# Towards Automated Deep Learning

**Frank Hutter**

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

@FrankRHutter
@AutoMLFreiburg

# Motivation: Successes of Deep Learning

Computer vision in self-driving cars

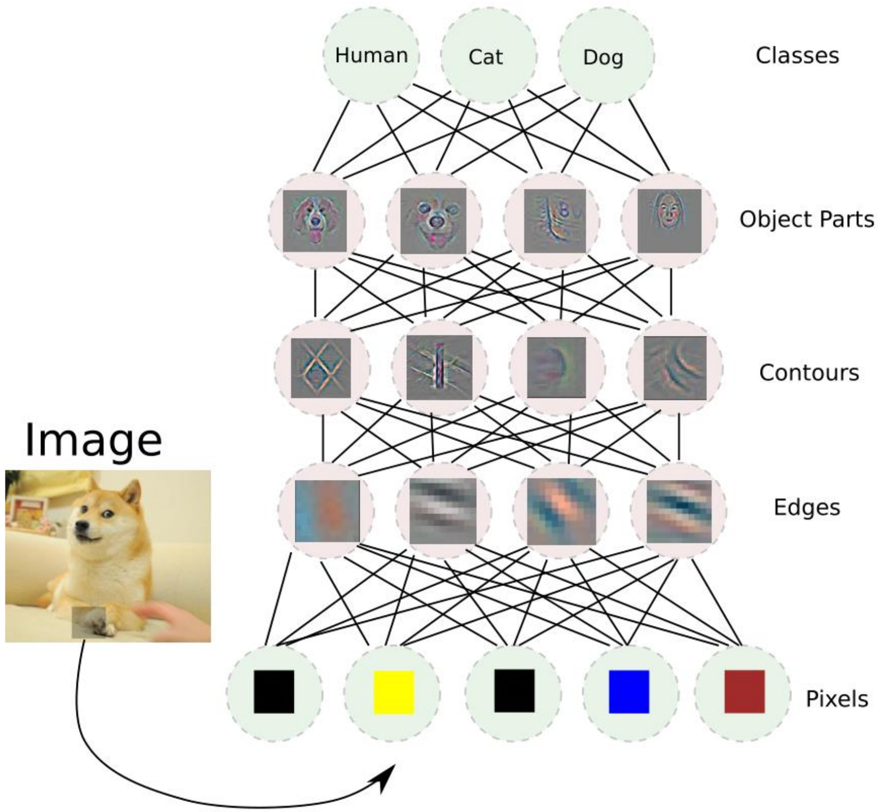
Speech recognition


Reasoning in games

# End-to-end learning

**Deep learning** learns features from raw data

- Multiple layers of abstractions
- **End-to-end learning:** joint optimization of a **single loss function**

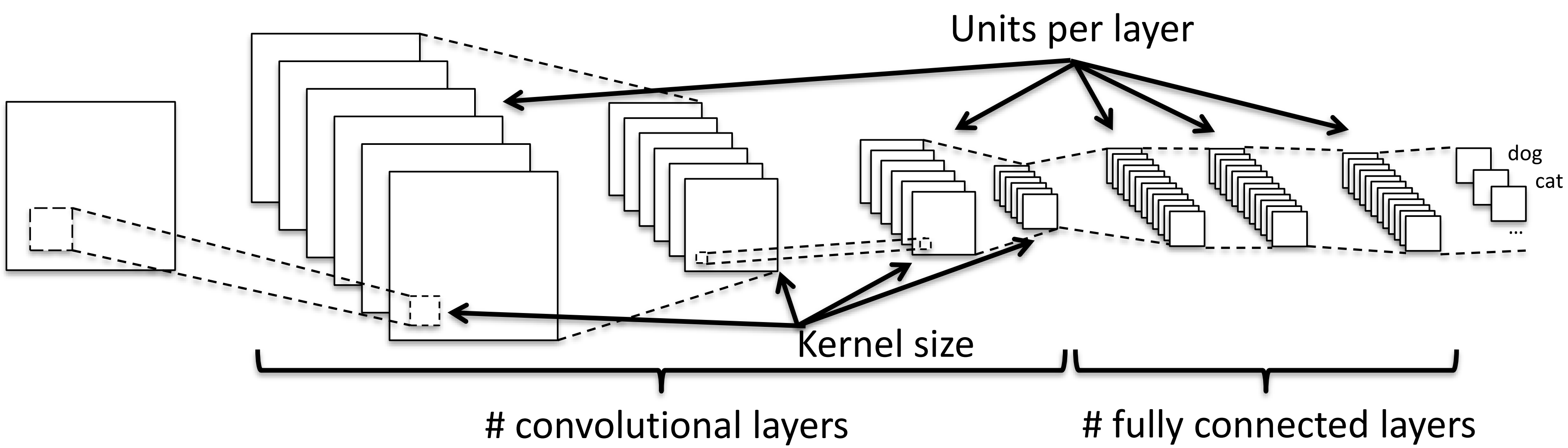



Visualizations of network activations taken from Zeiler [2014]

# One Problem of Deep Learning

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

– Architectural hyperparameters

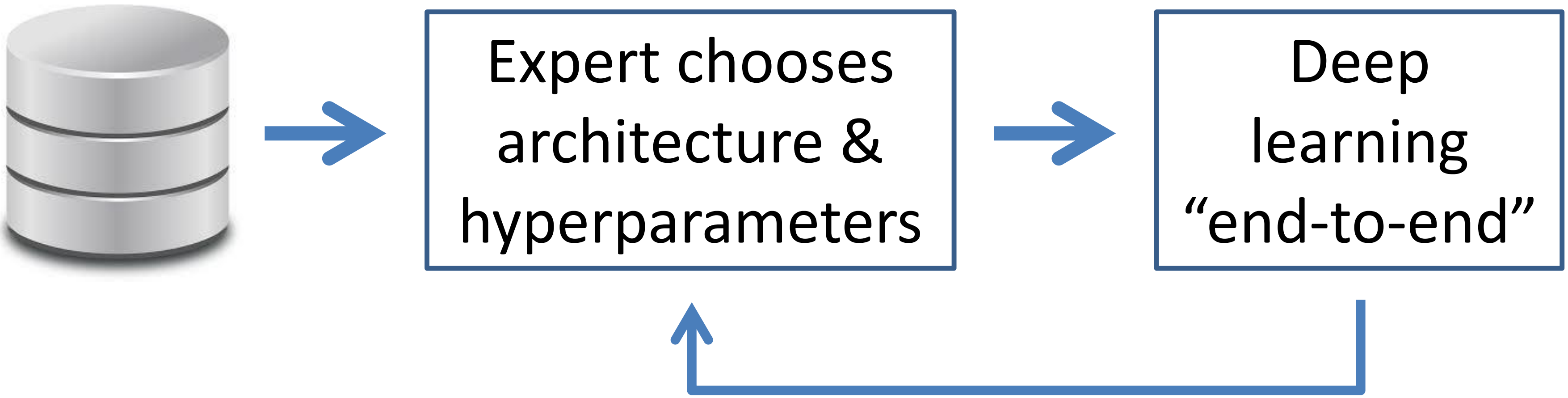



– Optimization algorithm, learning rates, momentum, batch normalization, batch sizes, dropout rates, weight decay, data augmentation, …

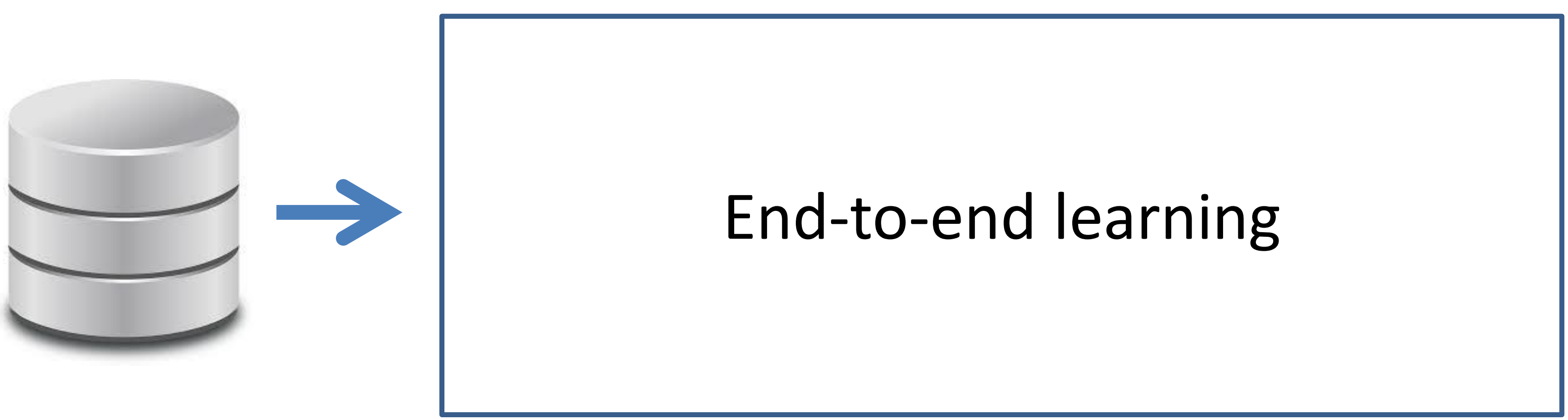
→ **Easily 20-50 design decisions**

# Deep Learning and AutoML

## Current deep learning practice

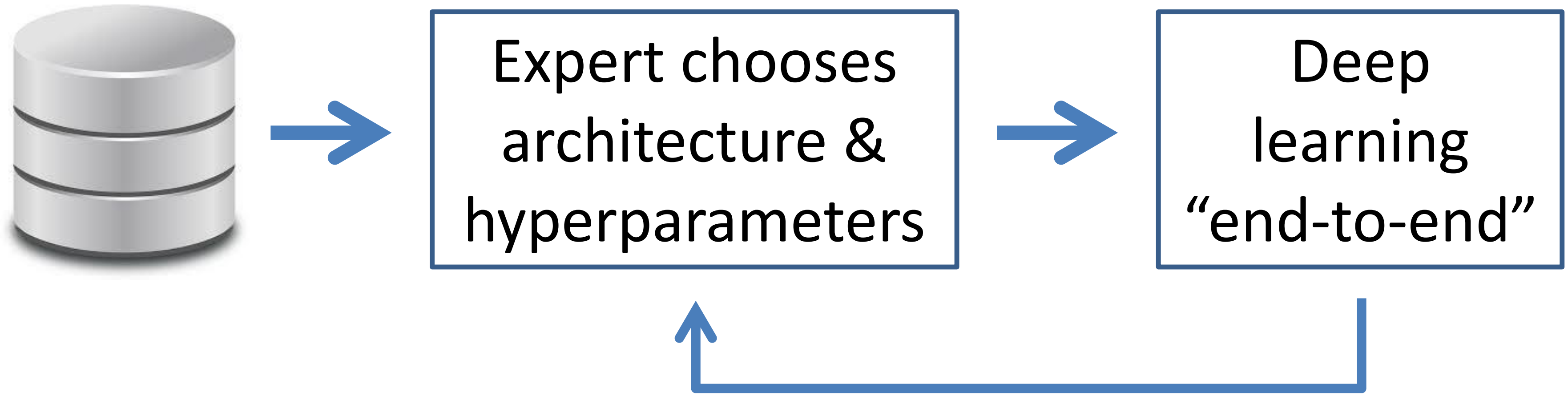



## AutoML: true end-to-end learning

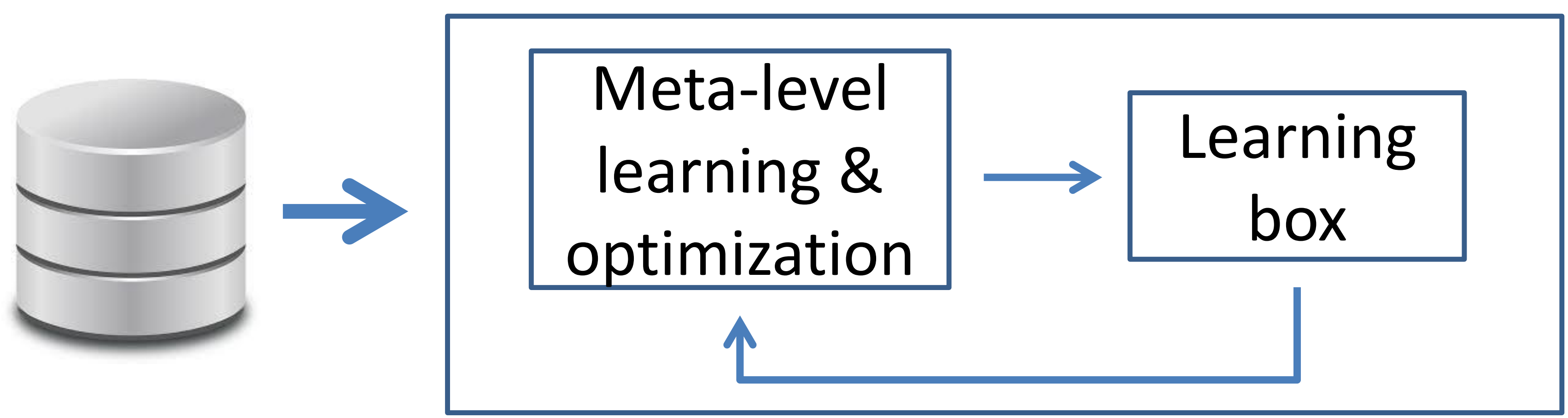


# Deep Learning and AutoML

## Current deep learning practice

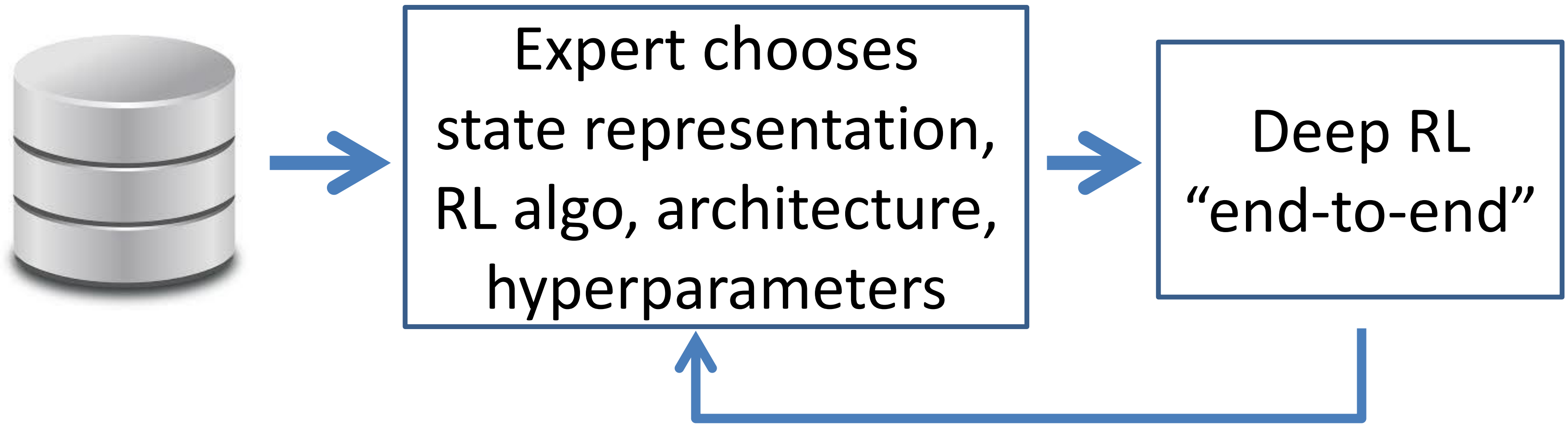



## AutoML: true end-to-end learning

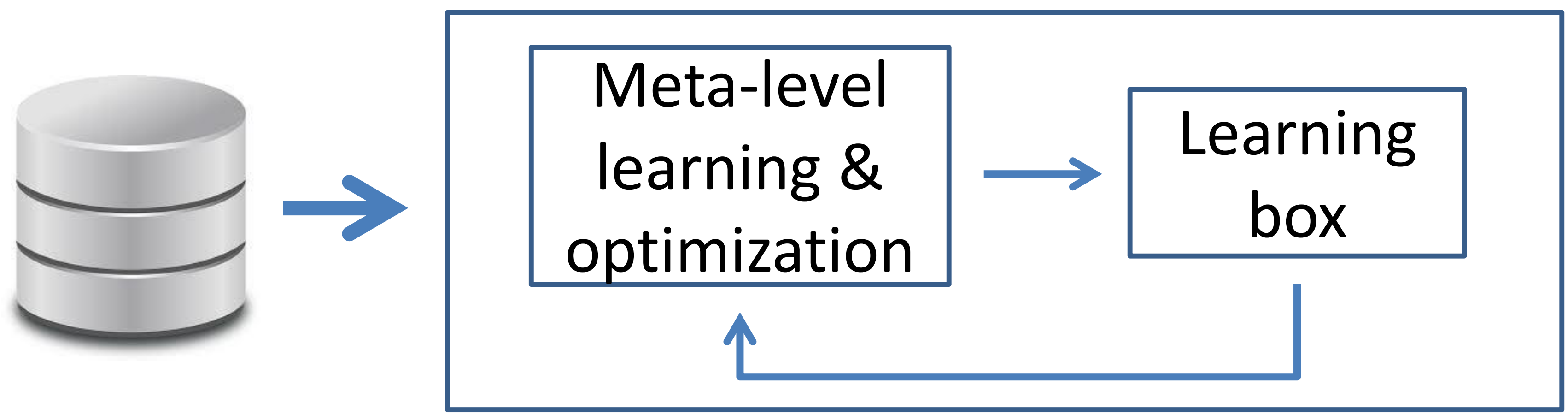


# Deep Reinforcement Learning and AutoML

## Current deep RL practice



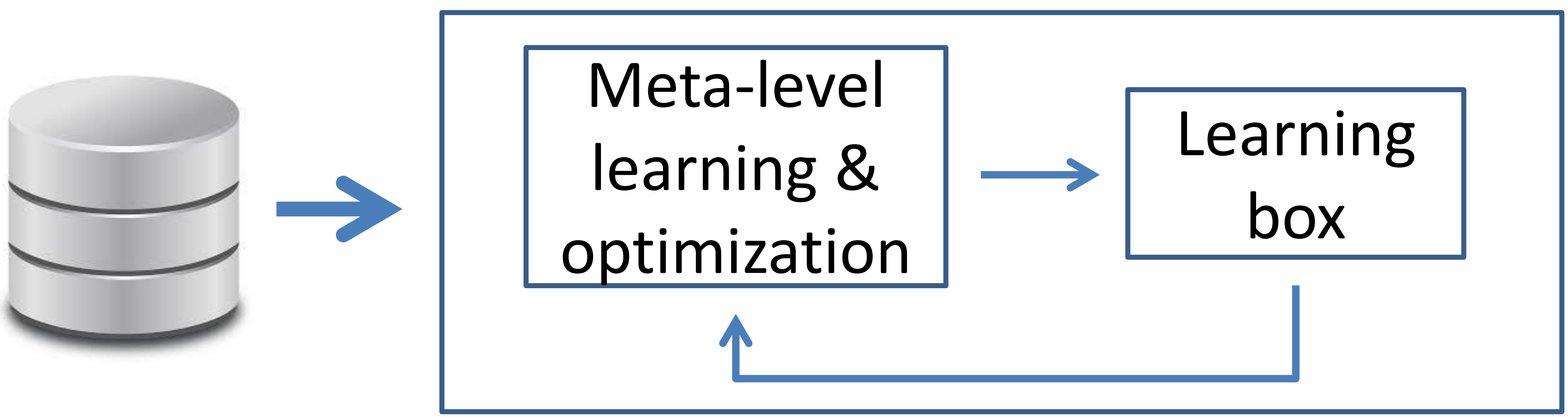

## AutoML: true end-to-end learning



# Overview

- Part 1: AutoML as Blackbox Optimization
- Part 2: Speeding up AutoML
- Part 3: “Auto-RL” for Learning to Design RNA

## AutoML: true end-to-end learning

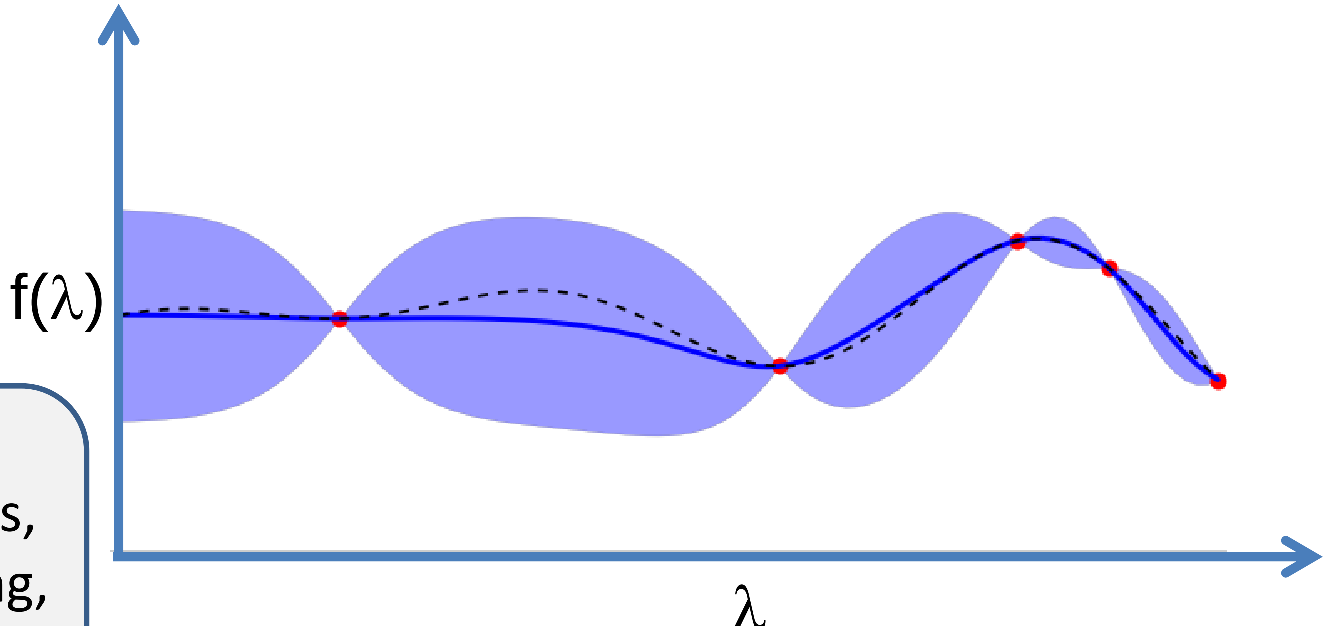


# Benchmark for Progress: AutoML Challenge

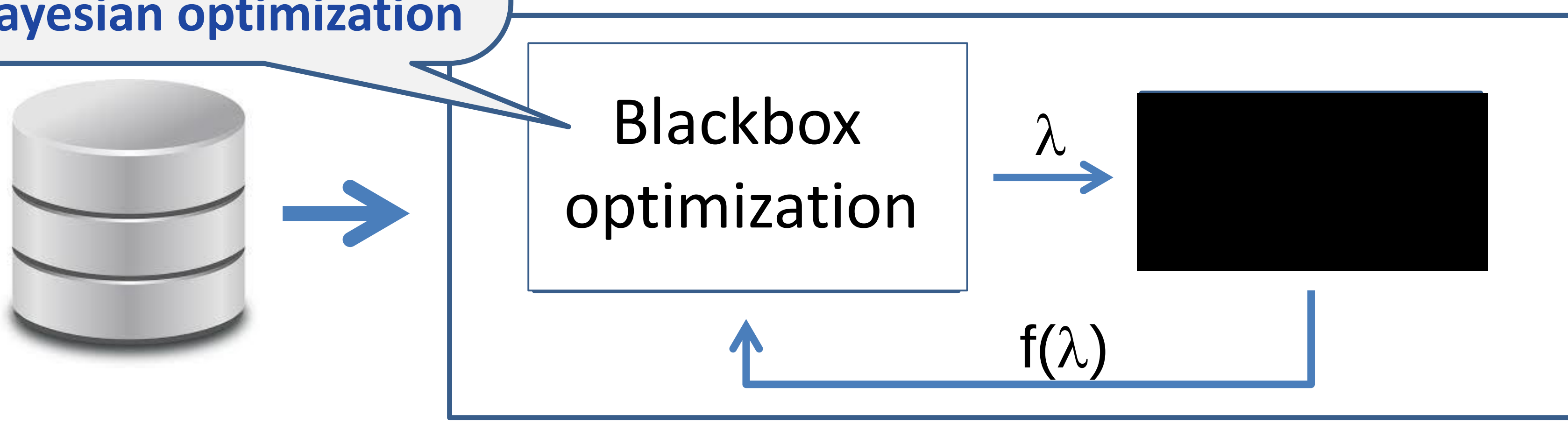
UNI FREIBURG

- **Large-scale challenge run by ChaLearn & CodaLab**
  - 17 months, 5 phases with 5 new datasets each (2015-2016)
  - 2 tracks: code submissions / Kaggle-like human track
- **Code submissions: true end-to-end learning necessary**
  - Get training data, learn model, make predictions for test data
  - 1 hour end-to-end
- **25 datasets from wide range of application areas**
  - Already featurized
  - Inputs: features X, targets y

# AutoML as Blackbox Optimization



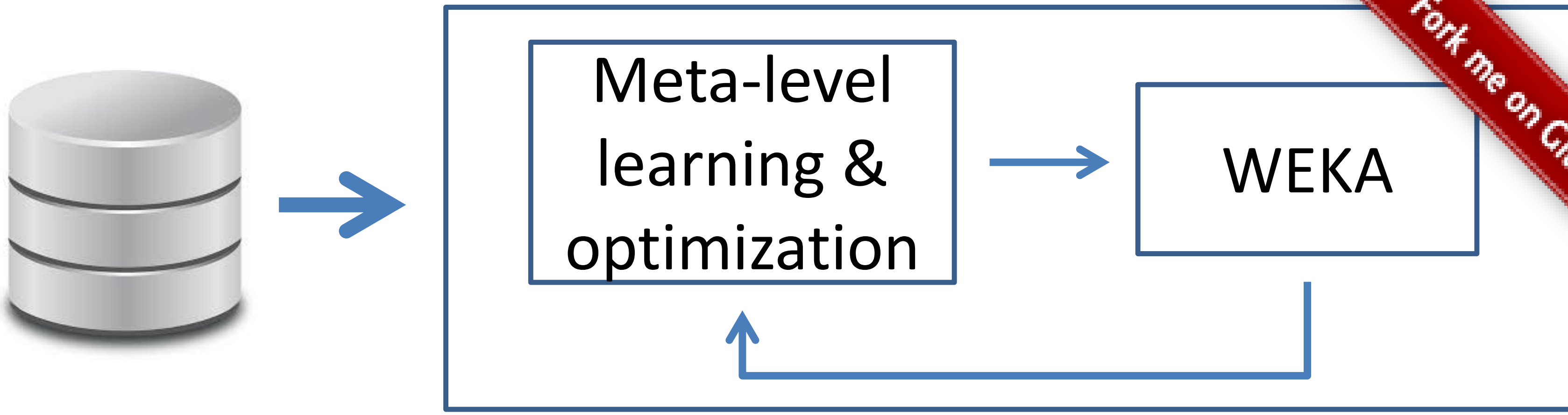

Random search,
evolutionary methods,
reinforcement learning,
…
**Bayesian optimization**



# AutoML System 1: Auto-WEKA

[Thornton, Hutter, Hoos, Leyton-Brown, KDD 2013; Kotthoff e… MLR 2016]

Fork me on GitHub




– **Parameterize ML framework: WEKA** [Witten et al, 1999-current]
  - 27 base classifiers (with up to 10 hyperparameters each)
  - 2 ensemble methods; in total: 786 hyperparameters

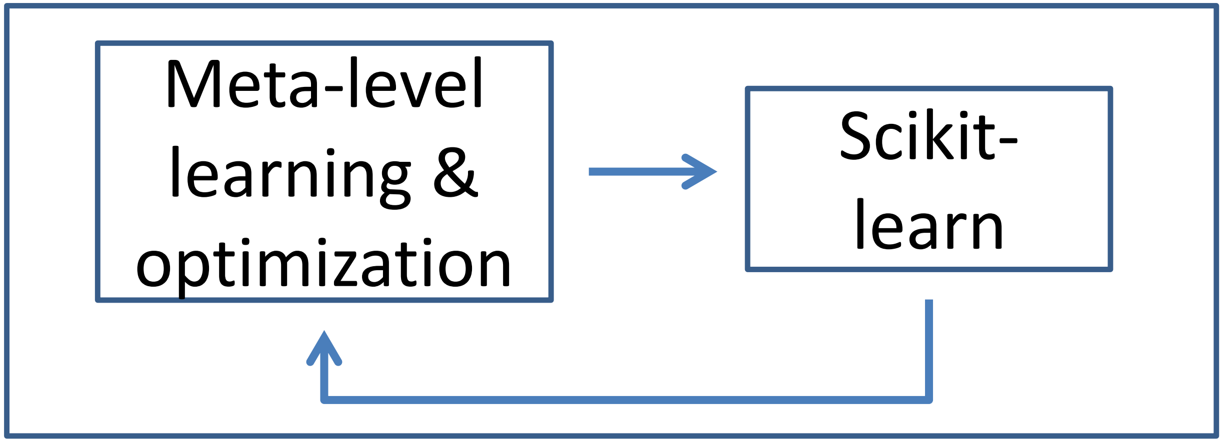
– Optimize **CV performance** by Bayesian optimization (SMAC)
  - Only evaluate more folds for good configurations
    – 5x speedups for 10-fold CV

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

Available in WEKA package manager; ≈400 downloads/week

# AutoML System 2: Auto-sklearn

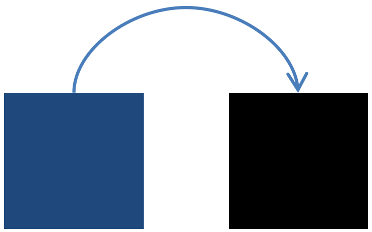
[Feurer, Klein, Eggensperger, Springenberg, Blum, Hutter; NIPS 2015]




- Optimize CV performance by SMAC

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

- **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 Bayesian optimization evaluated
  - Efficiently re-uses its data; improves robustness

# Auto-sklearn: Ready for Prime Time

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 exper**
  - **Final two rounds: won both tracks**

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

Used by 54 | Watch 213 | Star 4k | Fork 764

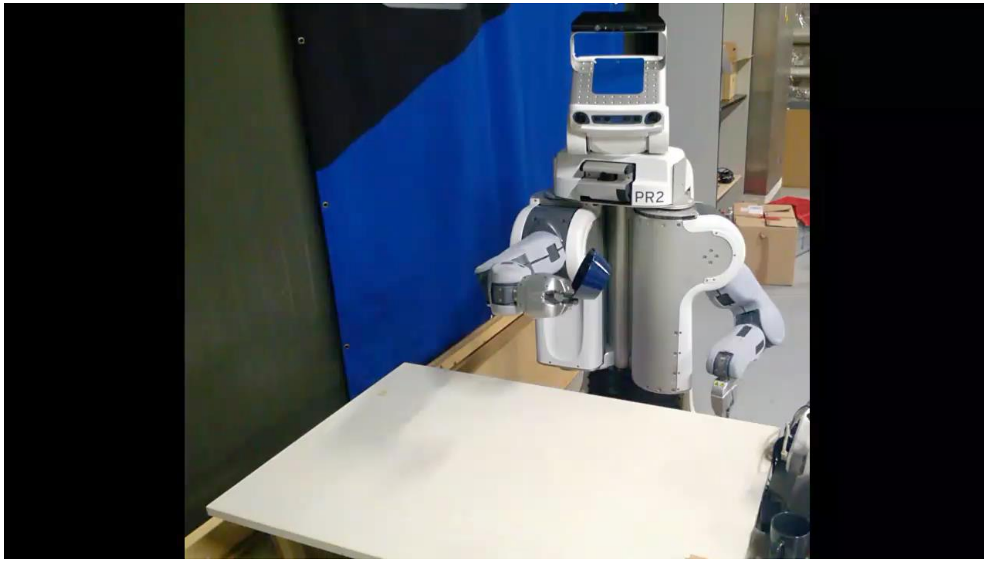
- Trivial to use, open source (BSD):

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

# Example Application: Robotic Object Handling

- Collaboration with Freiburg's robotics group

- Binary classification task for object placement: **will the object fall over?**



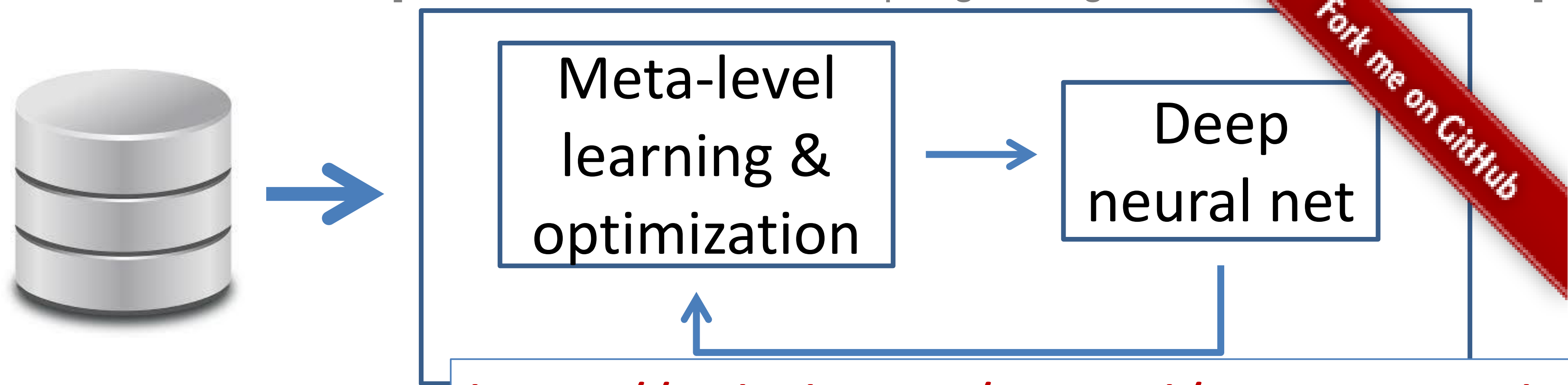

Video credit: Andreas Eitel

- Dataset
  - Based on BigBIRD and YCB Object and Model Set
  - 30000 data points
  - 50 features -- manually defined [BSc thesis, Hauff 2015]

- Performance
  - Strong BSc student, 3 months with Caffe: **2% error rate**
  - **Auto-sklearn: 0.6% error rate** (within 30 minutes)

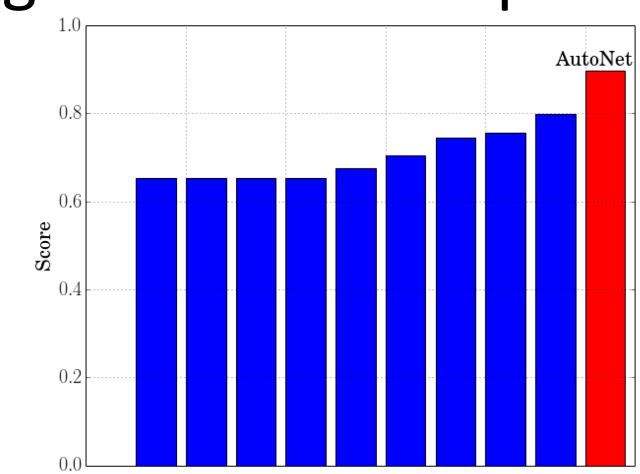
# AutoML System 3: Auto-Net

[Mendoza, Klein, Feurer, Springenberg & Hutte… oML 2016]




https://github.com/automl/Auto-PyTorch

- **Joint Architecture & Hyperparameter Optimization**
- Auto-Net won several datasets against human experts
  - E.g., Alexis data set:
    - 54491 data points,
      5000 features, 18 classes
  - **First automated deep learning system to win a ML competition data set** against human experts



# Since Then: Many Works on Architecture Search

- RL & Evolution for NAS by Google Brain [Quoc Le's group, '16-'18]
  - New state-of-the-art results for CIFAR-10, ImageNet, Penn Treebank, Cityscapes
  - Large computational demands
    - **800 GPUs for 2 weeks**
    - **12.800 architectures evaluated**
  - Hyperparameter optimization only as postprocessing

- **Recent work aims for efficiency**
  - Network morphisms [Chen et al, '16; Cai et al, '17&'18; Elsken et al, '17&18]
  - Weight sharing [Pham et al,'18; Bender et al, '18; Liu et al, '19]
  - Multi-fidelity optimization [Klein et al, '16; Li et al, '18; Falkner et al, '18]

# Outline

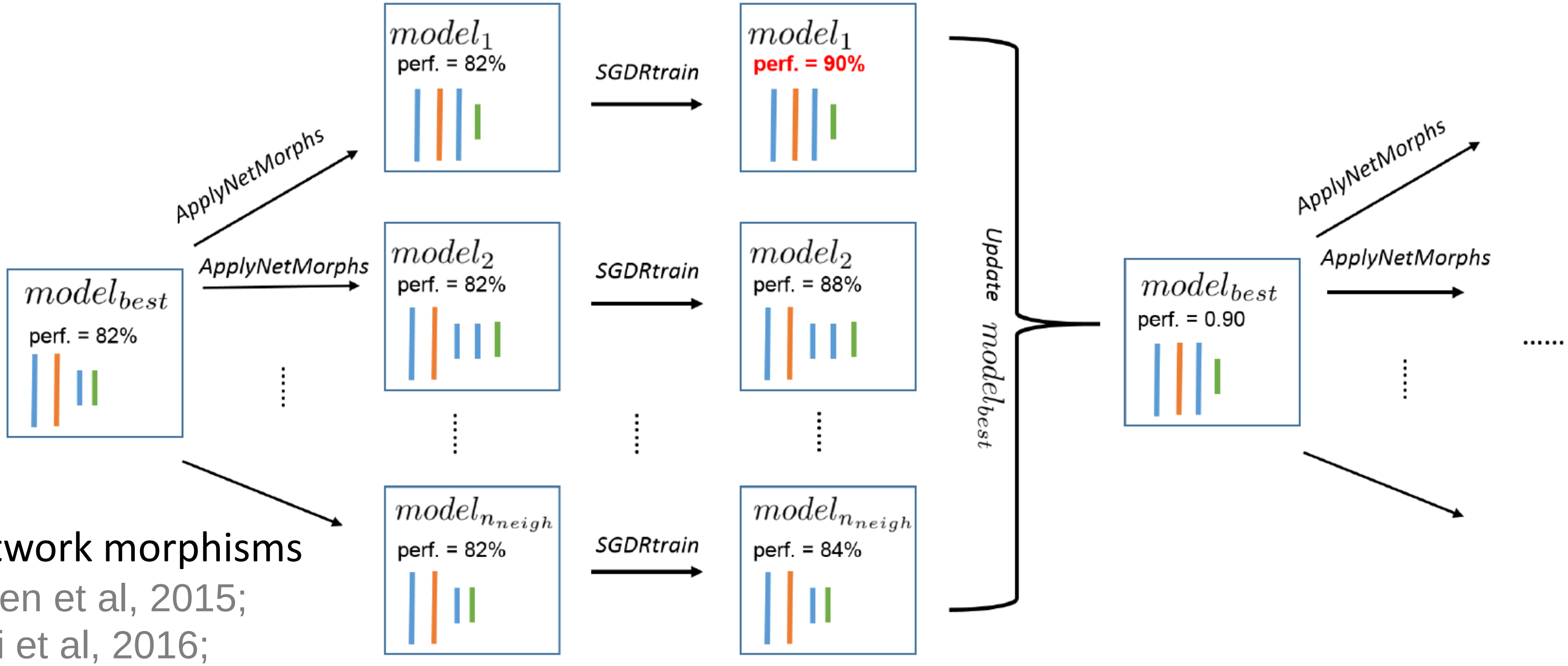
- Part 1: AutoML as Blackbox Optimization

- Part 2: Speeding up AutoML
  - Fast Neural Architecture Search via Network Morphisms
  - Fast Neural Architecture Search via Weight Sharing: DARTS
  - Fast Hyperparameter Optimization via Multi-fidelity Methods

- Part 3: “Auto-RL” for Learning to Design RNA

# Fast Architecture Search via Network Morphisms

[Elsken, Metzen & Hutter, MetaLearn 2017]




Network morphisms
[Chen et al, 2015;
Wei et al, 2016;
Cai et al, 2017]

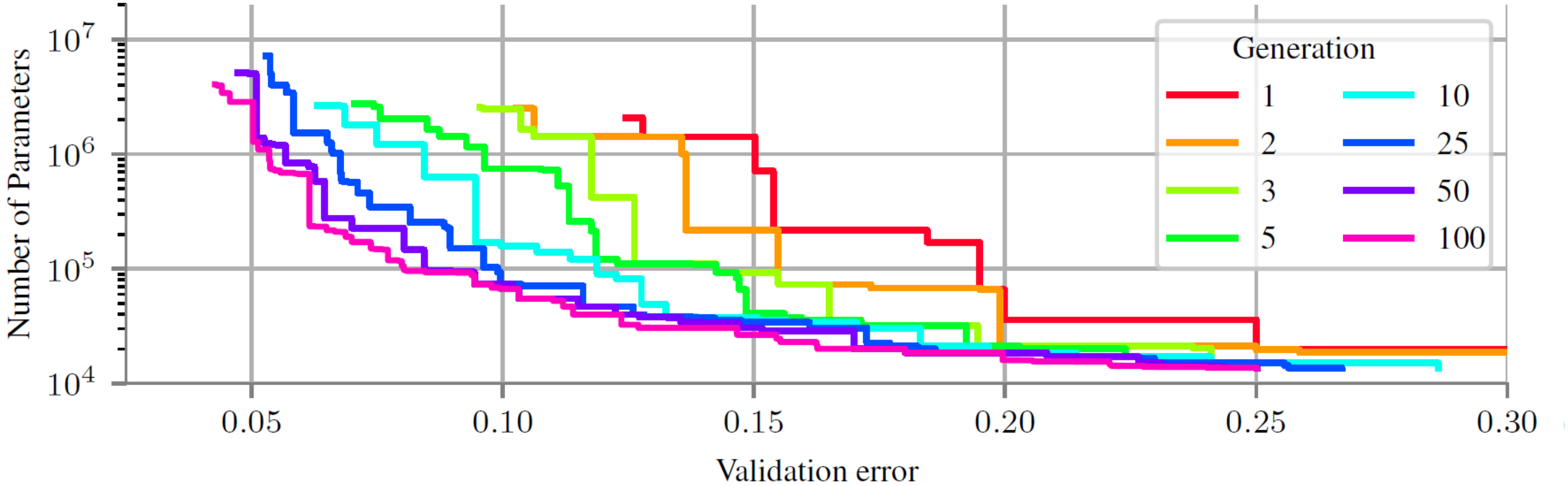
Cosine annealing
[Loshchilov & Hutter, 2017]

Result: enables **architecture search in 12 hours on 1 GPU**

# Efficient Multi-objective Architecture Search

[Elsken, Metzen & Hutter, ICLR 2019]

- To trade off network size vs. error, maintain a **Pareto front** of the **2 objectives**

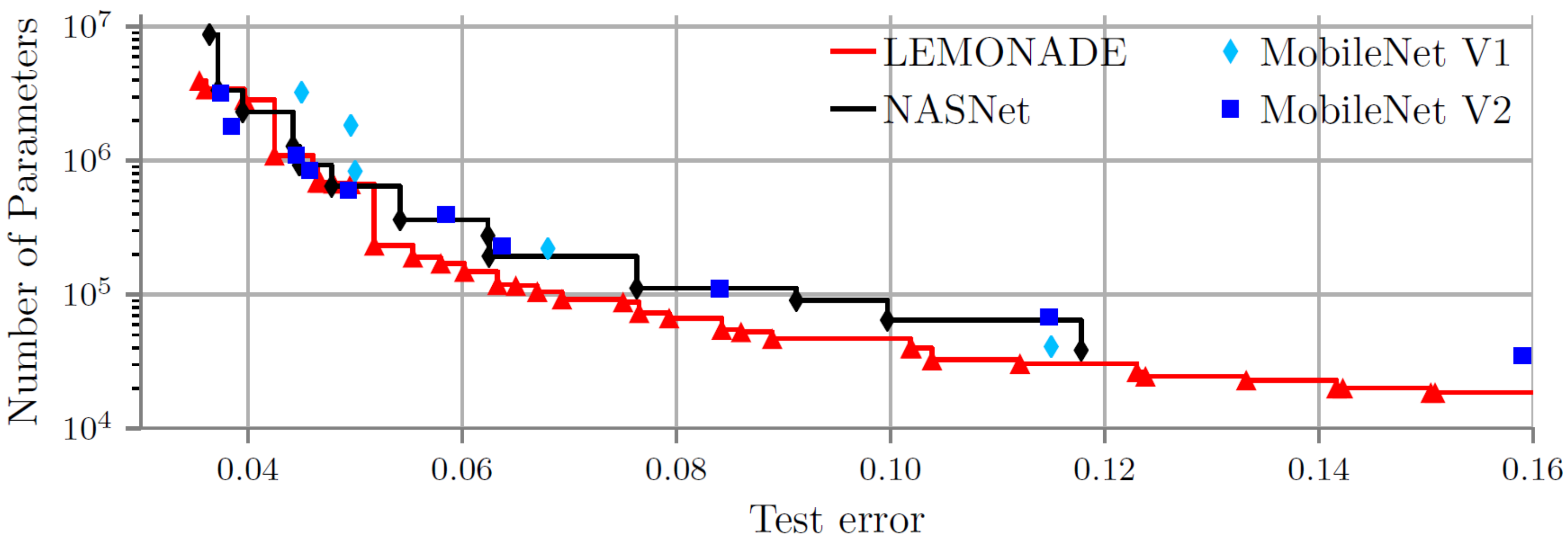



- Evolve a population of Pareto-optimal architectures over time
- **LEMONADE**: Lamarckian Evolution for Multi-Objective Neural Architecture DEsign
  – Weight inheritance through approximate morphisms
  – Still cheap: 1 week on 8 GPUs

# Efficient Multi-objective Architecture Search

[Elsken, Metzen & Hutter, ICLR 2019]

- **Comparison to existing mobile-sized networks**
  - Using the same training pipeline
  - Better than manually-constructed mobile architectures
  - Better results than NASNet and 35x faster search (56 vs. 2000 GPU days)

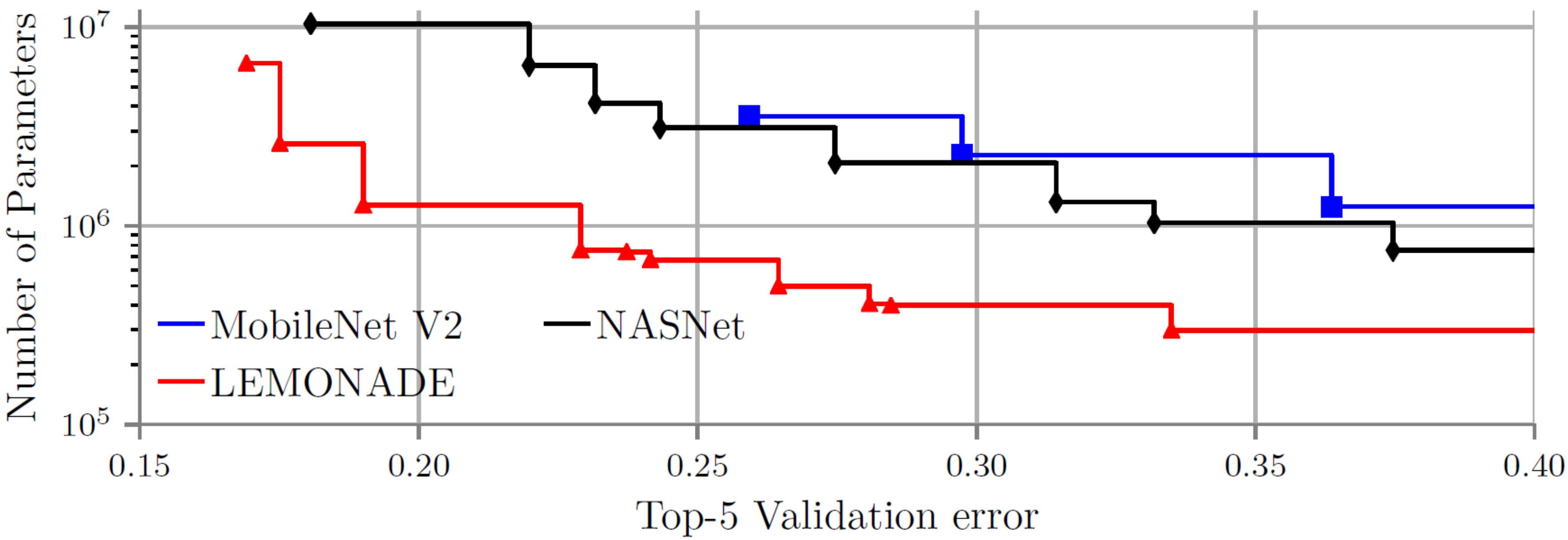



CIFAR-10

# Efficient Multi-objective Architecture Search

[Elsken, Metzen & Hutter, ICLR 2019]

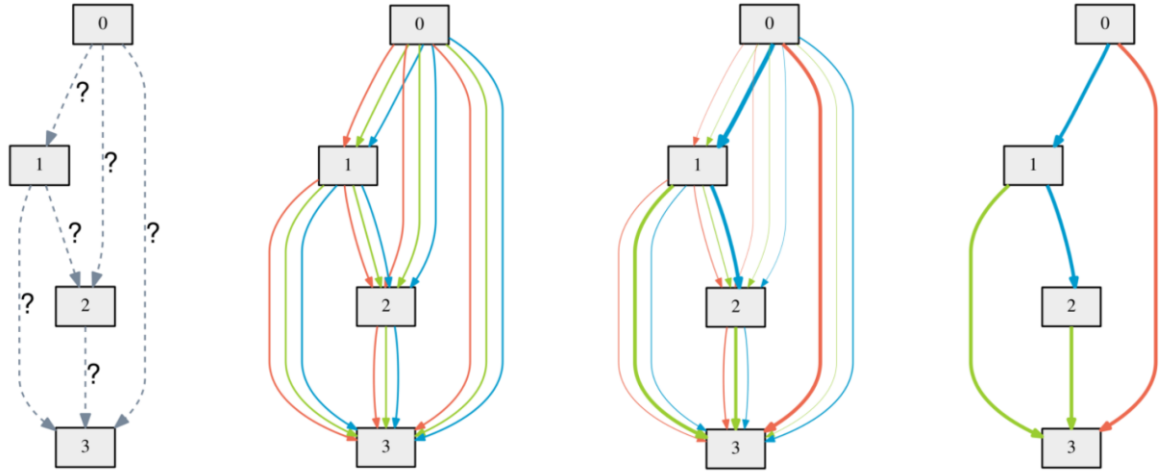
- **Comparison to existing mobile-sized networks**
  - Using the same training pipeline
  - Better than manually-constructed mobile architectures
  - Better results than NASNet and 35x faster search (56 vs. 2000 GPU days)




Transfer to ImageNet 64x64

# Weight Sharing: DARTS

[Liu et al, ICLR 2019]




(a) (b) (c) (d)

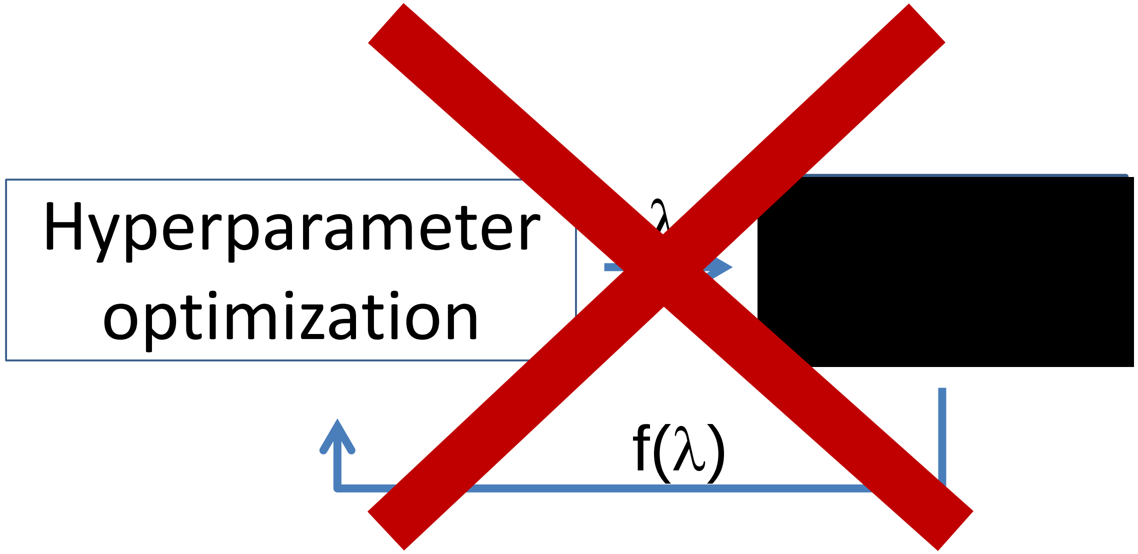
- Relax the discrete NAS problem (a->b)
  - One-shot model with continuous architecture weight α for each operator
  - Combined operator: $\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x)$
- Solve a bi-level optimization problem (c)

$$\min_{\alpha} \quad \mathcal{L}_{val}(w^*(\alpha), \alpha)$$
$$\text{s.t.} \quad w^*(\alpha) = \text{argmin}_w \; \mathcal{L}_{train}(w, \alpha)$$

- In the end, discretize to obtain a single architecture (d)

# Speeding up Hyperparameter Optimization

**Too slow for big data**

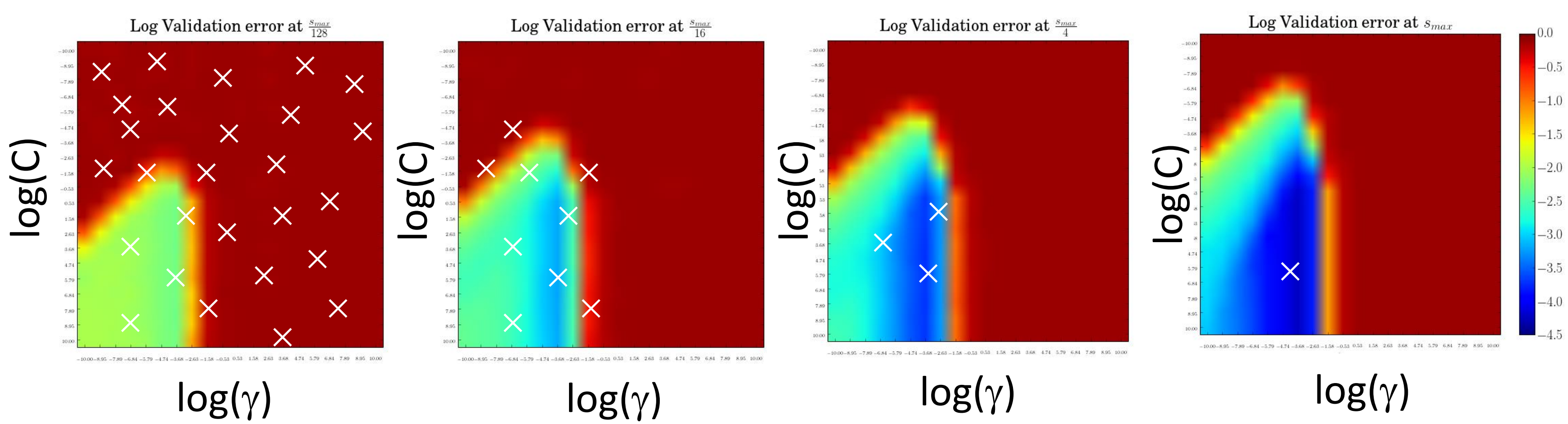



## → Multi-fidelity methods

In a nutshell: use cheaper-to-evaluate approximations of the blackbox,
performance on which correlates with the real blackbox

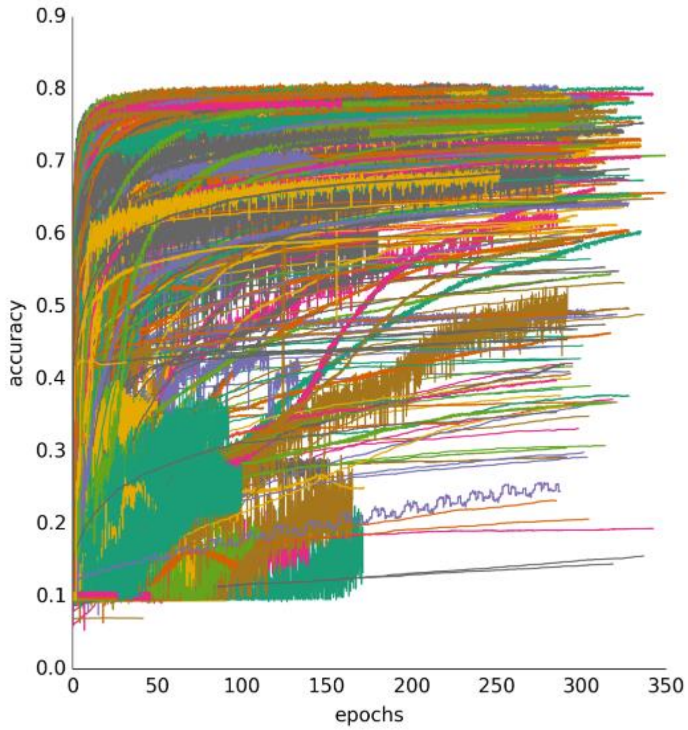
# Using Cheap Approximations of the Blackbox

- **One possible approximation: use a subset of the data**
  - E.g.: 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]

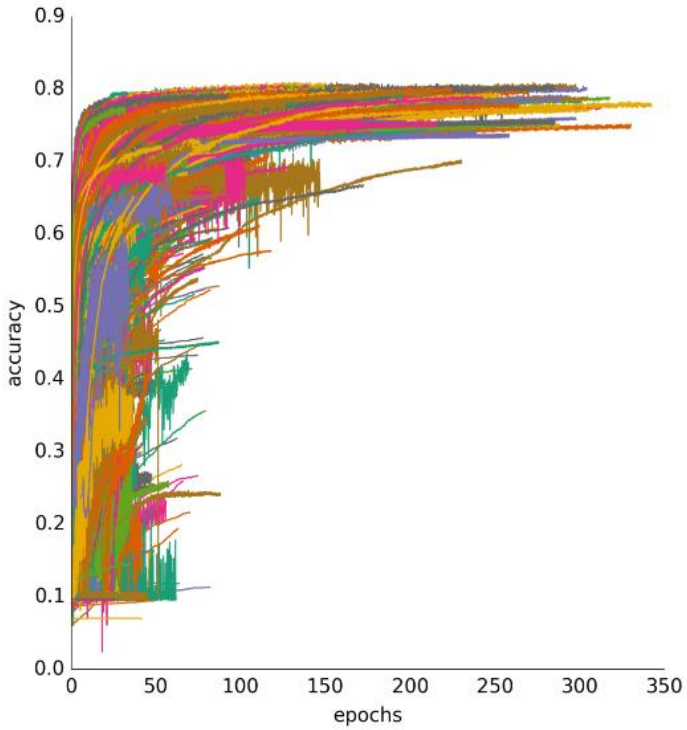


# Using Cheap Approximations of the Blackbox

- **One possible approximation: use less epochs of SGD**
  - [Swersky et al, arXiv 2014; Domhan et al, IJCAI 2015]




All learning curves




With predictive termination

# Using Cheap Approximations of the Blackbox

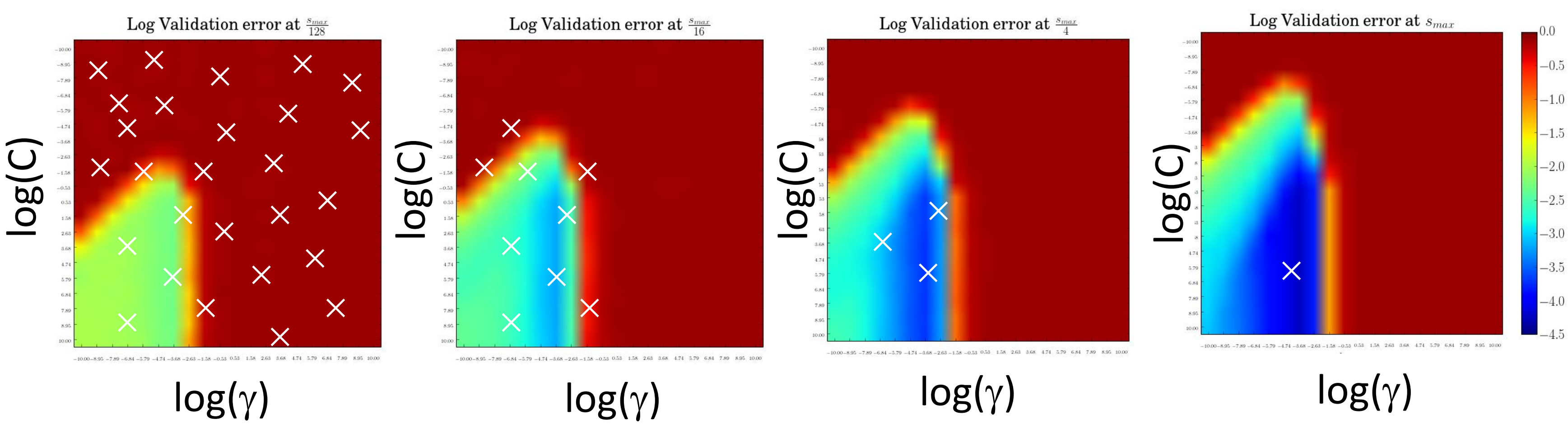
- **Cheap approximations exist in many applications**
  - Subset of data
  - Fewer epochs of iterative training algorithms (e.g., SGD)
  - Downsampled images in object recognition
  - Shorter MCMC chains in Bayesian deep learning
  - Fewer trials in deep reinforcement learning

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

# How to Exploit Cheap Approximations

- **Bayesian optimization** [Klein et al, 2017; Kandasamy et al, 2017]
  - Fit a predictive model f(λ,b) to predict performance as a function of hyperparameters λ and budget b
  - Extrapolate performance from small to large budgets

- **Simpler approach:**
  - Successive Halving [Jamieson & Talwalkar, AISTATS 2015]
  - Hyperband [Li et al, ICLR 2017]



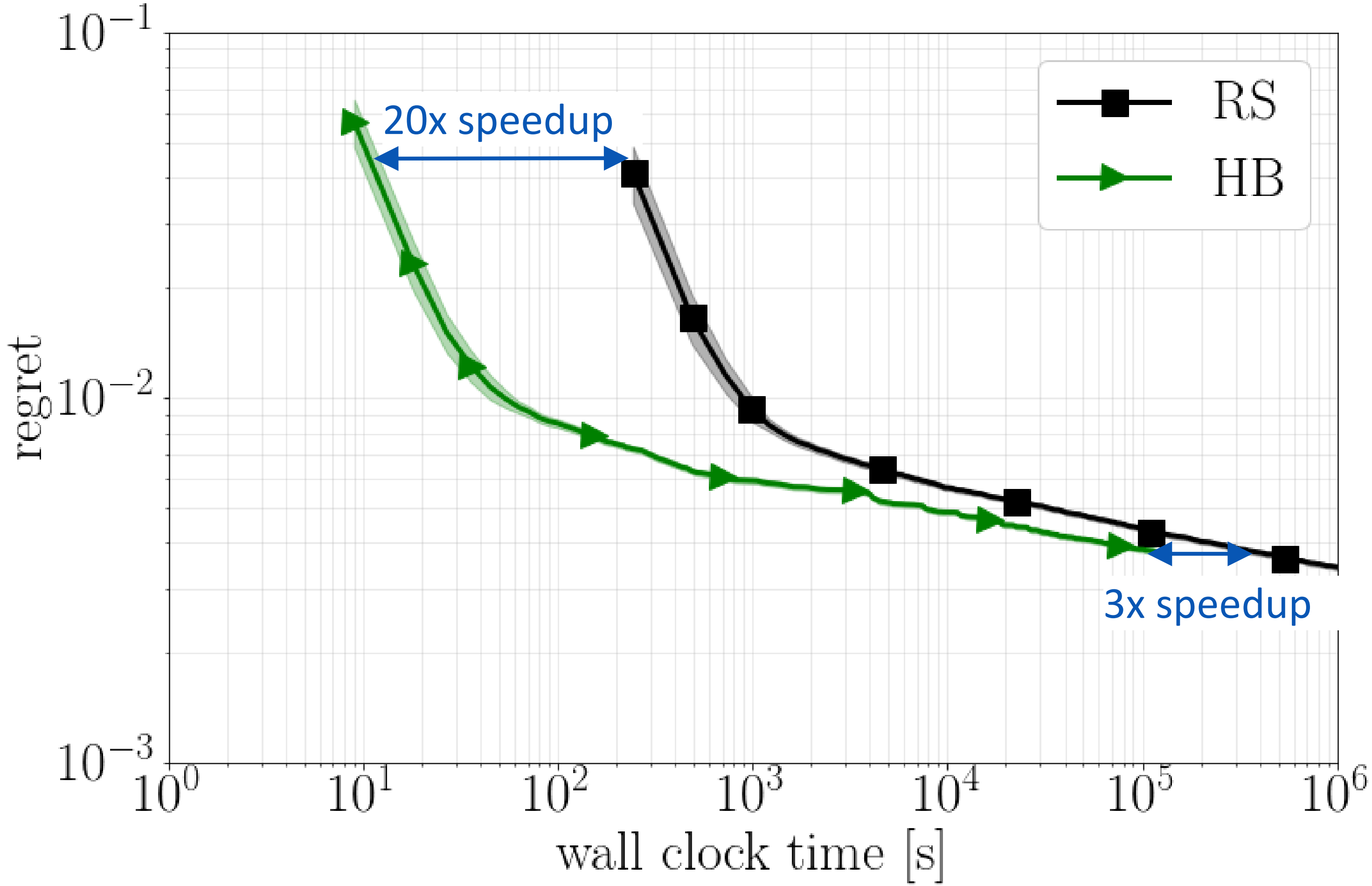
# BOHB: Bayesian Optimization & Hyperband

[Falkner, Klein & Hutter, ICML 2018]

- **Bayesian optimization**
  - for choosing the configuration to evaluate
- **Hyperband**
  - for deciding how to allocate budgets

- **Advantages**
  - All the 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
  - But also strong final performance (due to Bayesian optimization)

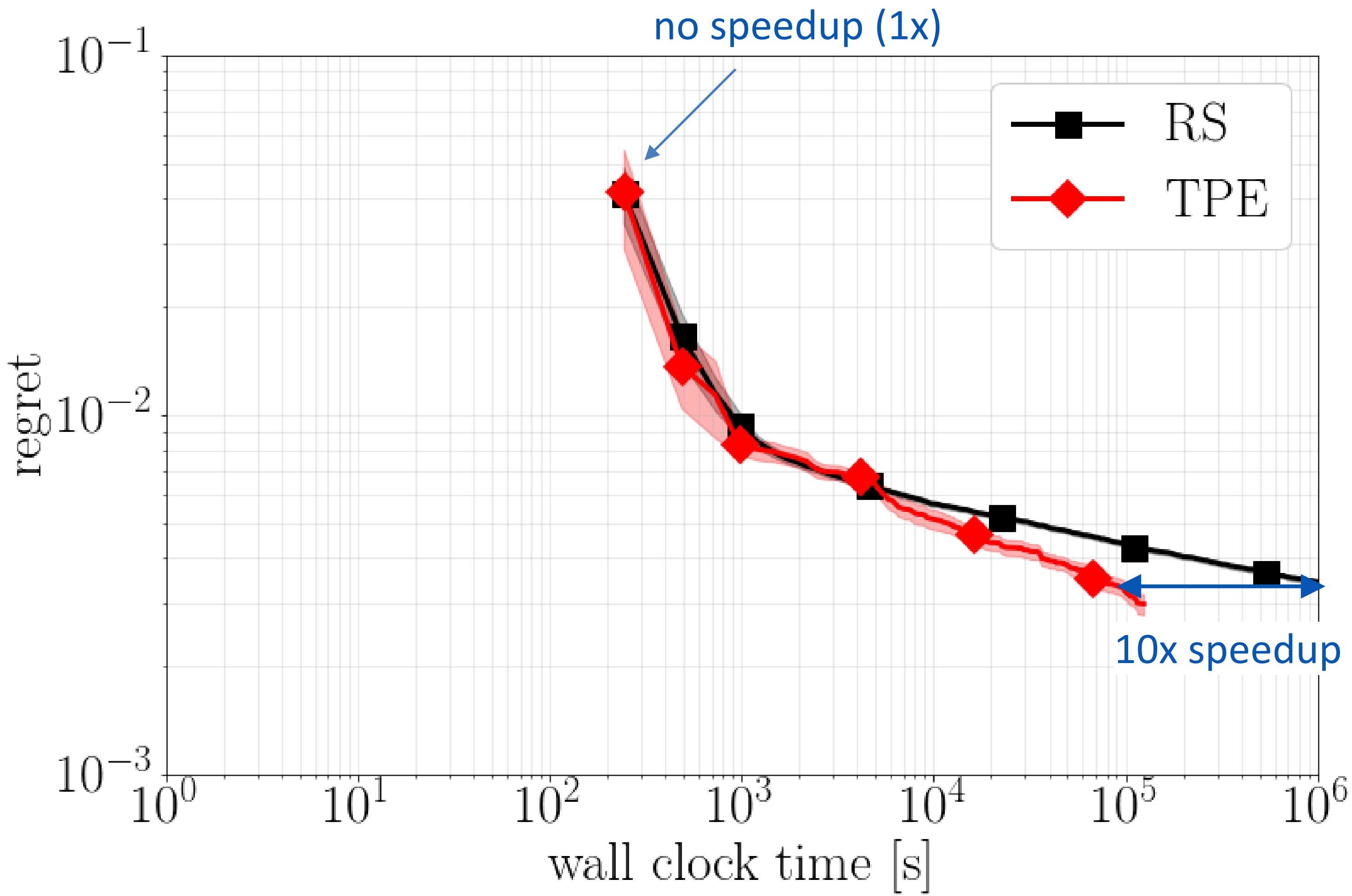
# Hyperband vs. Random Search




Biggest advantage: much improved **anytime performance**

Auto-Net on dataset adult

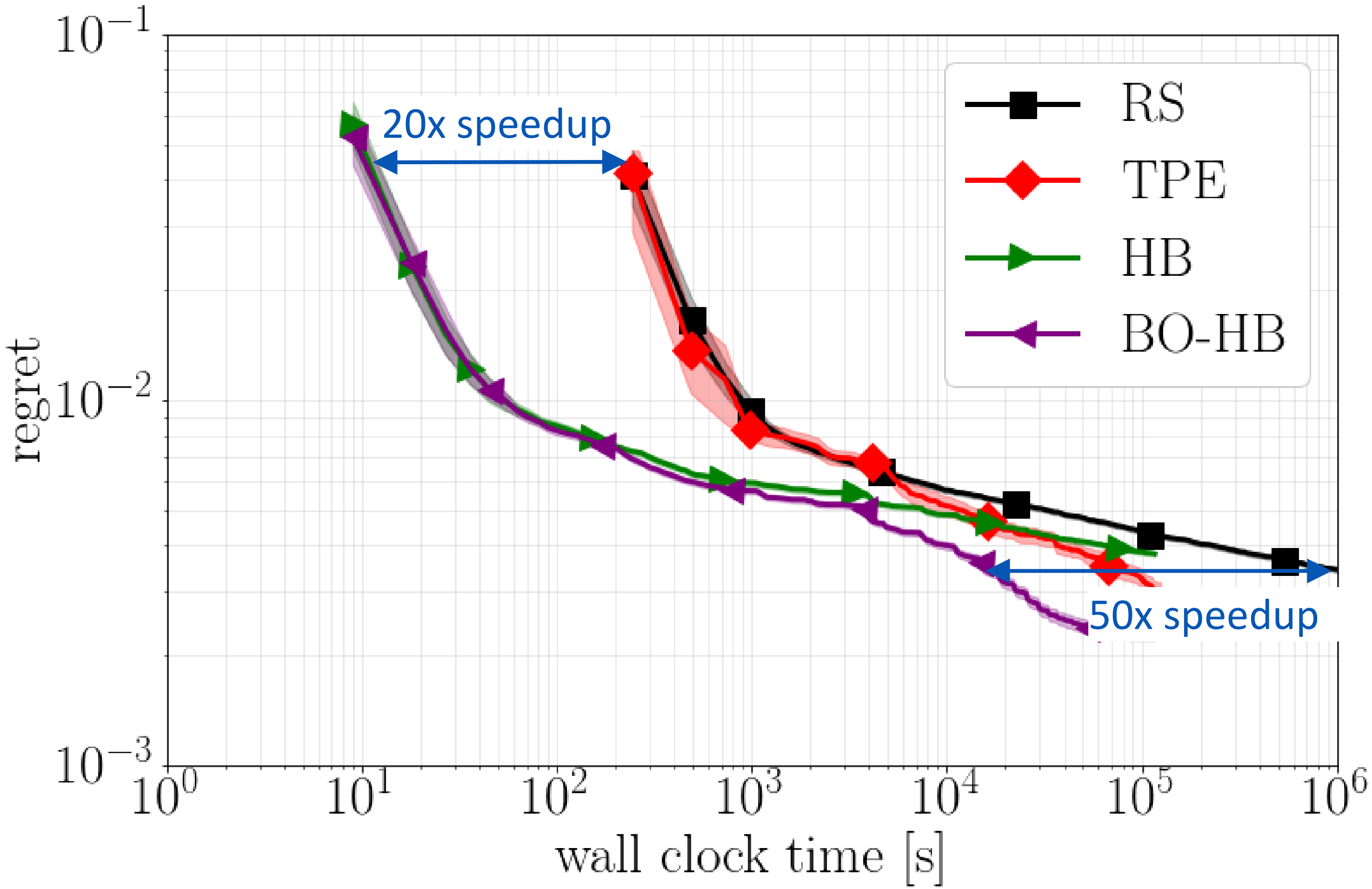
# Bayesian Optimization vs. Random Search




Biggest advantage: much improved **final performance**

Auto-Net on dataset adult

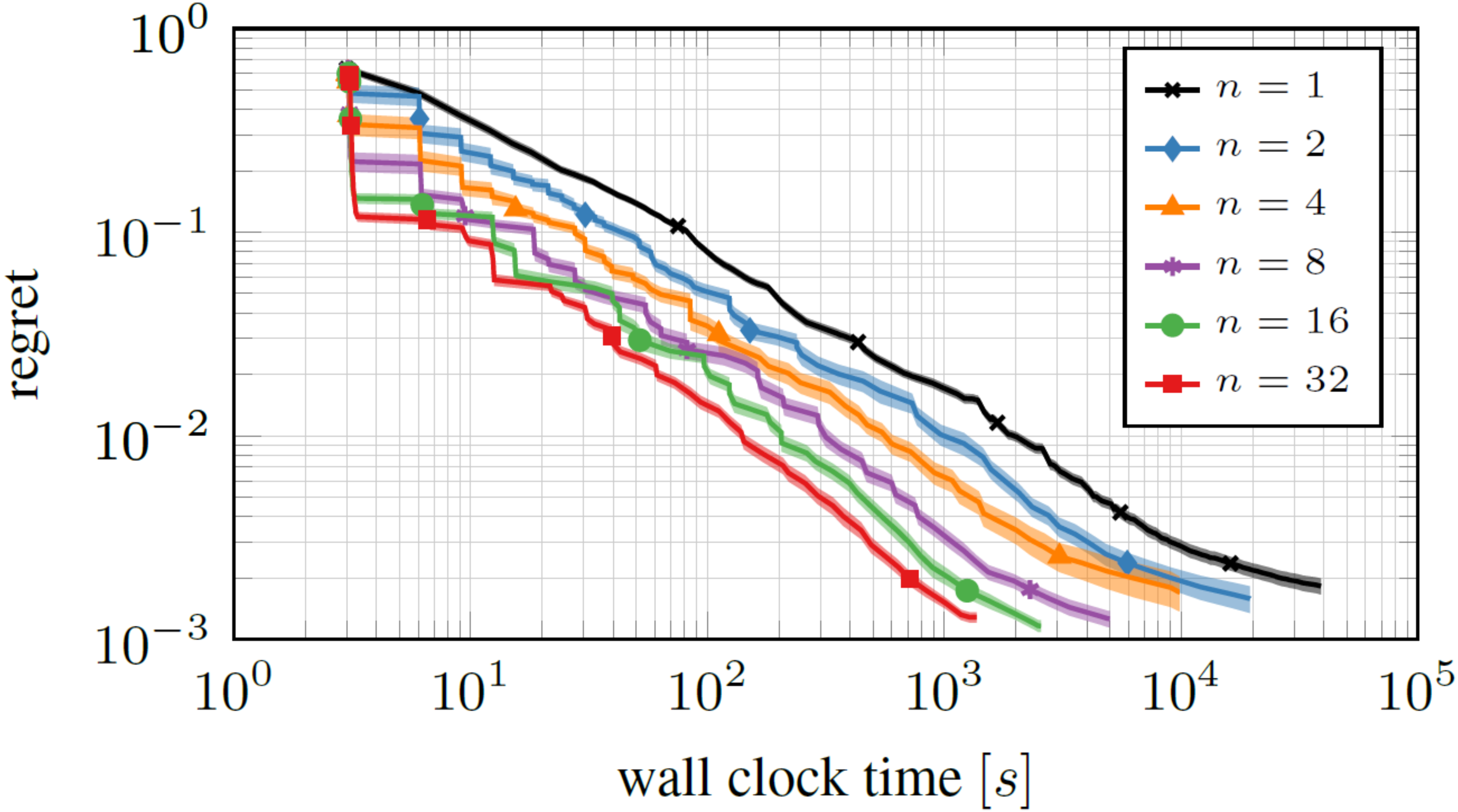
# Combining Bayesian Optimization & Hyperband




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

Auto-Net on dataset adult

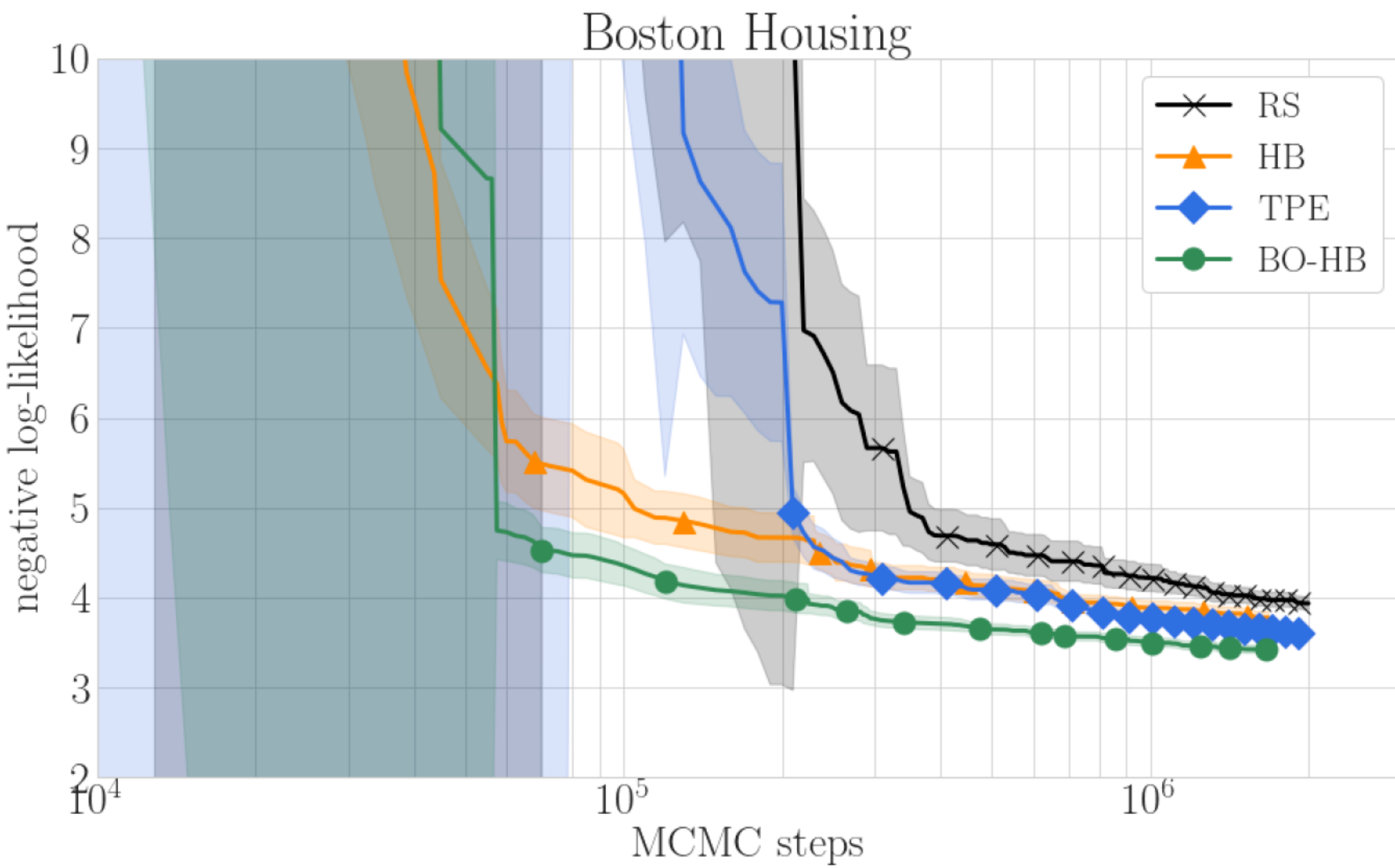
# Almost Linear Speedups By Parallelization




Auto-Net on dataset letter

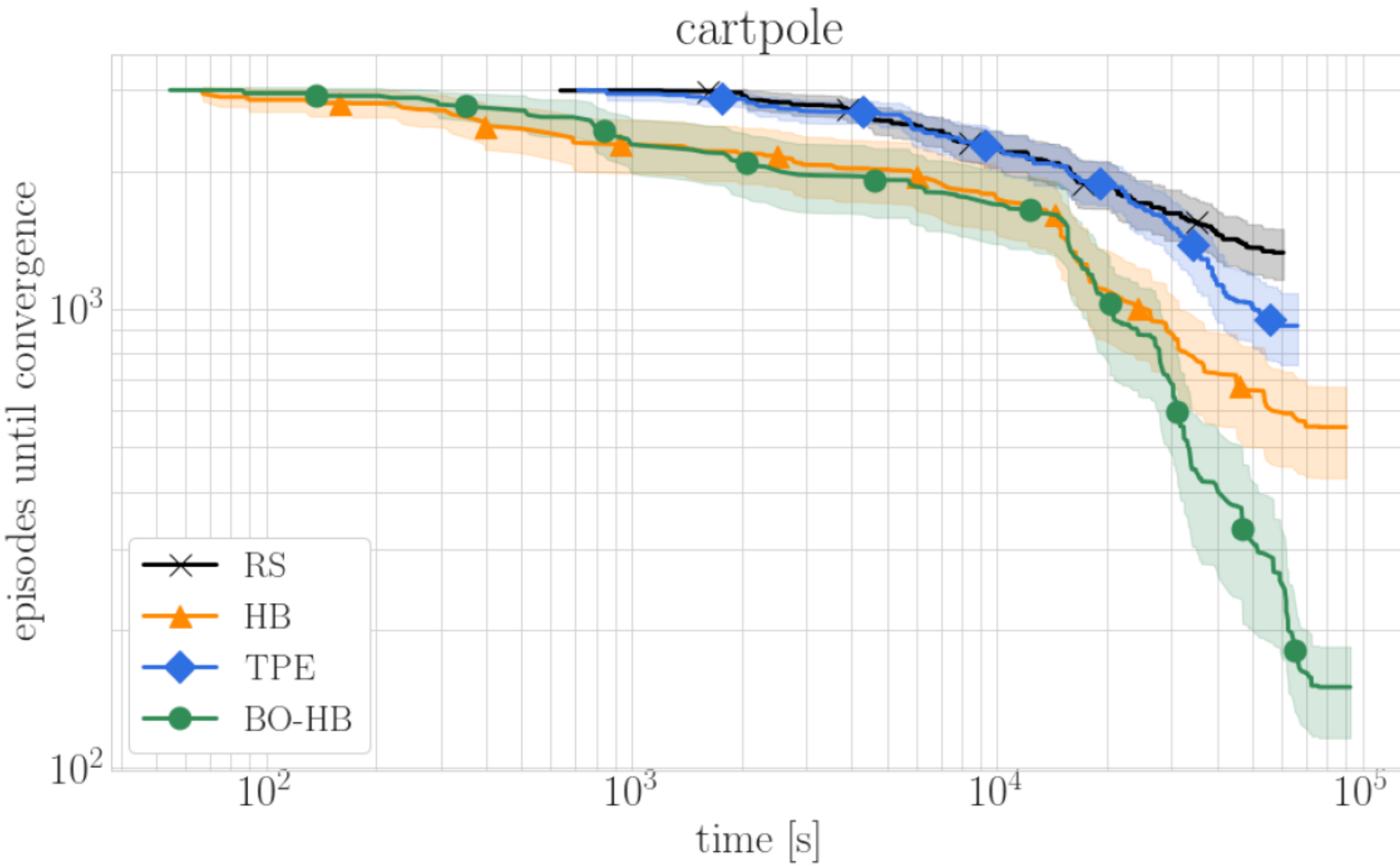
# Application to Bayesian Deep Learning

- **Stochastic Gradient Hamiltonian Monte Carlo**
- Budget: MCMC steps



# Application to Deep Reinforcement Learning

- **Proximal policy optimization** on cartpole benchmark
- Budget: trials (to find a robust policy)



# Application to Second AutoML Challenge

[Feuerer, Eggensperger, Falkner, Lindauer, Hutter; AutoML 2018]

- **Auto-sklearn 2.0**
  - Uses base algorithms from scikit-learn and XGBoost
  - Optimized using BOHB
  - Budgets: dataset size; number of training epochs
  - **More efficient for large datasets than Auto-sklearn 1.0**

- Use **meta-learning across datasets** to warmstart BOHB
  - 16 complementary configurations for the first phase of successive halving pre-selected with SMAC

- **Won the second international AutoML challenge** (2017 –2018)

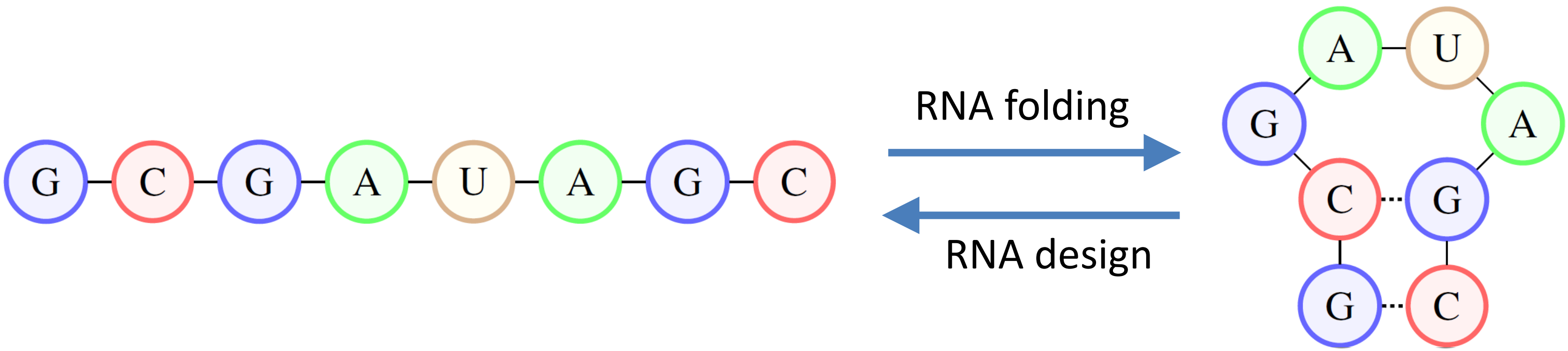
# Outline

- Part 1: AutoML as Blackbox Optimization
- Part 2: Speeding up AutoML
- Part 3: “Auto-RL” for Learning to Design RNA

# The RNA Design Problem

[Stoll, Runge, Falkner & Hutter, ICLR 2019]

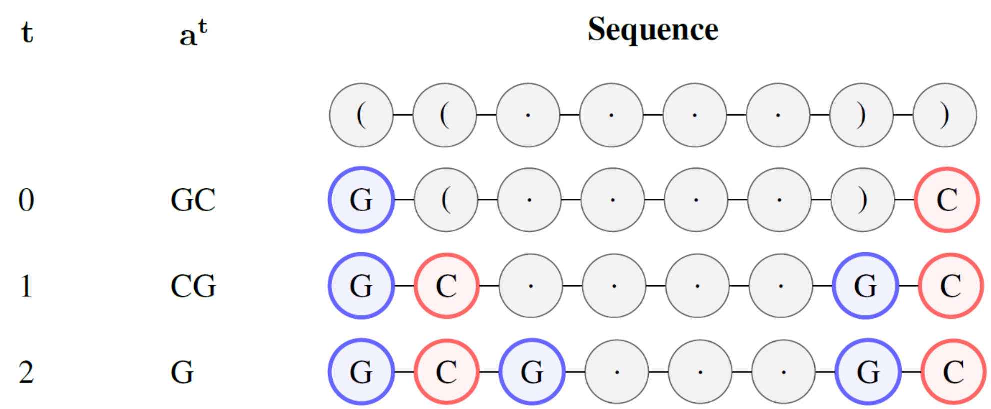
- Background on RNA:
  - Sequence of nucleotides (C, G, A, U)
  - Folds into a secondary structure, which determines its function
  - **RNA design**: find an RNA sequence that folds to a given structure



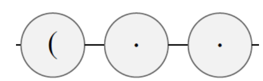

- RNA folding is $O(N^3)$ for sequences of length N
- RNA design is computationally hard
  - Typical approach: generate and test; local search
  - Here: learning a policy network to sequentially design the sequence

# RNA Design as an RL Problem

- **Actions**:
  - Place next nucleotide/ pair of nucleotides




- **State** at time t:
  - Simply a local n-gram centered at step t:




- (Episodic) **reward**:
  - Fold the designed sequence, measure agreement with target

- **Policy network**: maps the state to a probability distribution over actions

# RL and Meta-Learning for RNA Design

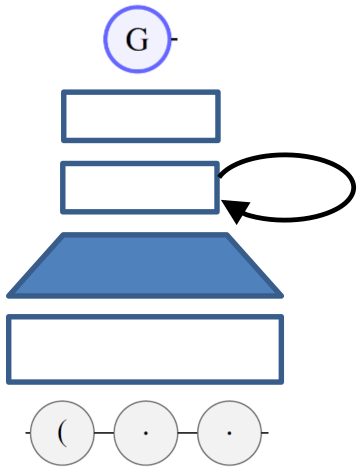
- **LEARNA**
  - Offline phase: -
  - Online phase:
    - Run PPO on the target structure
    - Run on 1 core, for 10 min (Rfam) or 1 day (Eterna); enough for about 100-10.000 episodes (depending on sequence length and policy network)

- **Meta-LEARNA**
  - Offline phase:
    - Optimize the policy network $\mathcal{P}$ with PPO, to maximize reward across a training set of RNA structures, for 1 hour on 20 parallel workers
    - This budget is less than the 24-hour budget for a single Eterna structure!
  - Online phase: iteratively sample from $\mathcal{P}$ on the target structure

- **Meta-LEARNA-adapt**
  - Offline phase: same as Meta-LEARNA
  - Online phase: continue running PPO on the target structure

# 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

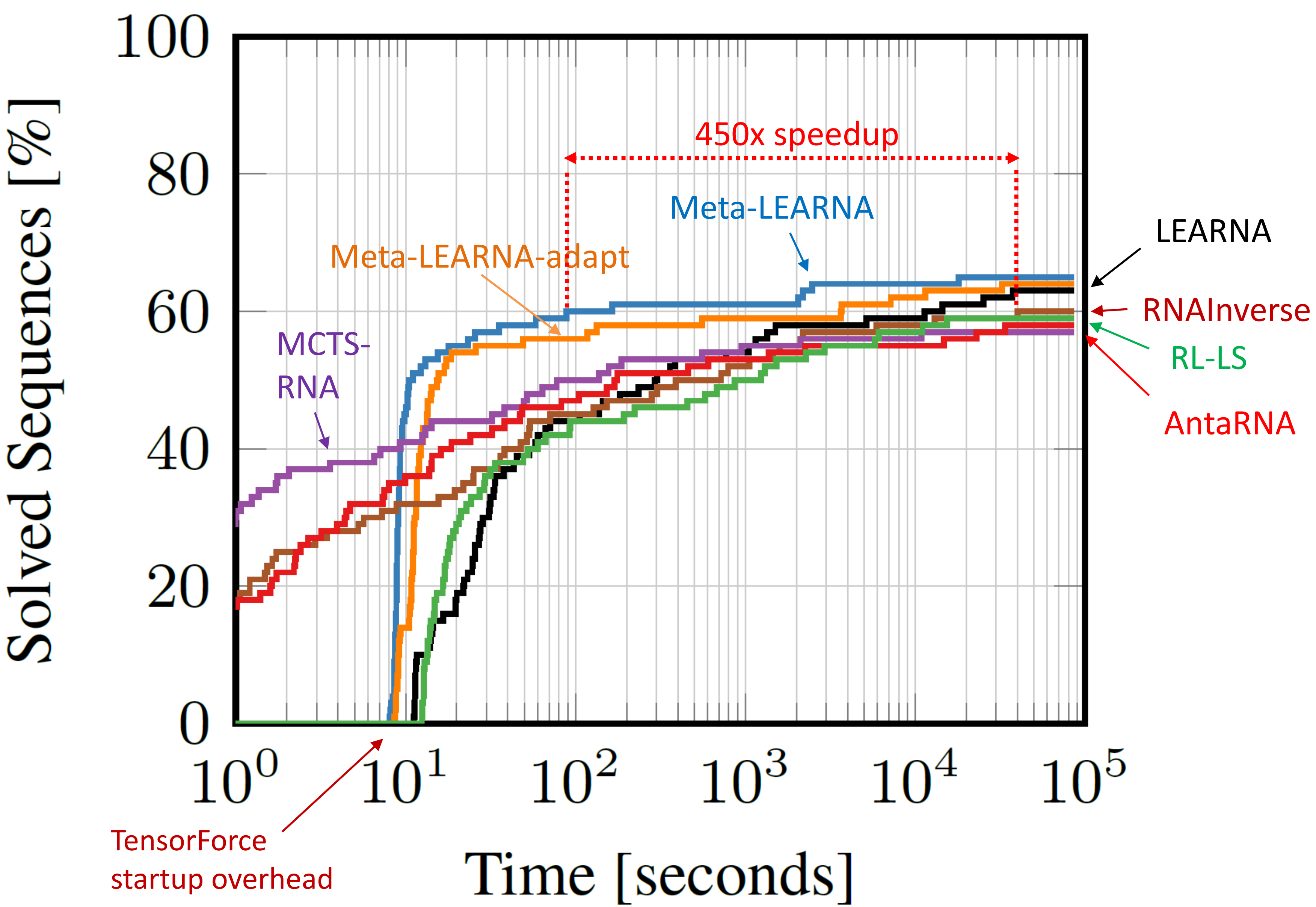
State representation: n-grams

- 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

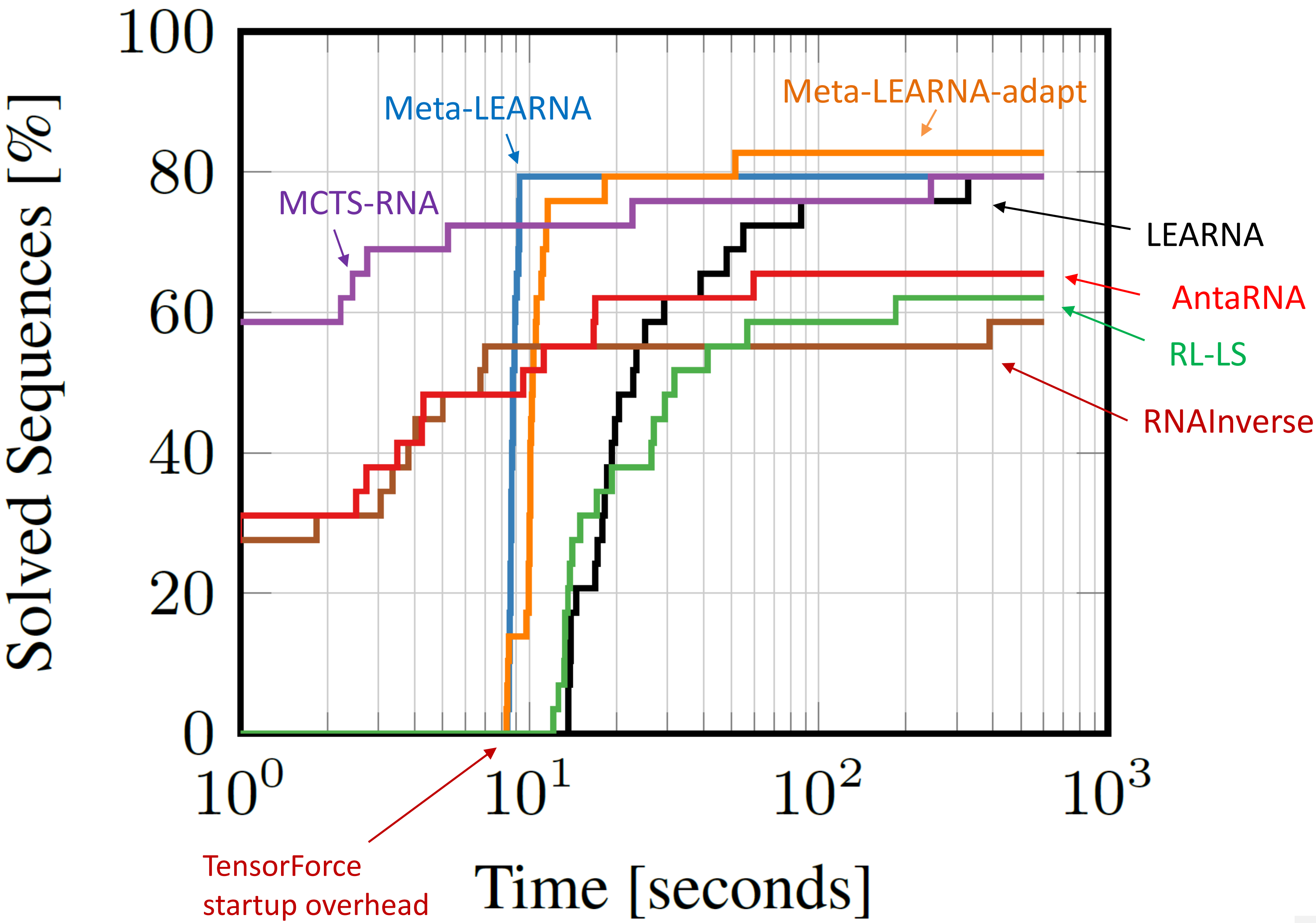
# Details for Optimization with BOHB

- Created a new set of RNA target structures for training
  - 65.000 structures for training, 100 for validation, 100 for test
- **Meta-optimizing LEARNA**
  - No offline learning phase, so directly optimized on the validation set
  - Full function evaluations on the Rfam dataset cost 10 minutes = 600s
  - Multi-fidelity budgets: 22s, 66s, 200s, 600s
  - Overall optimization budget: about 1 day on 180 CPU cores
- **Meta-optimizing Meta-LEARNA**
  - Maximum runtimes we used: 1h (on 20 workers)
  - Multi-fidelity budgets: 400s, 1200s, 3600s
  - Overall optimization budget: about 1 day on 1,000 CPU cores

# Results: Eterna100



# Results: Rfam-Taneda



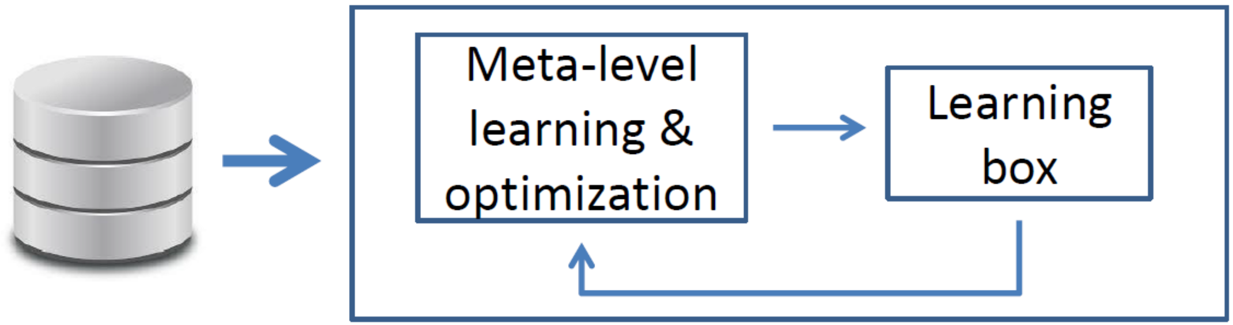
# Outline

- Part 1: AutoML as Blackbox Optimization
- Part 2: Speeding up AutoML
- Part 3: “Auto-RL” for Learning to Design RNA

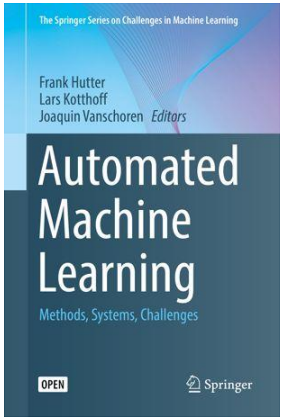
Conclusion

# Conclusion

- AutoML: **true end-to-end learning**




- **Large speedups by going beyond blackbox optimization**
  - Speedups in NAS and hyperparameter optimization
  - BOHB: combination of Bayesian optimization and Hyperband
  - AutoML is directly applicable to RL and Meta-Learning
  - Application to “Auto-RL” for learning to design RNA etc)
- Links to code: [http://automl.org](http://automl.org)
- Book on AutoML: [http://automl.org/book](http://automl.org/book)



# Thank you for your attention!

## Funding sources

BOSCH

erc European Research Council

Emmy Noether-Programm
DFG Deutsche Forschungsgemeinschaft

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

DFG Priority Programme 1527

DFG Deutsche Forschungsgemeinschaft

BrainLinks BrainTools acting.thoughts

European Commission

## My fantastic team

I'm looking for additional great postdocs!

@FrankRHutter
@automlfreiburg

AutoML.org