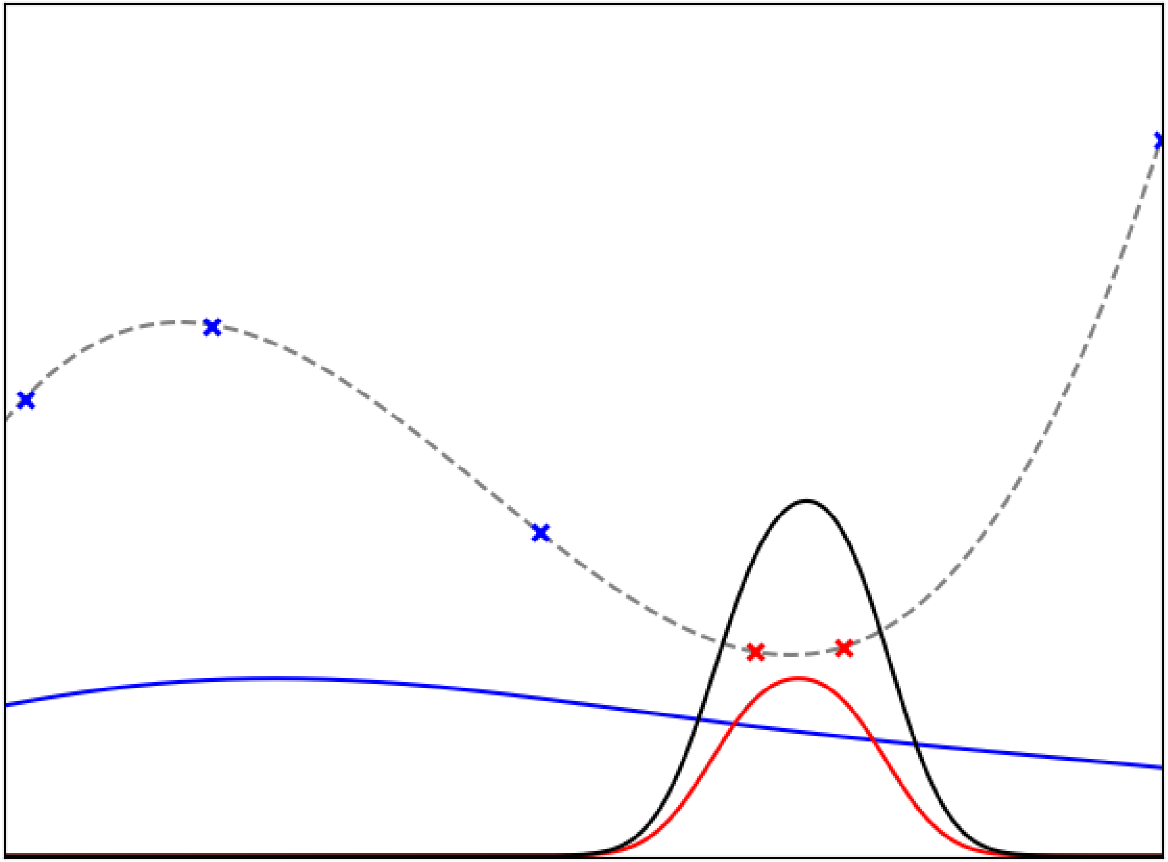
- So far not studied for:
  - High dimensionality
  - Conditional hyperparameters



Tree of Parzen Estimators (TPE)

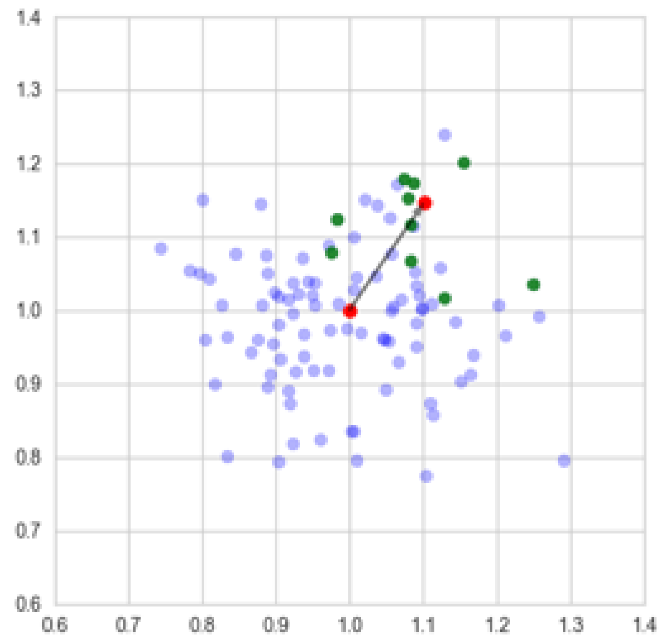
[Bergstra et al, NIPS 2011]

- 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



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 & Hutter, 2016]
    - Embarassingly parallel
    - Purely continuous

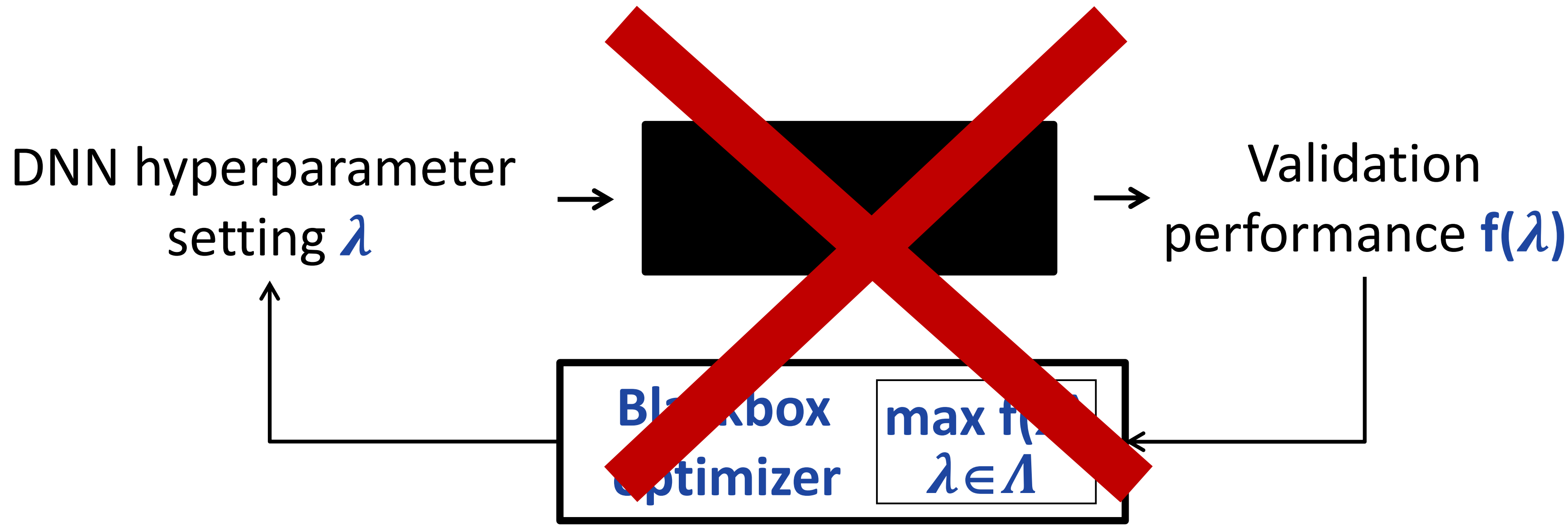


Outline

1. Modern Hyperparameter Optimization
   - AutoML as Hyperparameter Optimization
   - Blackbox Optimization
   - Beyond Blackbox Optimization

2. Neural Architecture Search
   - Search Space Design
   - Blackbox Optimization
   - Beyond Blackbox Optimization

Beyond Blackbox Hyperparameter Optimization



Too slow for DL / big data

Main Approaches Going Beyond Blackbox HPO

- Hyperparameter gradient descent
- Extrapolation of learning curves
- Multi-fidelity optimization
- Meta-learning [part 3 of this tutorial]

Hyperparameter Gradient Descent

- Formulation as bilevel optimization problem [e.g., Franceschi et al, ICML 2018]

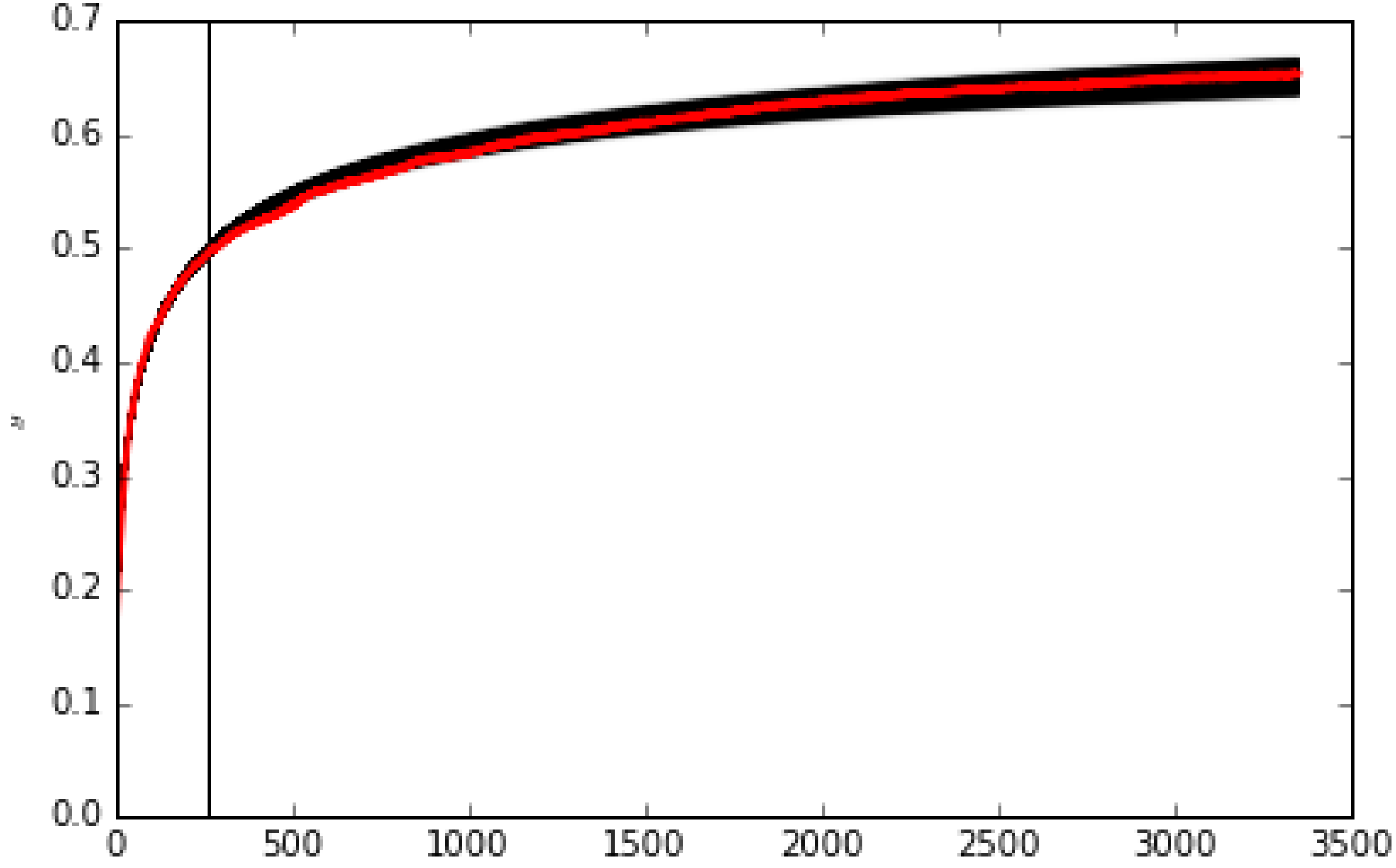
$$\boxed{\begin{aligned} & \min_\lambda \ \mathcal{L}_{val}\left(w^*(\lambda), \lambda\right) \\ s.t. \quad w^*(\lambda) \ &= \ \operatorname{argmin}_w \mathcal{L}_{train}\left(w, \lambda\right) \end{aligned}}$$

- Derive through the entire optimization process [MacLaurin et al, ICML 2015]

- Interleave optimization steps [Luketina et al, ICML 2016]

$$\boxed{\begin{aligned} &\text{Hyperparameter gradient step w.r.t. } \nabla_\lambda \mathcal{L}_{val} \\ &\text{Parameter gradient step w.r.t. } \nabla_w \mathcal{L}_{train} \end{aligned}}$$

Probabilistic Extrapolation of Learning Curves



- Parametric learning curve models [Domhan et al, IJCAI 2015]
- Bayesian neural networks [Klein et al, ICLR 2017]

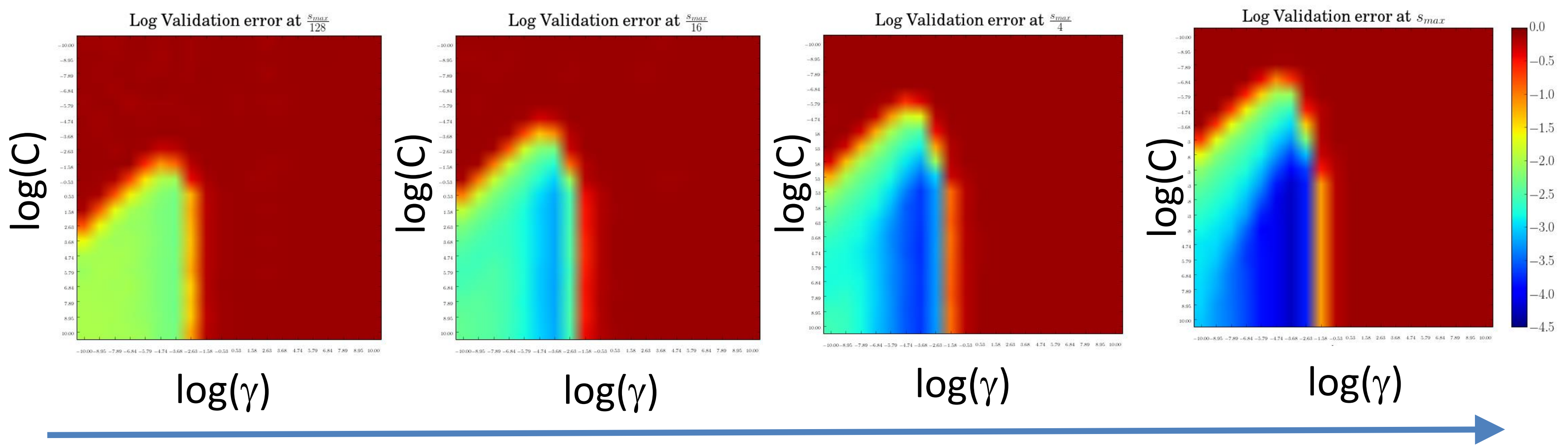
Multi-Fidelity Optimization

- 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

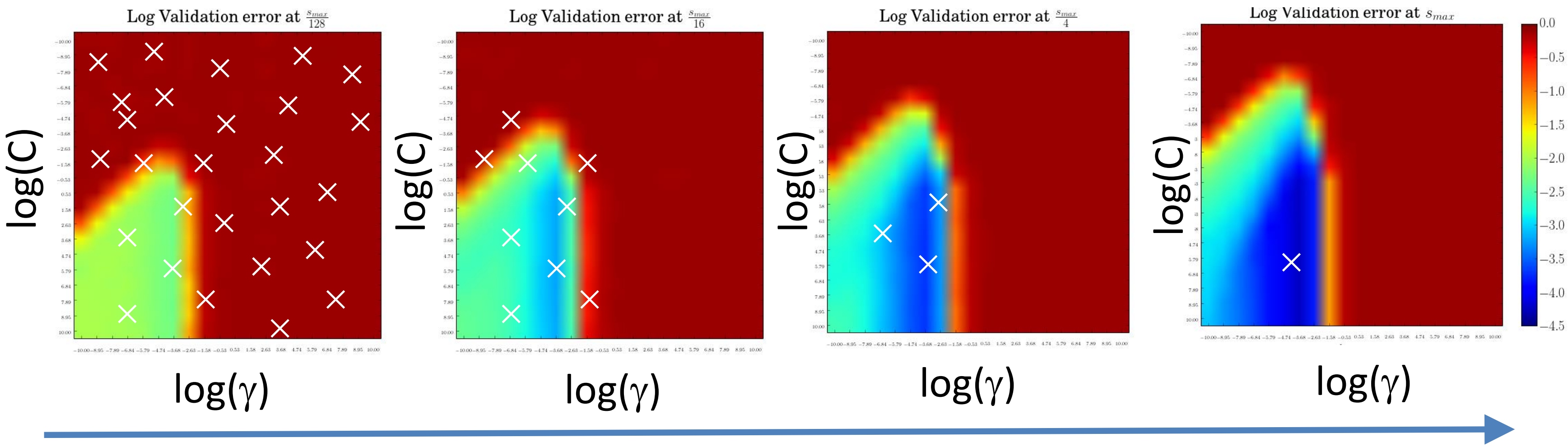
- **Make use of cheap low-fidelity evaluations**
  - E.g.: subsets of the data (here: SVM on MNIST)



Size of subset (of MNIST)

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

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



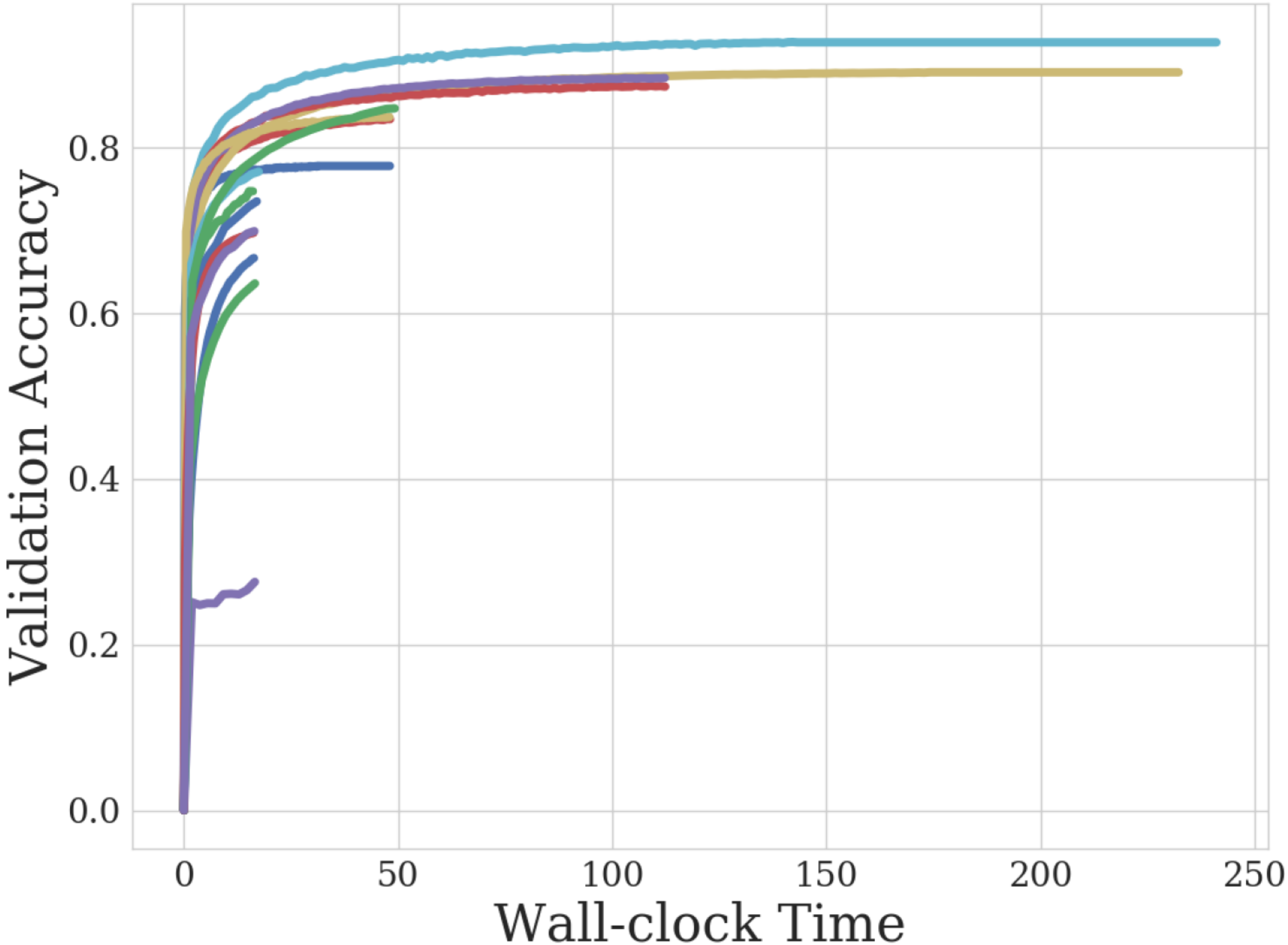
Size of subset (of MNIST)

  - 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]

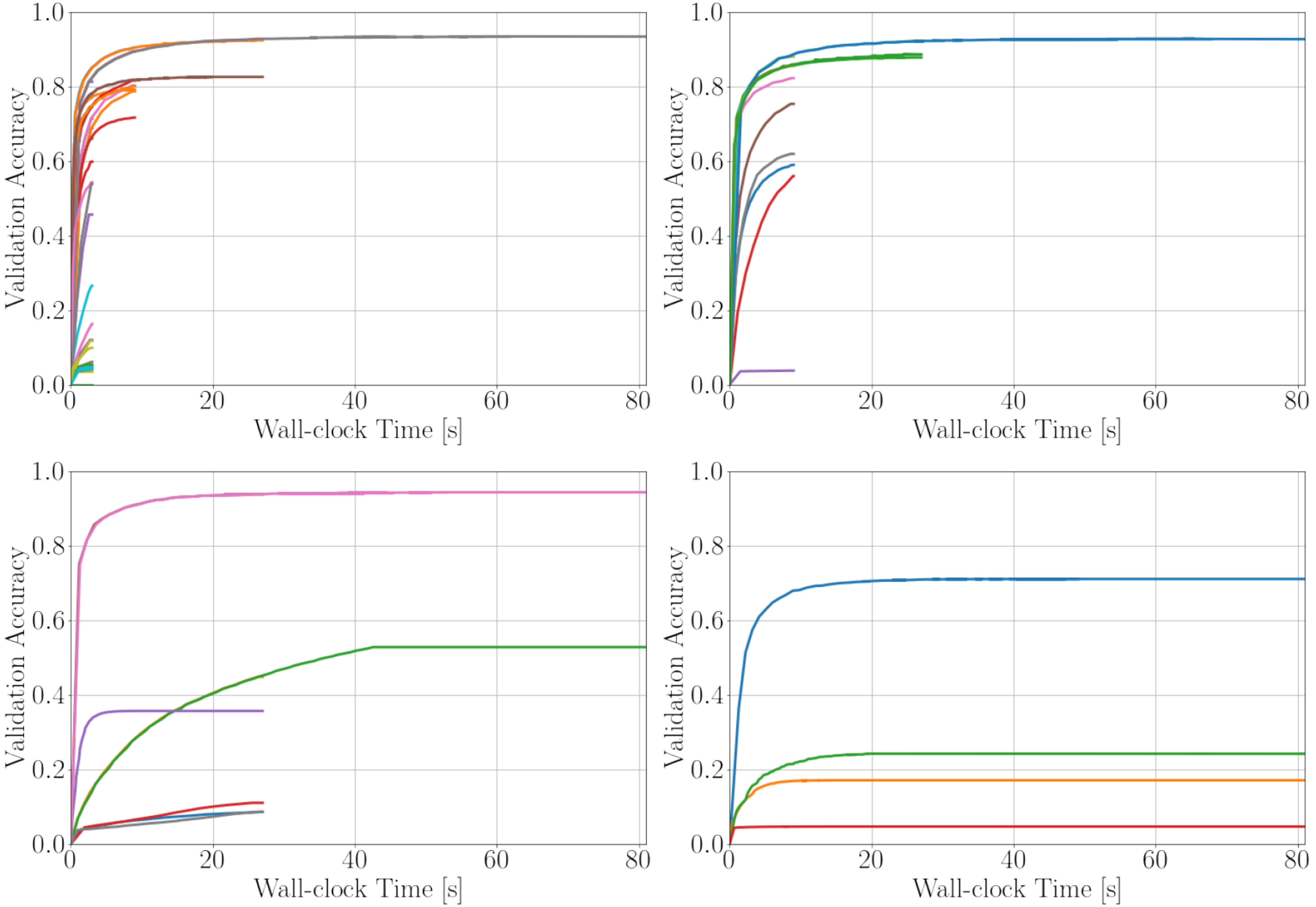
A Simpler Approach: Successive Halving (SH)

[Jamieson & Talwalkar, AISTATS 2016]



Hyperband (its first 4 calls to SH)

[Li et al, ICLR 2017]



BOHB: Bayesian Optimization & Hyperband

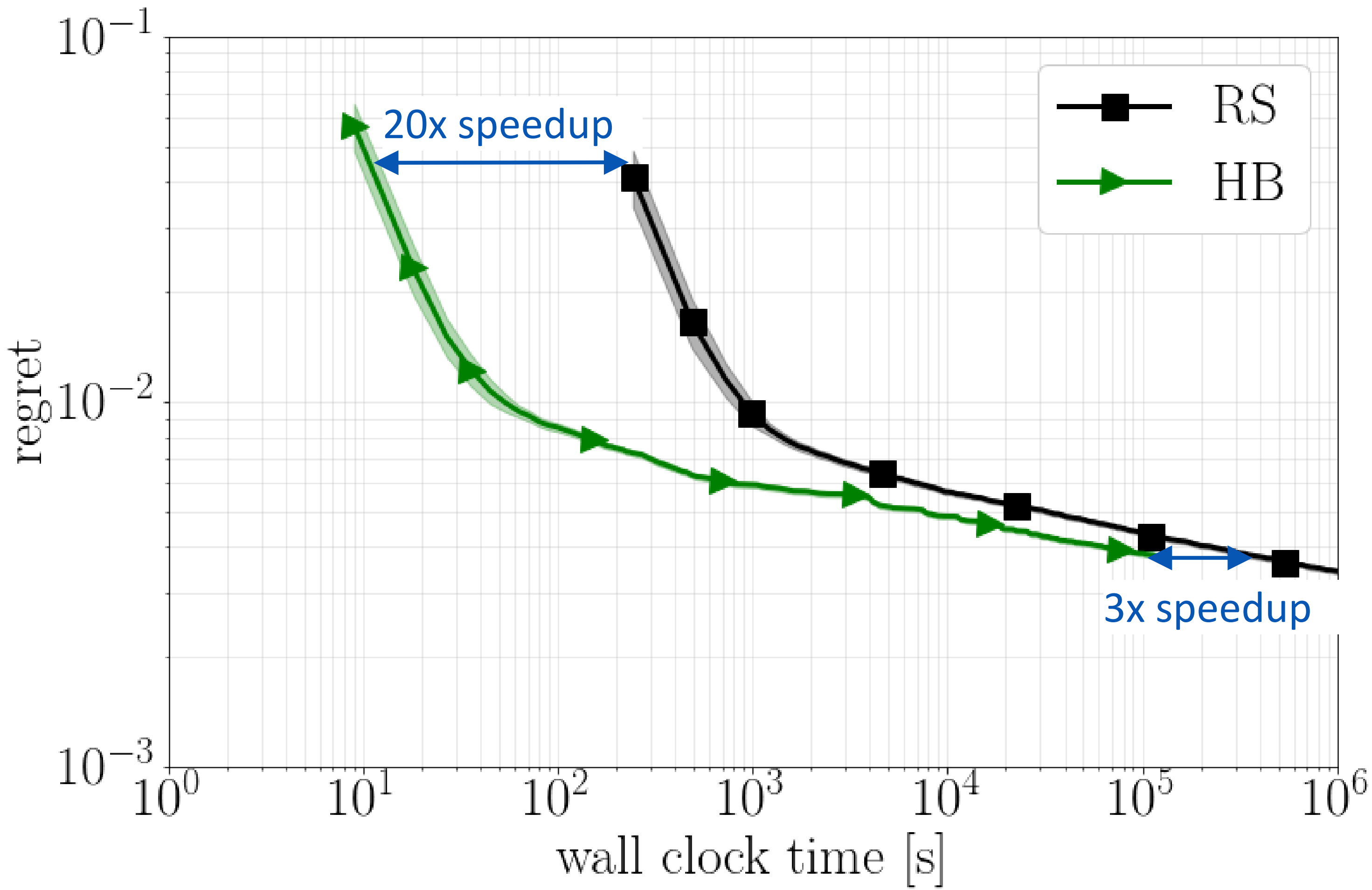
[Falkner, Klein & Hutter, ICML 2018]

- 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 configuration to evaluate (using a TPE variant)
  - Hyperband
    - for deciding how to allocate budgets

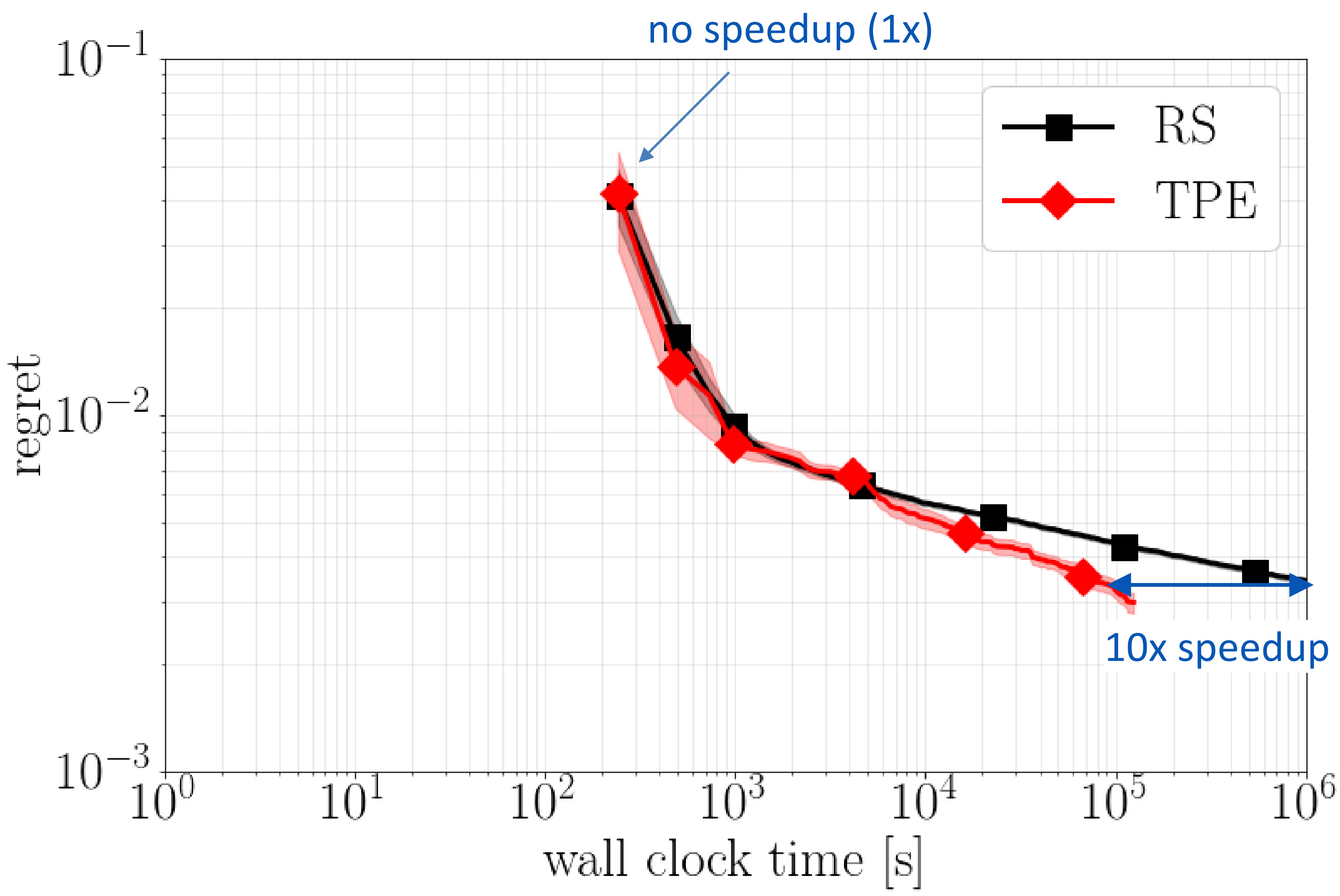
Hyperband vs. Random Search



Biggest advantage: much improved **anytime performance**

Auto-Net on dataset adult

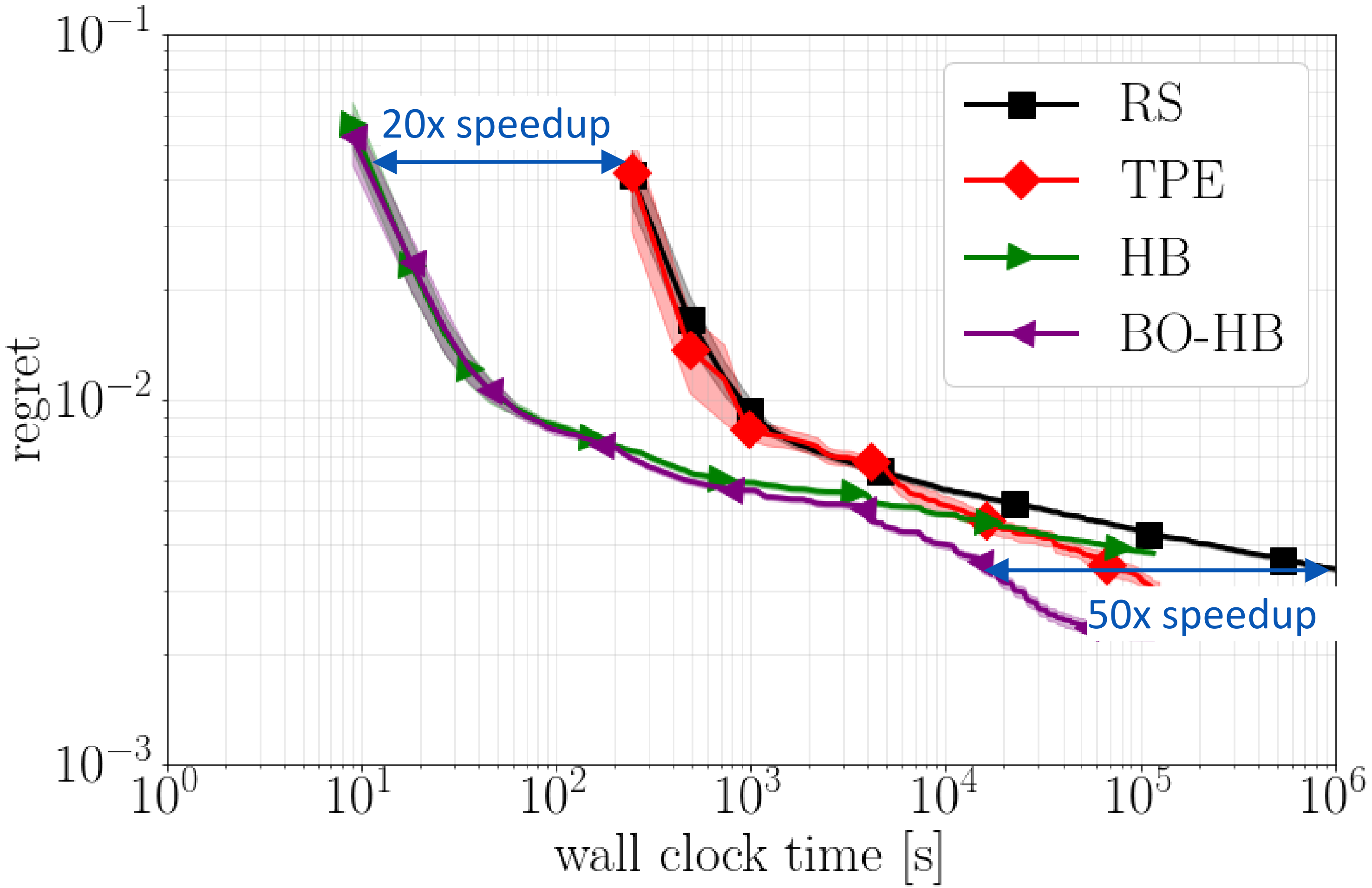
Bayesian Optimization vs Random Search



Biggest advantage: much improved **final performance**

Auto-Net on dataset adult

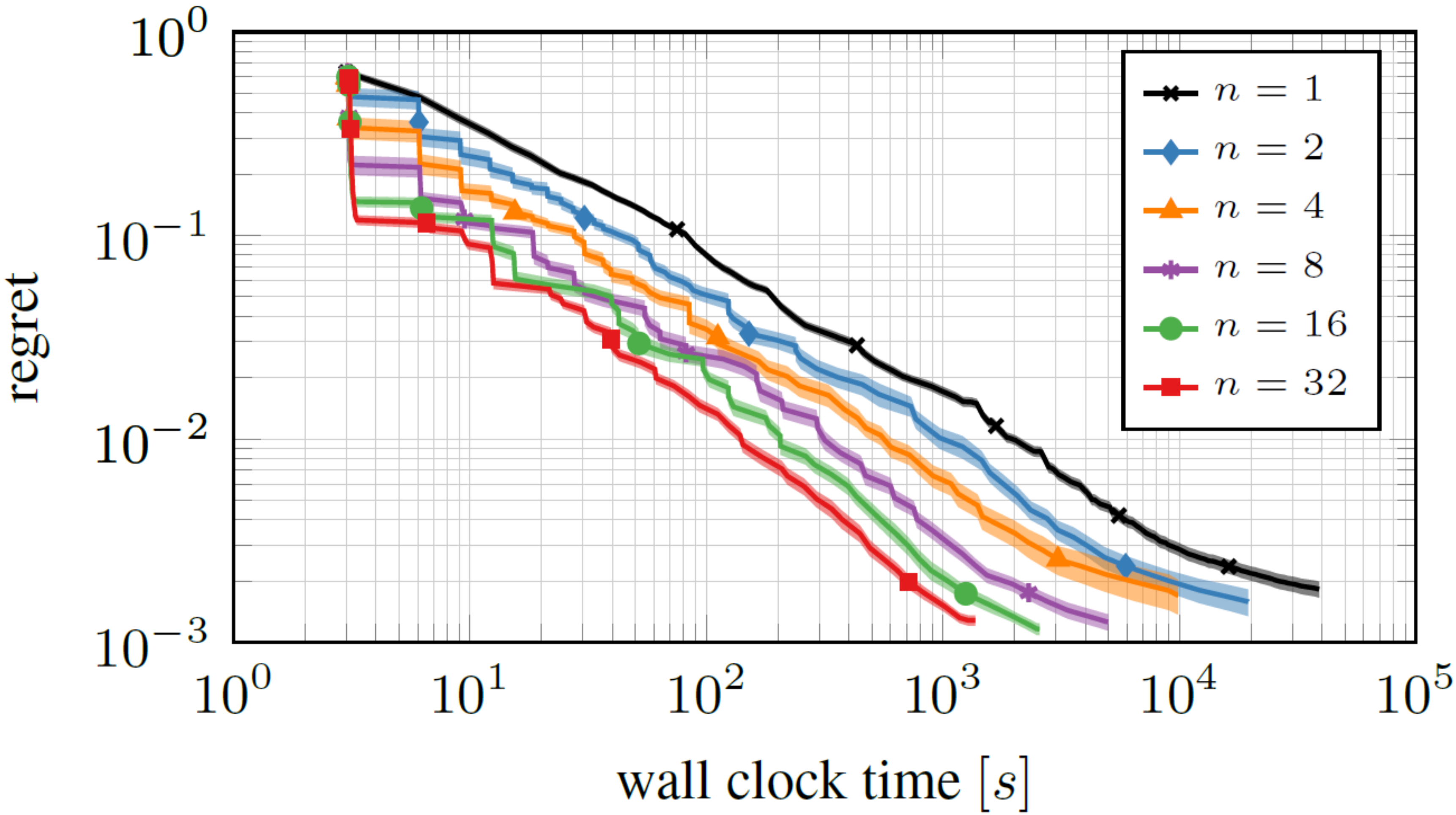
Combining Bayesian Optimization & Hyperband



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

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, NIPS 2015]
  - Based on scikit-learn & SMAC / BOHB
  - Uses meta-learning and posthoc ensembling
  - Won AutoML competitions 2015-2016 & 2017-2018
- TPOT [Olson et al, EvoApplications 2016]
  - Based on scikit-learn and evolutionary algorithms
- H2O AutoML [so far unpublished]
  - Based on random search and stacking

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)
```

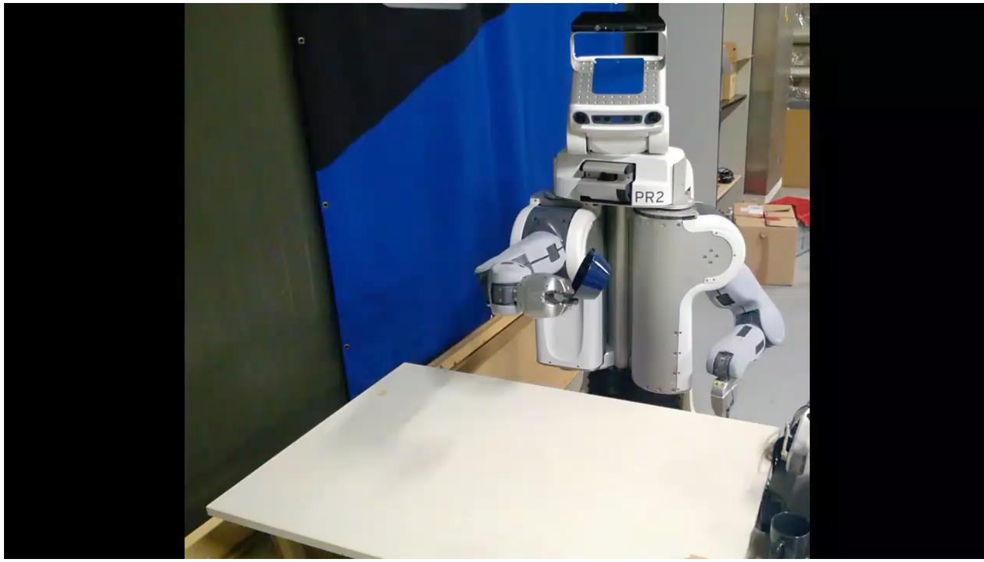
https://github.com/automl/auto-sklearn

Watch 182 | Star 2,601 | Fork 517

→ Effective machine learning for everyone!

Example Application: Robotic Object Handling

- Collaboration with Freiburg's robotics group
- Binary classification task for object placement: will the object fall over?
- Dataset
  - 30000 data points
  - 50 features -- manually defined [BSc thesis, Hauff 2015]
- Performance
  - Caffe framework & BSc student for 3 months: 2% error rate
  - **Auto-sklearn: 0.6% error rate** (within 30 minutes)



Video credit: Andreas Eitel

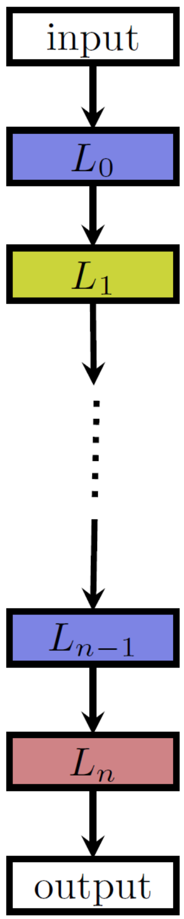
Outline

1. Modern Hyperparameter Optimization
   - AutoML as Hyperparameter Optimization
   - Blackbox Optimization
   - Beyond Blackbox Optimization

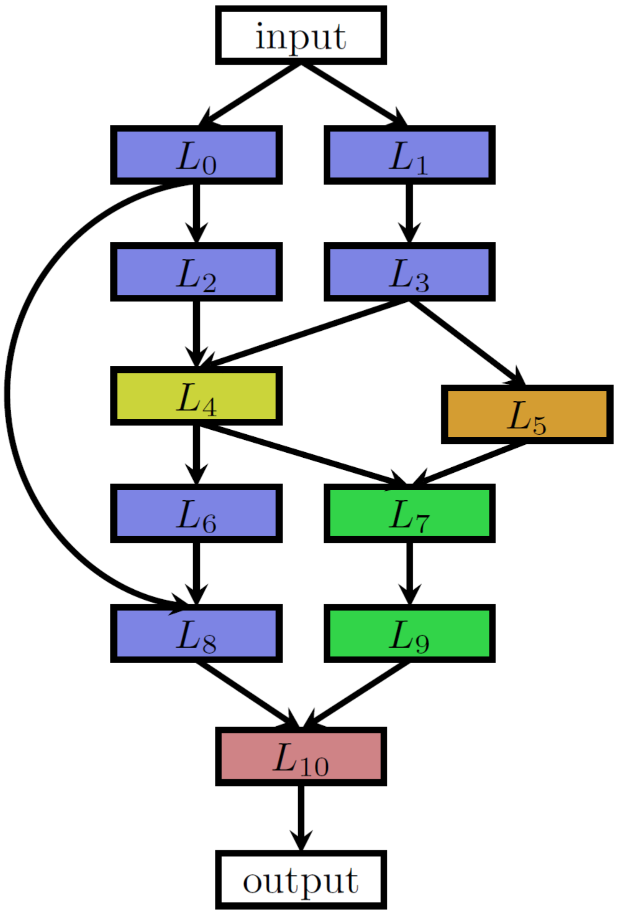
2. Neural Architecture Search
   - Search Space Design
   - Blackbox Optimization
   - Beyond Blackbox Optimization


Based on: Elsken, Metzen and Hutter
[Neural Architecture Search: a Survey, arXiv 2018; also Chapter 3 of the AutoML book]

Basic Neural Architecture Search Spaces



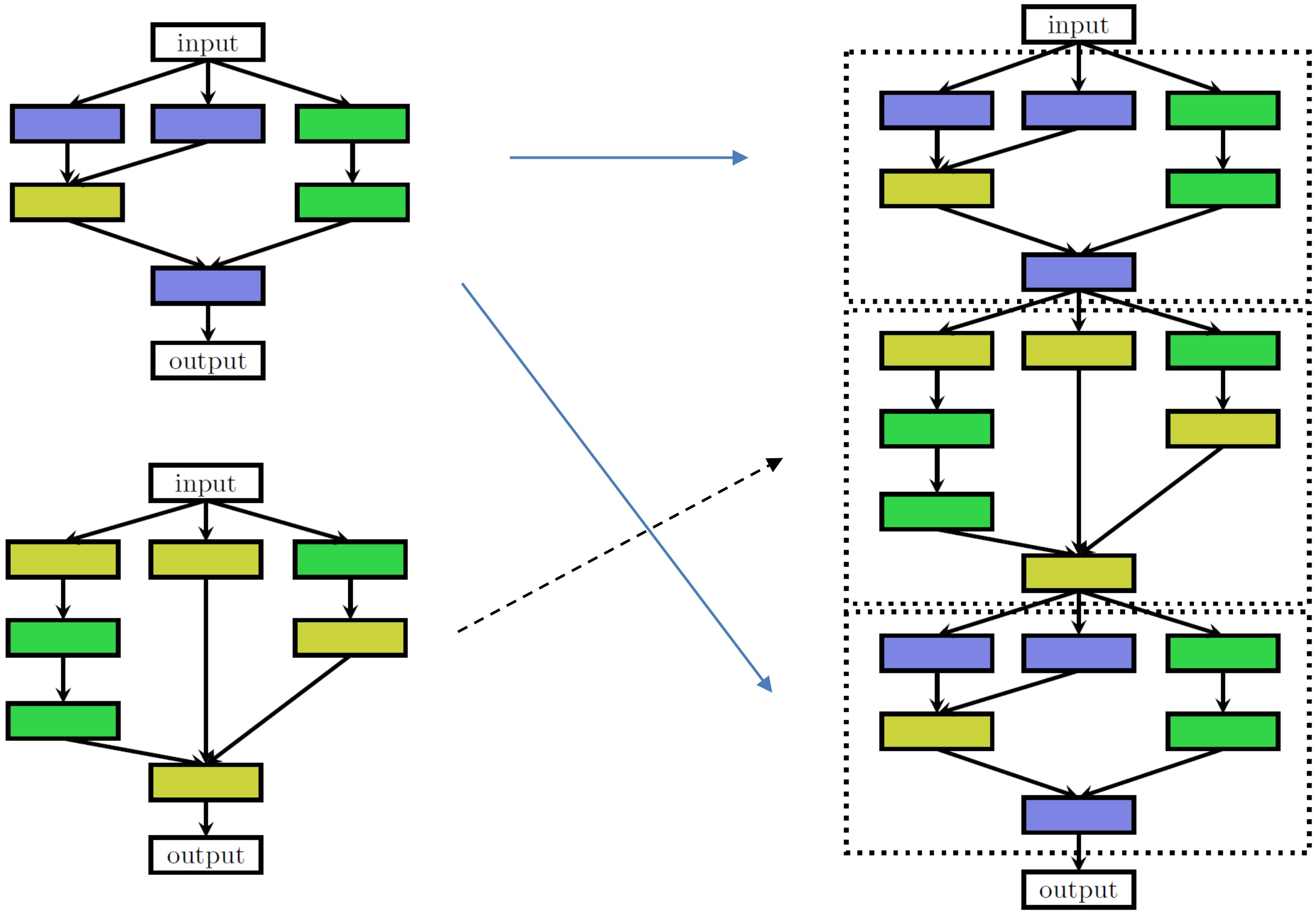
Chain-structured space
(different colours:
different layer types)



More complex space
with multiple branches
and skip connections

Cell Search Spaces

Introduced by Zoph et al [CVPR 2018]

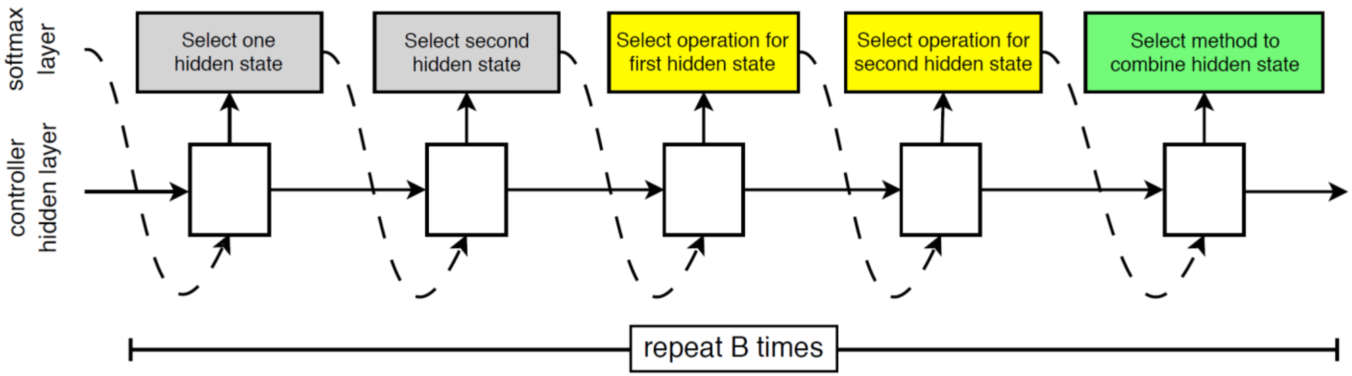


Two possible cells

Architecture composed of stacking together individual cells

NAS as Hyperparameter Optimization

- 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)

- Unrestricted search space
  – Possible with conditional hyperparameters
  (but only up to a prespecified maximum number of layers)
  – Example: chain-structured search space
    - Top-level hyperparameter: number of layers L
    - Hyperparameters of layer k conditional on L >= k

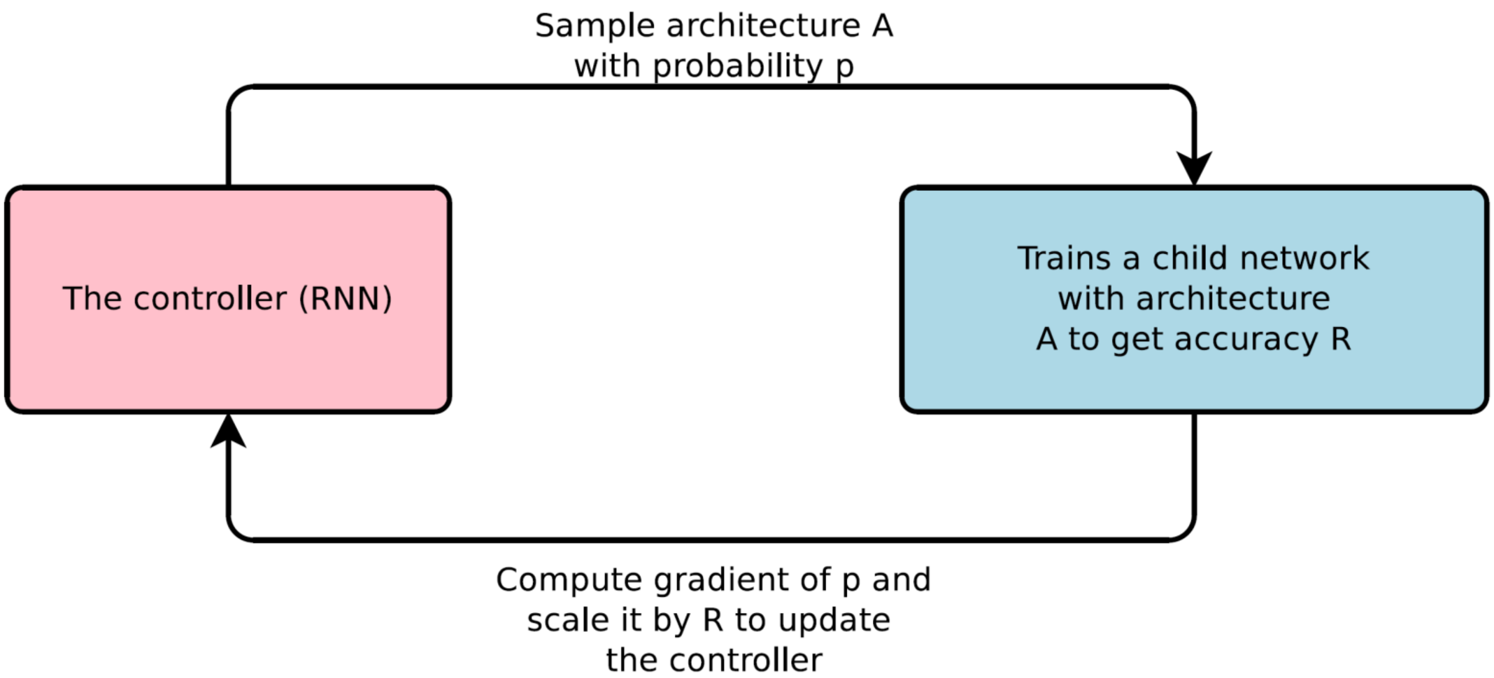
Outline

1. Modern Hyperparameter Optimization
   - AutoML as Hyperparameter Optimization
   - Blackbox Optimization
   - Beyond Blackbox Optimization

2. Neural Architecture Search
   - Search Space Design
   - Blackbox Optimization
   - Beyond Blackbox Optimization

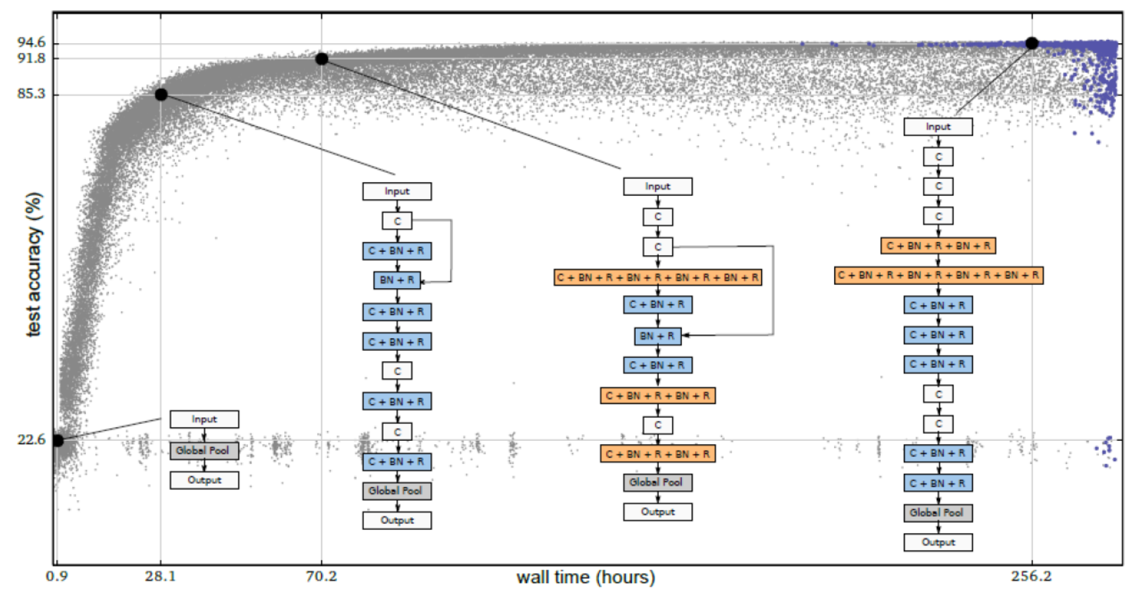
Reinforcement Learning

- NAS with Reinforcement Learning [Zoph & Le, ICLR 2017]
  - State-of-the-art results for CIFAR-10, Penn Treebank
  - Large computational demands
    - **800 GPUs for 3-4 weeks, 12.800 architectures evaluated**



Evolution

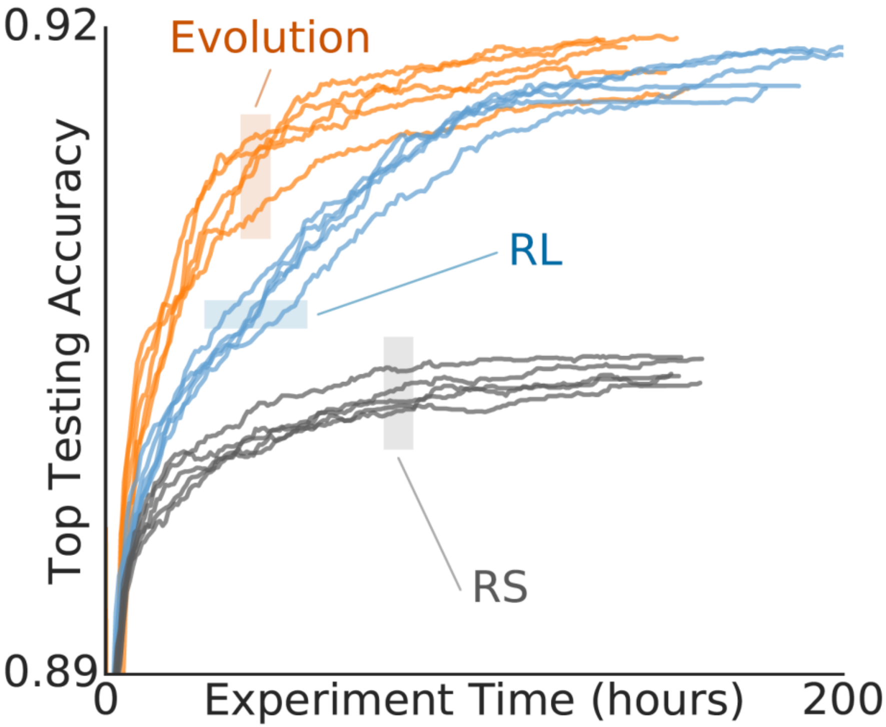
- Neuroevolution (already since the 1990s)
  - Typically optimized both architecture and weights with evolutionary methods
    [e.g., Angeline et al, 1994; Stanley and Miikkulainen, 2002]
  - Mutation steps, such as adding, changing or removing a layer
    [Real et al, ICML 2017; Miikkulainen et al, arXiv 2017]



Regularized / Aging Evolution

- Standard evolutionary algorithm [Real et al, AAAI 2019]
  - But oldest solutions are dropped from the population (even the best)
- State-of-the-art results (CIFAR-10, ImageNet)
  - Fixed-length cell search space

Comparison of evolution, RL and random search



Bayesian Optimization

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

- Auto-Net
  - Joint architecture and hyperparameter search with SMAC
  - First Auto-DL system to win a competition dataset against human experts [Mendoza et al, AutoML 2016]

- 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. Modern Hyperparameter Optimization
   - AutoML as Hyperparameter Optimization
   - Blackbox Optimization
   - Beyond Blackbox Optimization

2. Neural Architecture Search
   - Search Space Design
   - Blackbox Optimization
   - Beyond Blackbox Optimization

Main approaches for making NAS efficient

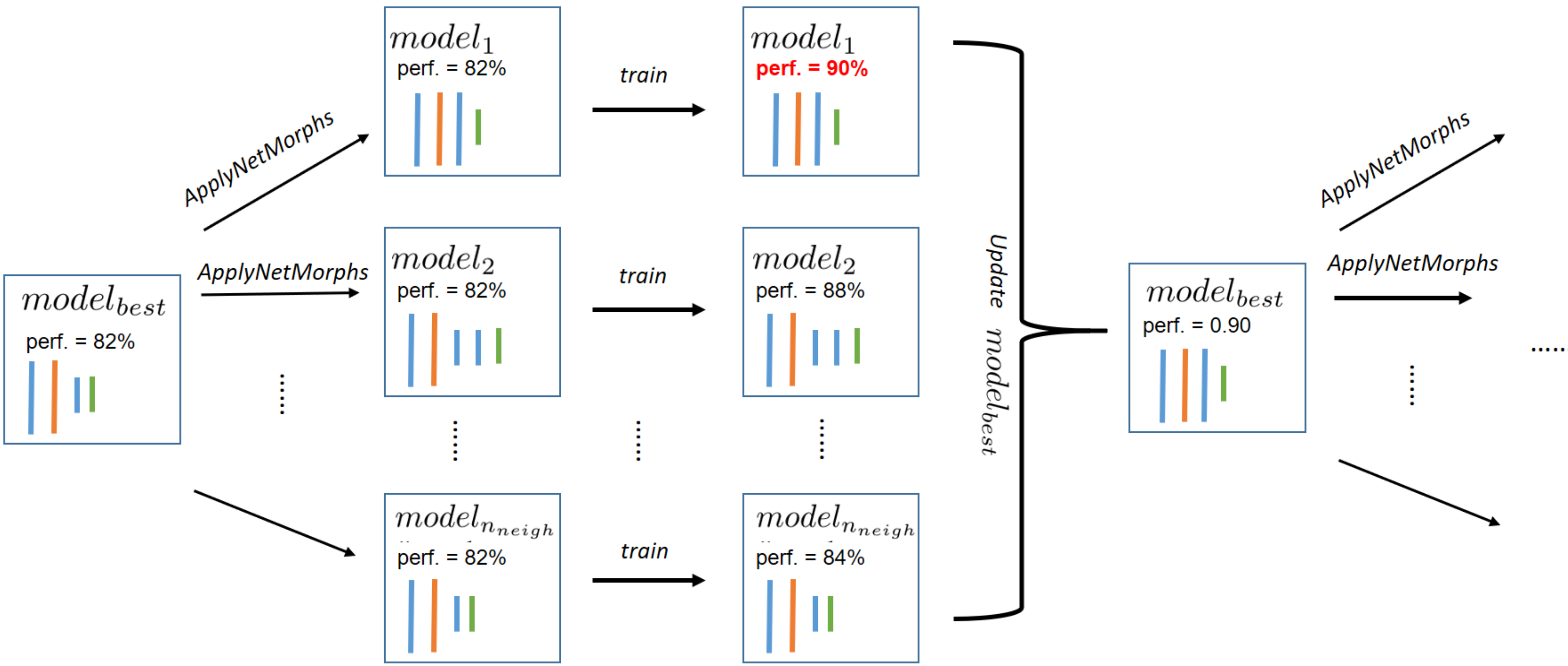
- Weight inheritance & network morphisms
- Weight sharing & one-shot models
- Multi-fidelity optimization
  [Zela et al, AutoML 2018, Runge et al, MetaLearn 2018]
- Meta-learning [Wong et al, NIPS 2018]

Network morphisms

- Network morphisms [Chen et al, 2016; Wei et al, 2016; Cai et al, 2017]
  - Change the network structure, but not the modelled function
    - I.e., for every input the network yields the same output as before applying the network morphism
  - Allow efficient moves in architecture space

Weight inheritance & network morphisms

[Cai et al, AAAI 2018; Elsken et al, MetaLearn 2017; Cortes et al, ICML 2017; Cai et al, ICML 2018]



→ enables efficient architecture search

Weight Sharing & One-shot Models

- Convolutional Neural Fabrics [Saxena & Verbeek, NIPS 2016]
  - Embed an exponentially large number of architectures
  - Each path through the fabric is an architecture

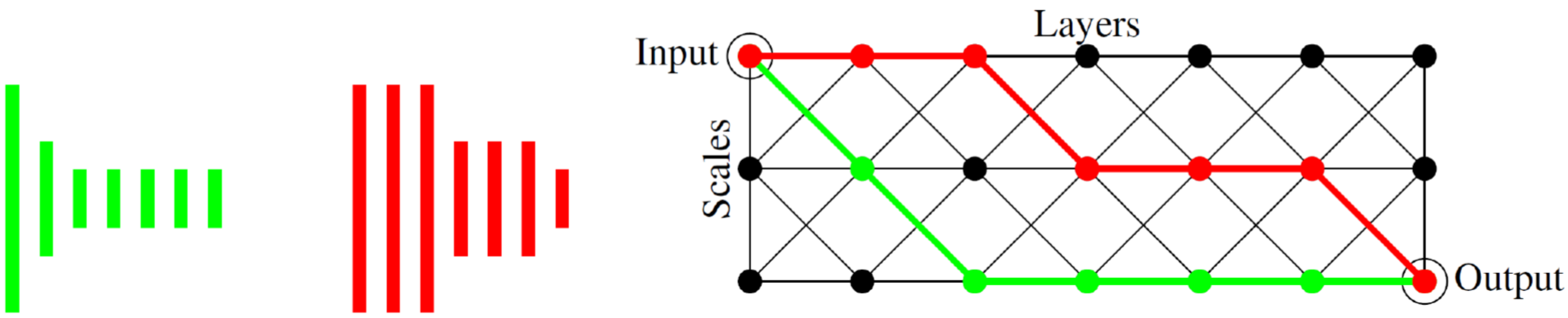
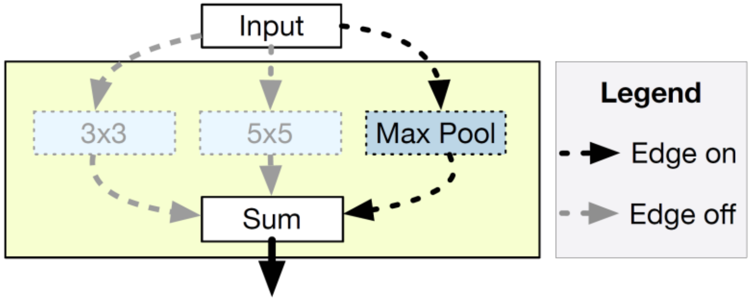


Figure: Fabrics embedding two 7-layer CNNs (red, green).
Feature map sizes of the CNN layers are given by height.

Weight Sharing & One-shot Models

- Simplifying One-Shot Architecture Search [Bender et al, ICML 2018]
  - Use path dropout to make sure the individual models perform well by themselves



- ENAS [Pham et al, ICML 2018]
  - Use RL to sample paths (=architectures) from one-shot model

- SMASH [Brock et al, MetaLearn 2017]
  - Train hypernetwork that generates weights of models

DARTS: Differentiable Neural Architecture Search

[Liu et al, Simonyan, Yang, arXiv 2018]

- Relax the discrete NAS problem
  - One-shot model with continuous architecture weight α for each operator
  - Use a similar approach as Luketina et al [ICML'16] to interleave optimization steps of α (using validation error) and network weights

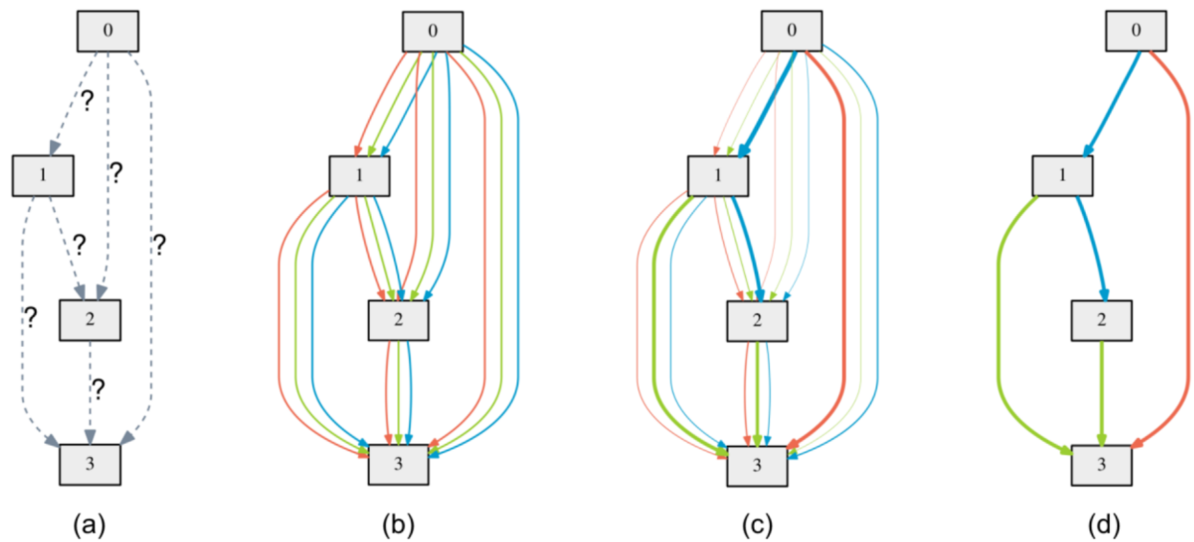


Figure 1: An overview of DARTS: (a) Operations on the edges are initially unknown. (b) Continuous relaxation of the search space by placing a mixture of candidate operations on each edge. (c) Joint optimization of the mixing probabilities and the network weights by solving a bilevel optimization problem. (d) Inducing the final architecture from the learned mixing probabilities.

Some Promising Work Under Review

- Anonymous ICLR submissions based on DARTS
  - SNAS: Use Gumbel softmax on architecture weights α [link]
  - Single shot NAS: use L1 penalty to sparsify architecture [link]
  - Proxyless NAS: (PyramidNet-based) memory-efficient variant of DARTS that trains sparse architectures only [link]

- Graph hypernetworks for NAS [Anonymous ICLR submission]

- Multi-objective NAS
  - MNasNet: scalarization [Tan et al, arXiv 2018]
  - LEMONADE: evolution & (approximate) network morphisms [Anonymous ICLR submission]

Remarks on Experimentation in NAS

- Final results are often incomparable due to
  - Different training pipelines without available source code
    - Releasing the final architecture does not help for comparisons
  - Different hyperparameter choices
    - Very different hyperparameters for training and final evaluation
  - Different search spaces / initial models
    - Starting from random or from PyramidNet?

→ Need for looking beyond the error numbers on CIFAR

→ Need for benchmarks including training pipeline & hyperparams

- Experiments are often very expensive

→ Need for cheap benchmarks that allow for many runs

HPO and NAS Wrapup

- Exciting research fields, lots of progress
- Several ways to speed up blackbox optimization
  - Multi-fidelity approaches
  - Hyperparameter gradient descent
  - Weight inheritance
  - Weight sharing & hypernetworks
- More details in AutoML book: automl.org/book
- Advertisement: we‘re building up an Auto-DL team
  - Building research library of building blocks for efficient NAS
  - Building open-source framework Auto-PyTorch
  - We have several openings on all levels (postdocs, PhD students, research engineers); see automl.org/jobs

AutoML and Job Loss Through Automation

- Concern about too much automation, job loss
  - AutoML will allow humans to become more productive
  - Thus, it will eventually reduce the work left for data scientists
  - But it will also help many domain scientists use machine learning that would otherwise not have used it
    - This creates more demand for interesting and creative work

- Call to arms: let‘s use AutoML to create and improve jobs
  - If you can think of a business opportunity that‘s made feasible by AutoML (robust, off-the-shelf, effective ML), now is a good time to act on it ...

AutoML: Further Benefits and Concerns

+ Democratization of data science
+ We directly have a strong baseline
+ We can codify best practices
+ Reducing the tedious part of our work, freeing time to focus on problems humans do best (creativity, interpretation, …)

− People will use it without understanding anything