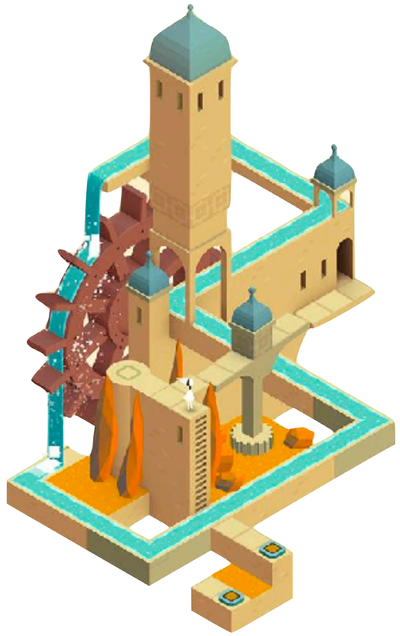


Automated Machine Learning

Open Data Science Conference
March 30th, 2021

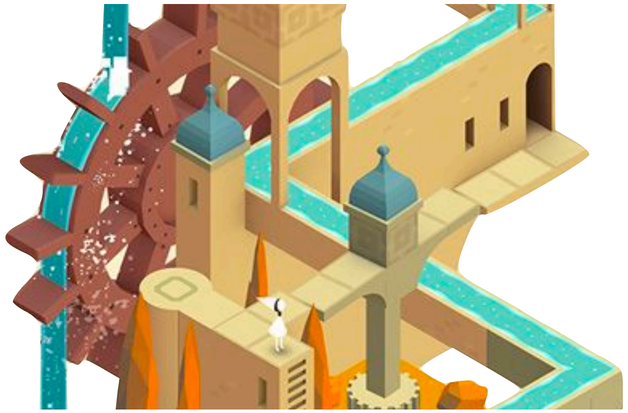
Joaquin Vanschoren & Pieter Gijsbers
Eindhoven University of Technology

image credit: ustwo

Overview

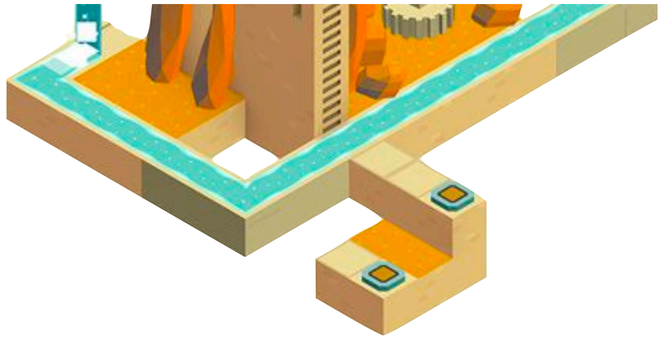
Part 1: *Why* automate machine learning?

High-level goals



Part 2: *How* AutoML works

The machinery



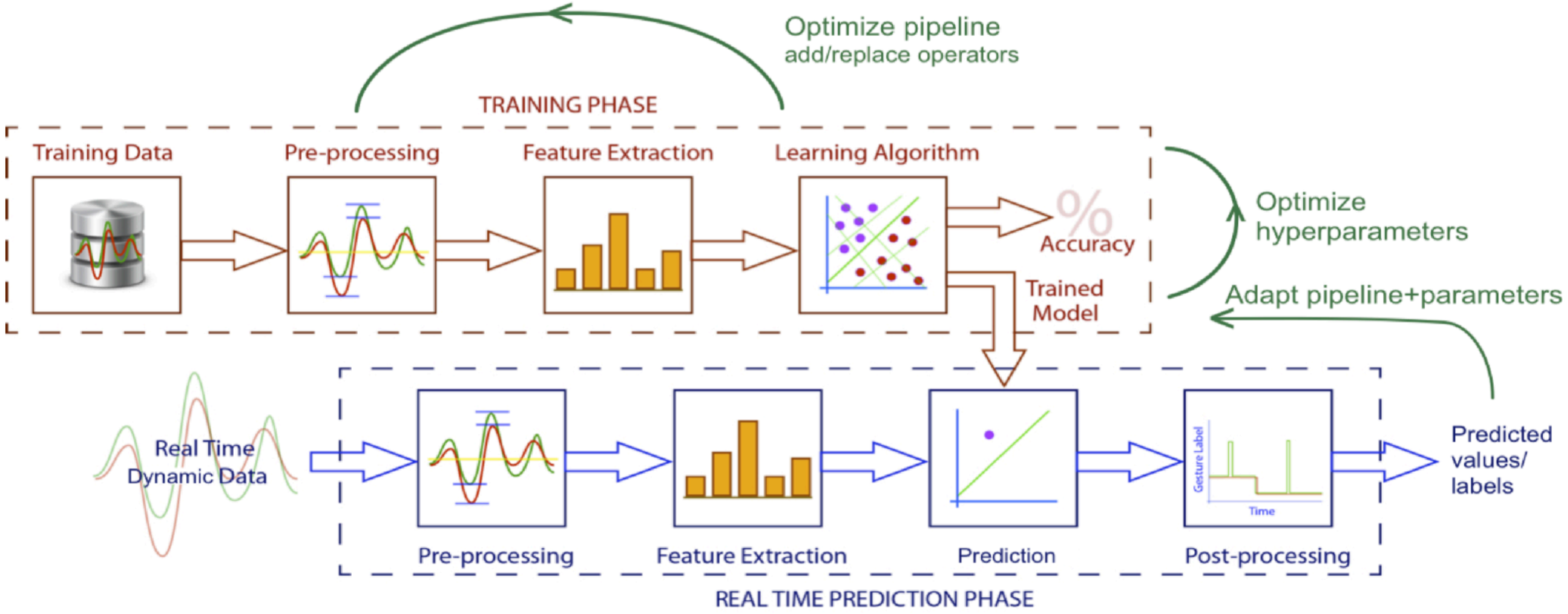
Part 3: Learning how to do AutoML

Closing the loop

Part 1: *Why* automate machine learning?

High-level goals

Doing machine learning requires lots of expertise and exploration

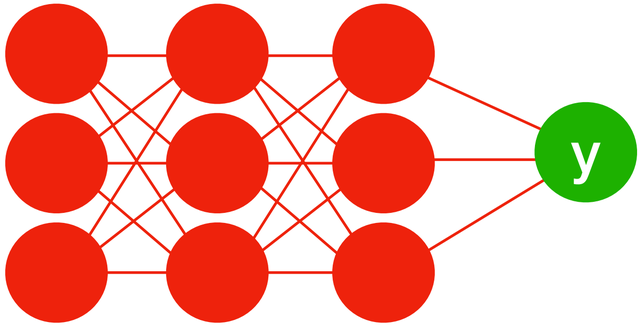


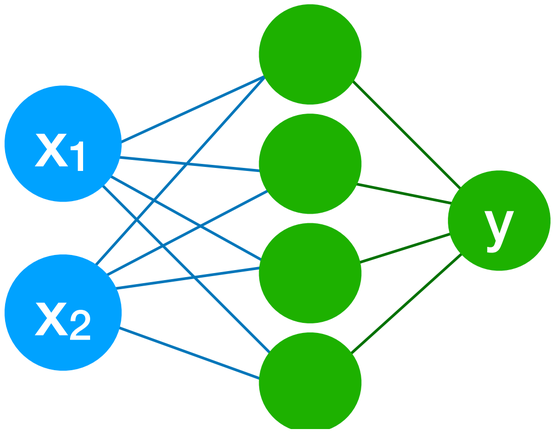
Cleaning, preprocessing, feature selection/engineering features, model selection, hyperparameter tuning, adapting to concept drift,…

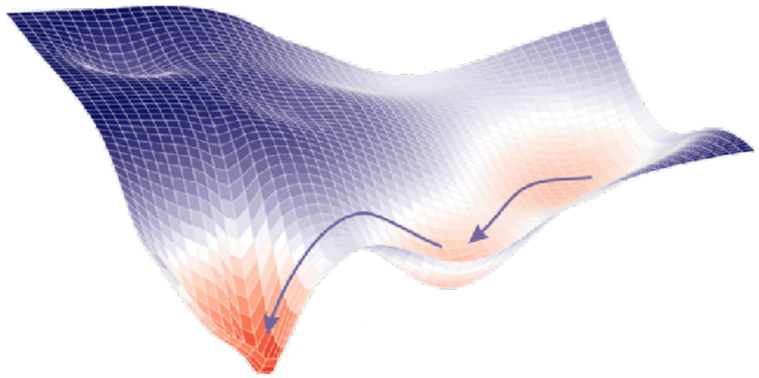
Doing machine learning requires lots of expertise and exploration

Architecture:

- Type of operators
- Size of layers
- Filter sizes
- Skip connections
- Pre-trained layers
- Transformers
- …

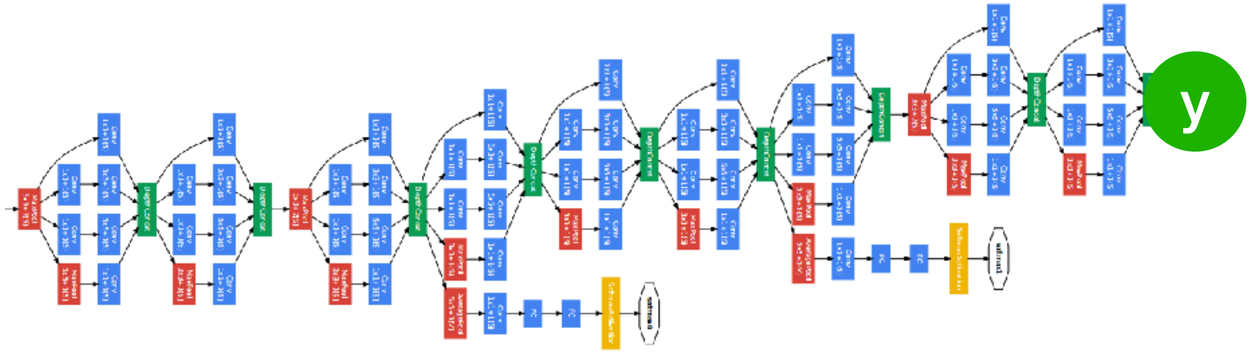






Optimization:

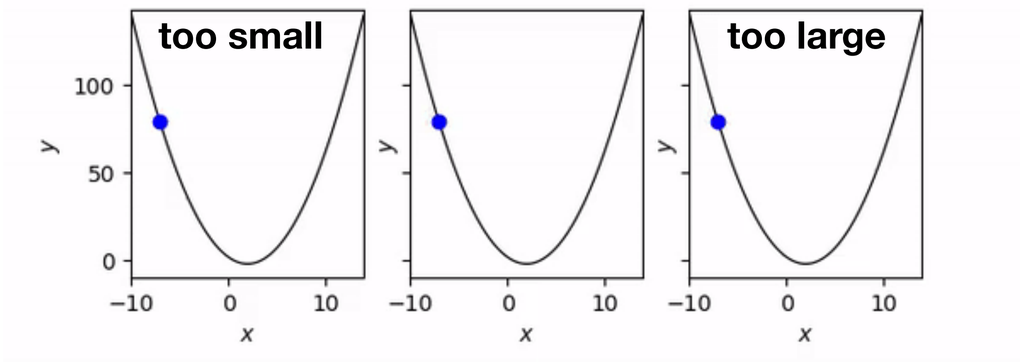
- Gradient descent hyperparameters
- Regularization
- …



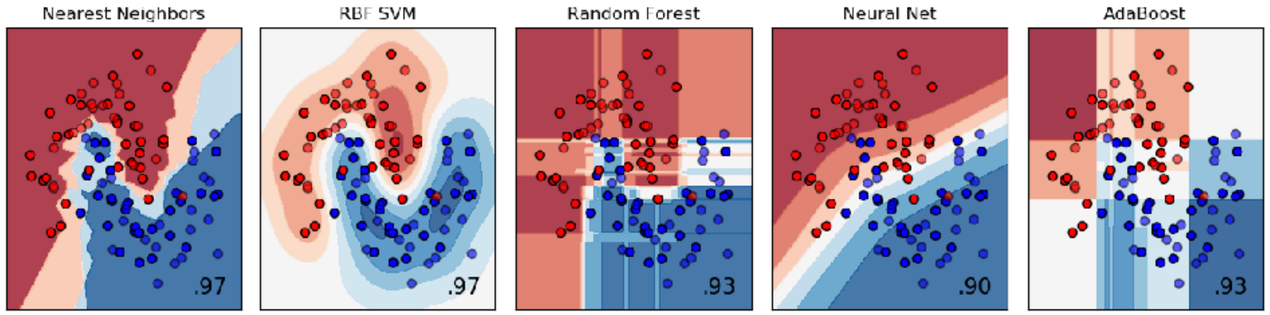
Hyperparameters

Every design decision usually made by the *user (architecture, operators, tuning,…)*

- Numeric
  - e.g. learning rate

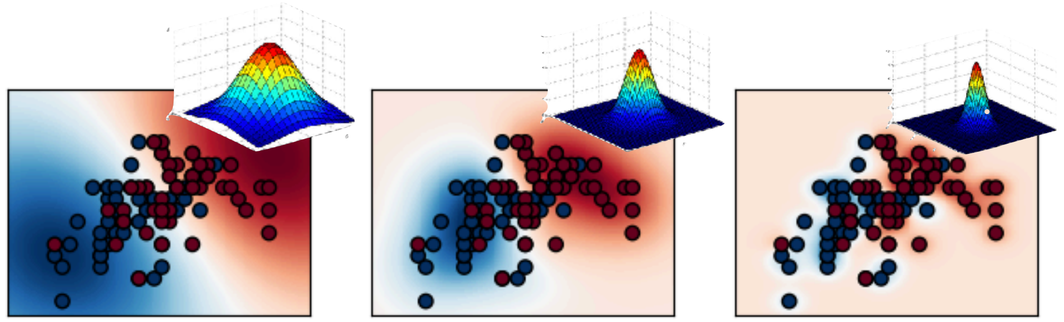


- Categorical
  - e.g. classifier



- Conditional
  - SVM -> kernel?

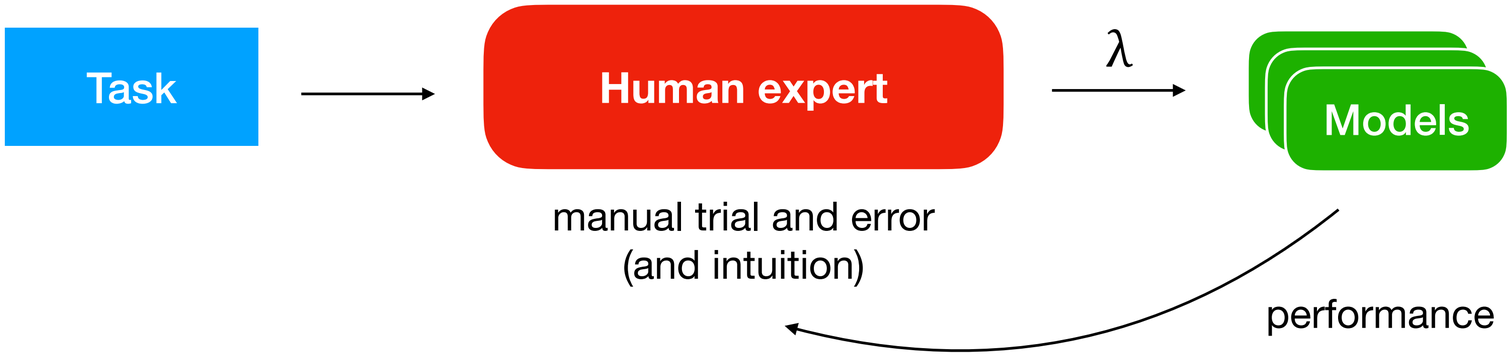
    RBF kernel -> gamma?



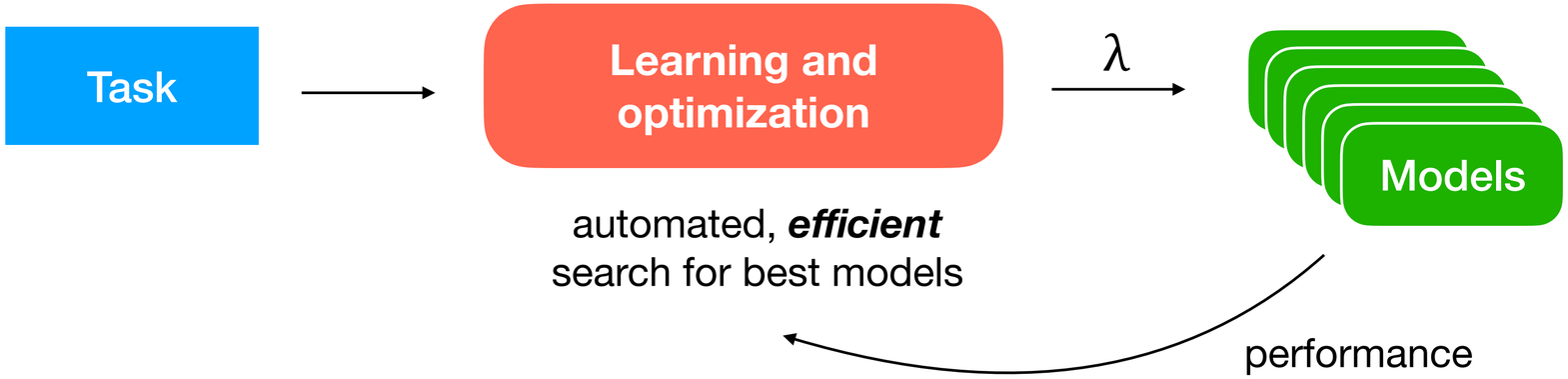
Hutter et al. 2019

Automatic Machine Learning (AutoML)

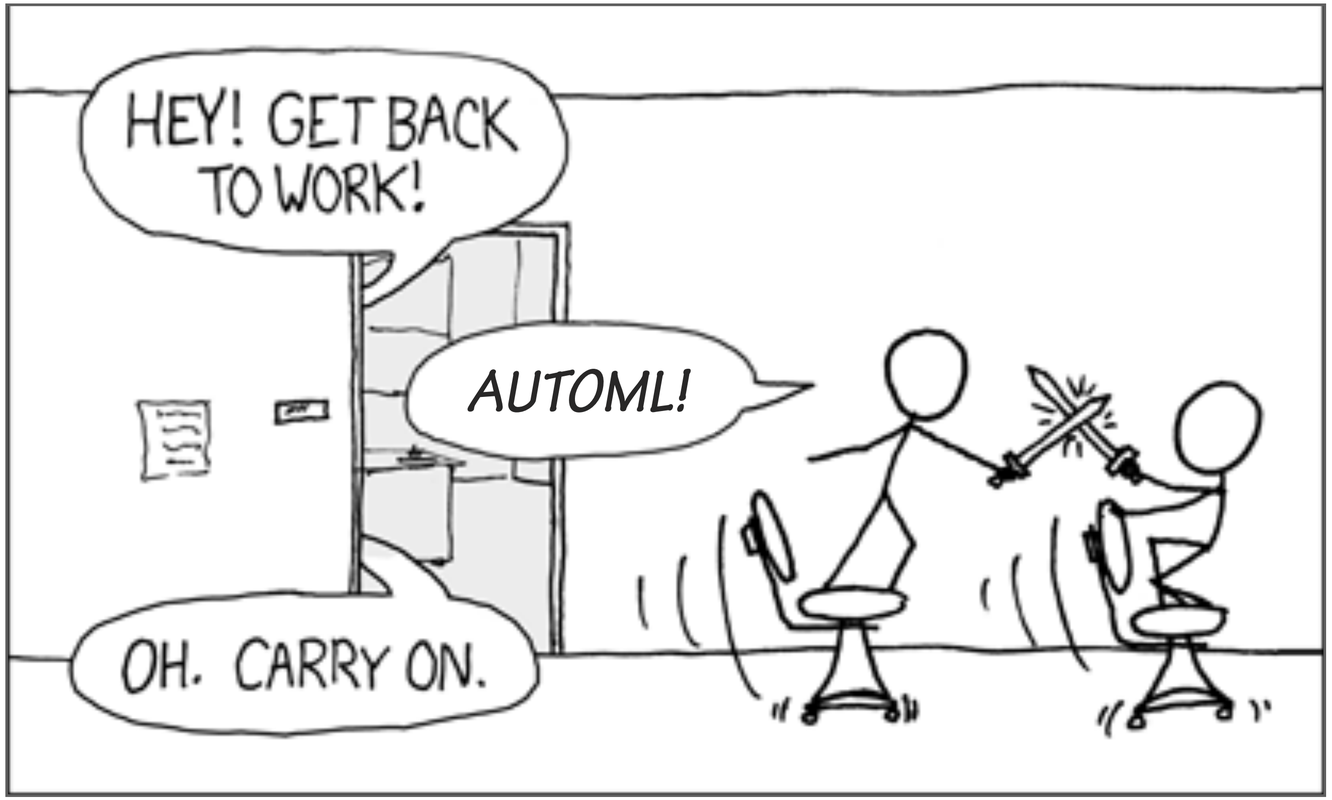
Manual machine learning



AutoML: build models in a *data-driven, intelligent, purposeful* way



THE DATA SCIENTIST'S #1 EXCUSE FOR LEGITIMATELY SLACKING OFF: "THE AUTOML TOOL IS OPTIMIZING MY MODELS!"



Adapted from XKCD (Randall Munroe)

> Antenna broken. No communication with Earth.

*Sigh. I'm going to have to AutoML the s*** out of this*

Many open challenges in AutoML!

Human in the loop

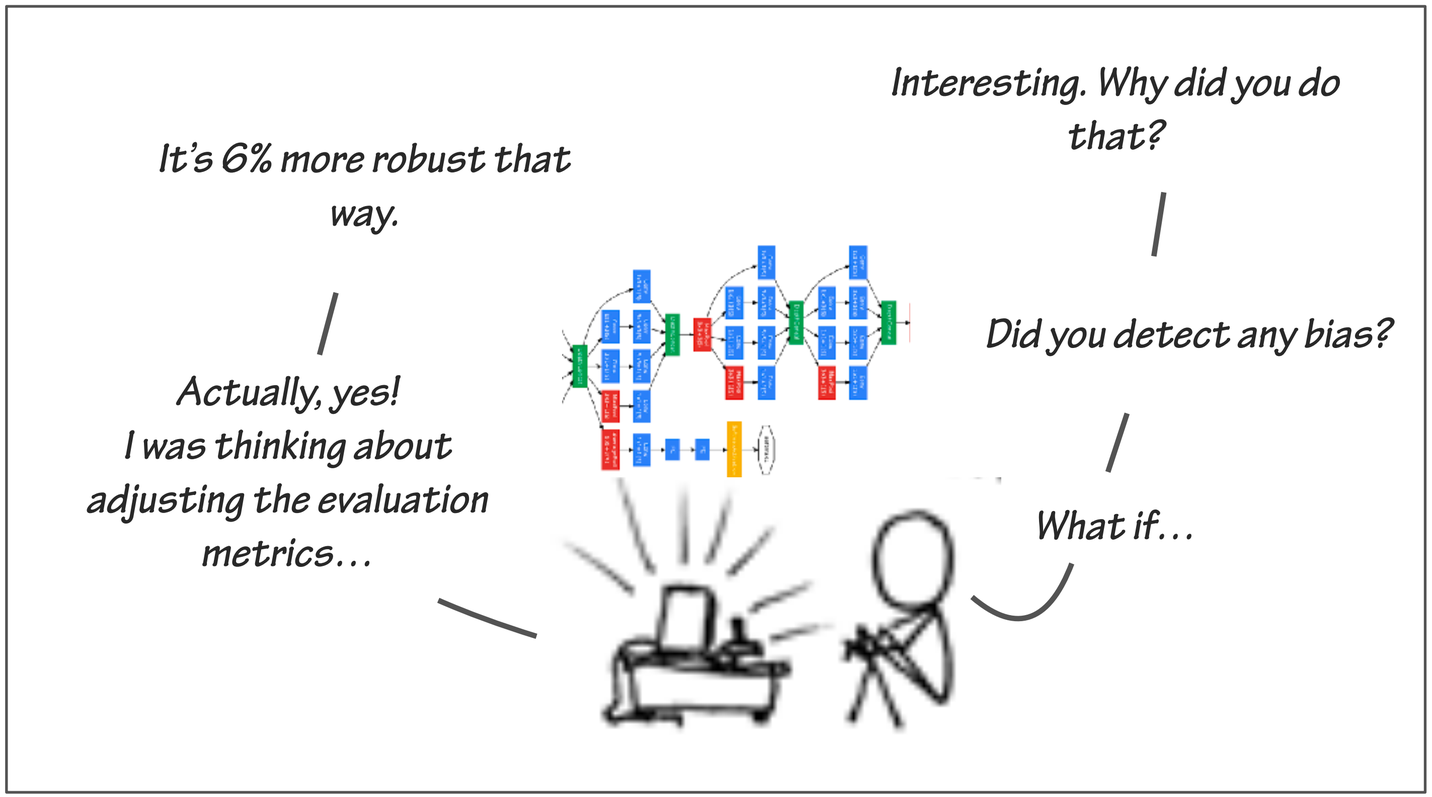
Explainable process/models

Ask for help/data

Self-assess trustworthiness

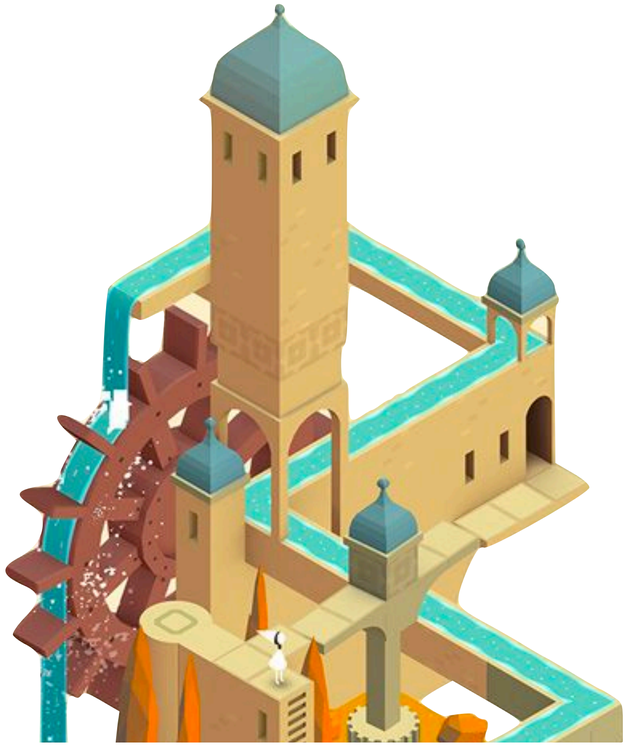
Real-world constraints

Meta-learning



Adapted from XKCD (Randall Munroe)

Overview

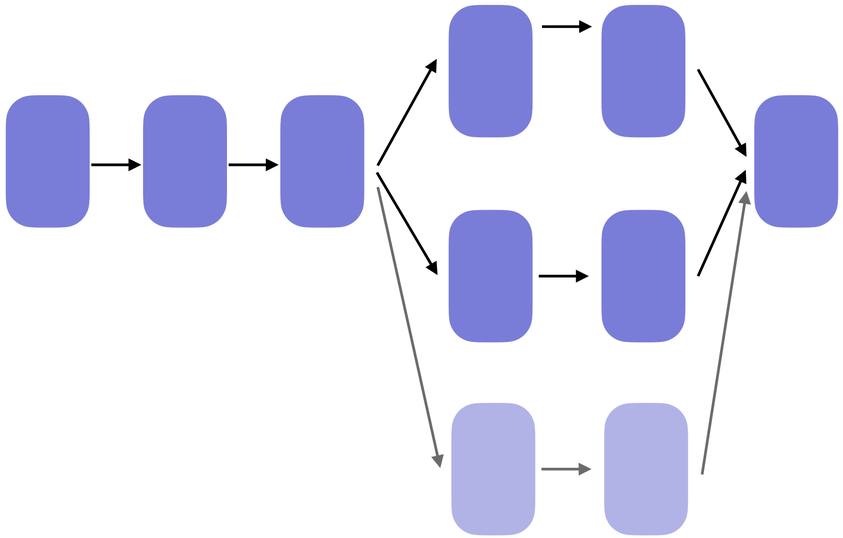


Part 2: *How* AutoML works

The machinery

AutoML: subproblems

- **Architecture search space**: *represent* all pipelines or neural architectures
  - Pipeline operators, neural layers, interconnections,…
  - Defines a (complex) search space

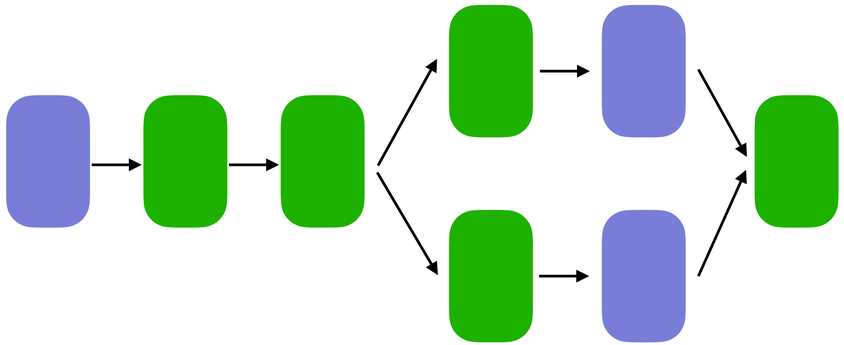


```
make_pipeline(
 OneHotEncoder(),
 Imputer(),
 StandardScaler(),
 SVC())
```

```
model.add(Conv2D(32, (3, 3))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3))
```

AutoML: subproblems

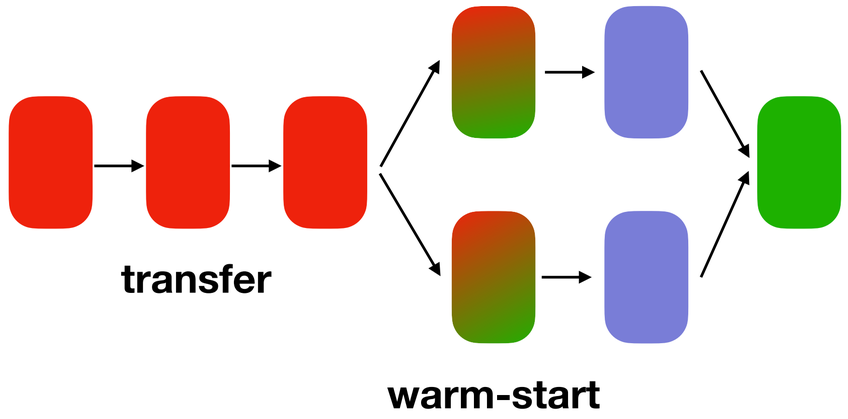
- **Architecture search space**: *represent* all possible architectures
- **Optimization**:
  - What is the best architecture? Which hyperparameters are important?
  - How to optimize them? What is the (multi-) objective function?



```
hyper_space = {'SVC__C': expon(scale=100),
               'SVC__gamma': expon(scale=.1)}
RandomizedSearchCV(pipe, param_distributions=hyper_space, n_iter=200)
```

AutoML: subproblems

- **Architecture search space**: *represent* all possible architectures
- **Optimization**: *optimize* architecture and hyperparameters
- **Meta-learning**: how can we transfer *experience* from previous tasks?
  - Don’t start from scratch (search space is too large)
  - Transfer learning: reuse good architectures/configurations/weights
  - Warm starting: start from promising architectures/configurations/initializations

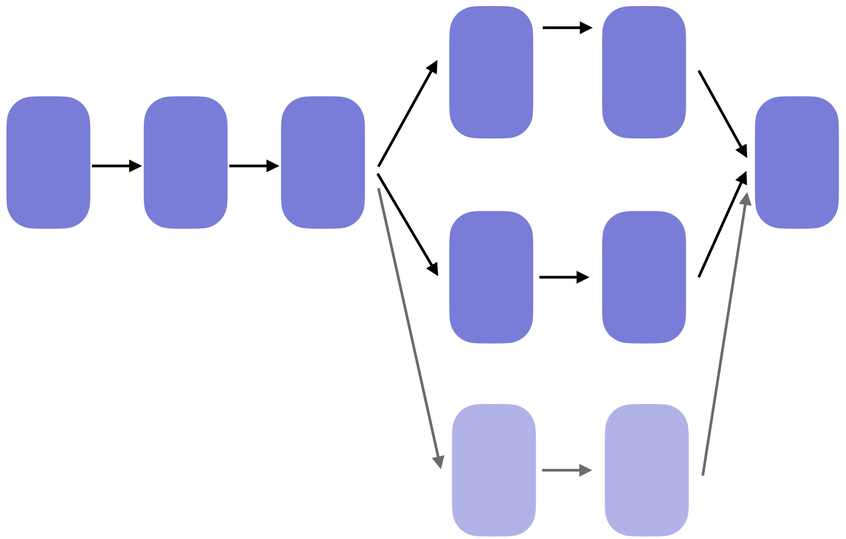


AutoML: subproblems

- **Architecture search**: *represent* all possible architectures
- **Optimization**: *optimize* architecture and hyperparameters
- **Meta-learning**: how can we transfer experience from previous tasks?

Many combinations are possible!
They can be done *consecutively*, *simultaneously* or *interleaved*

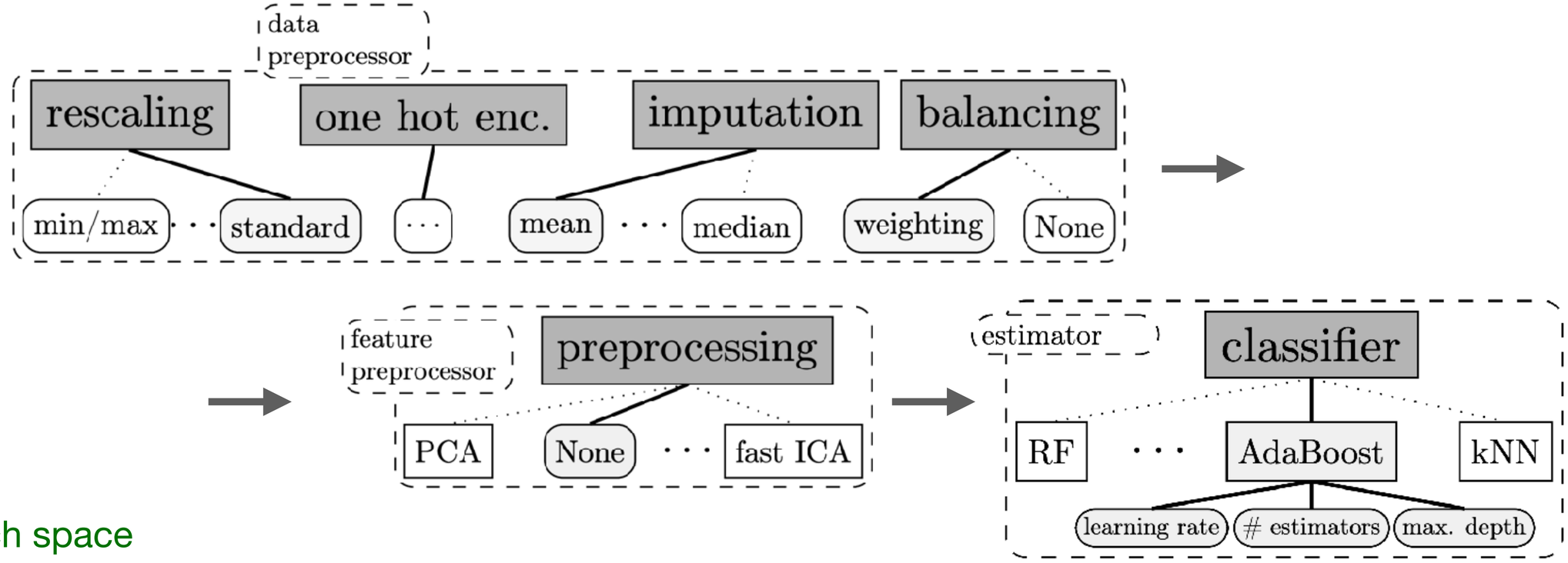
Architecture search space



Parameterized architectures

autosklearn *Feurer et al. 2015*
autoWEKA *Thornton et al. 2013*
hyperopt-sklearn *Komer et al. 2014*

- Manual bias: most successful pipelines have a similar structure
- Fix architecture, encode ***all*** choices as extra hyperparameters
  - *Architecture search becomes hyperparameter optimization*

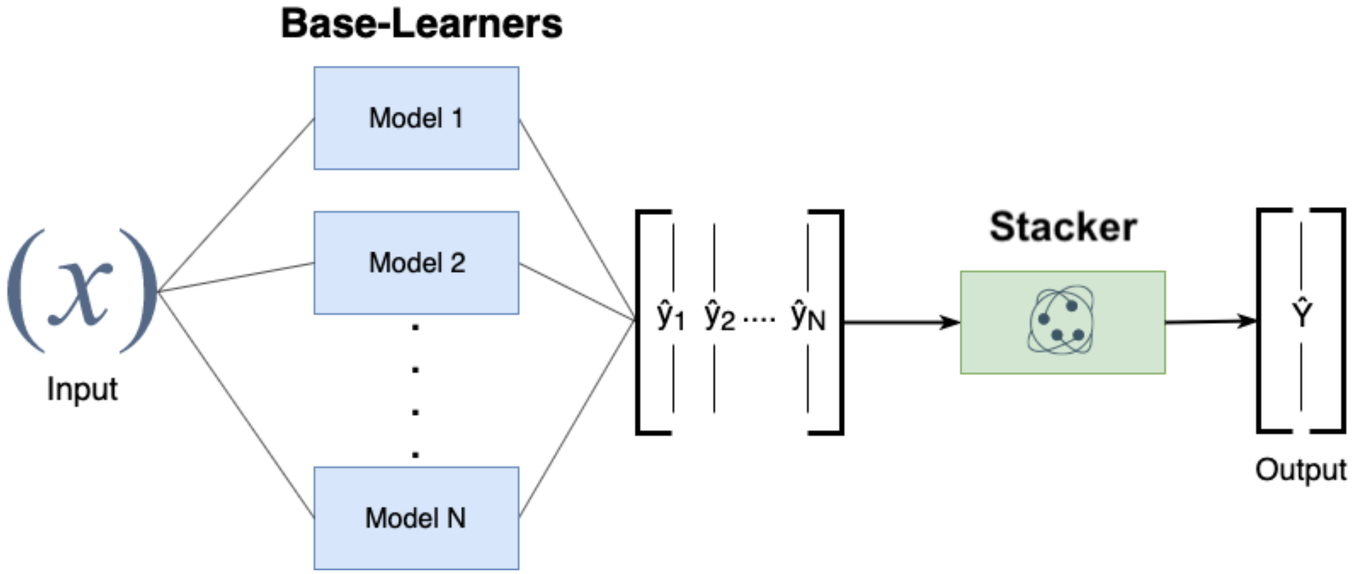


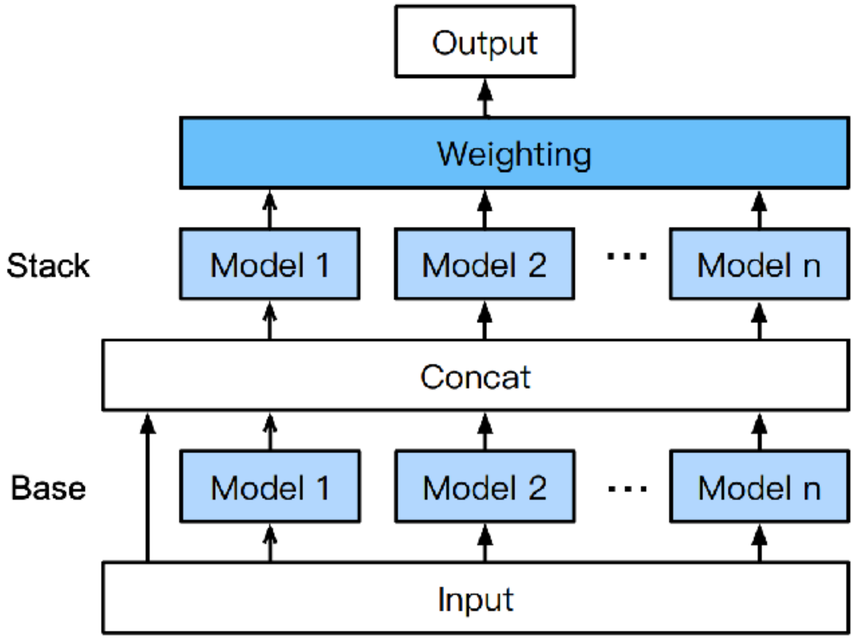
+ smaller search space
- you can't learn entirely new architectures

Ensembling

autosklearn *Feurer et al. 2015*
GAMA *Gijsbers & Vanschoren, 2020*
AutoGluon-Tabular, *Erikson et al. 2020*
H2O AutoML, *LeDell et al.*

- Build ensembles of multiple pipelines to avoid overfitting
  - RandomForests (Auto-sklearn, GAMA,…)
  - Stacking (AutoGluon-Tabular, H2O AutoML,…)





Multi-level stacking

Elsken et al. 2019

Parameterized neural architectures

Parameterized Sequential

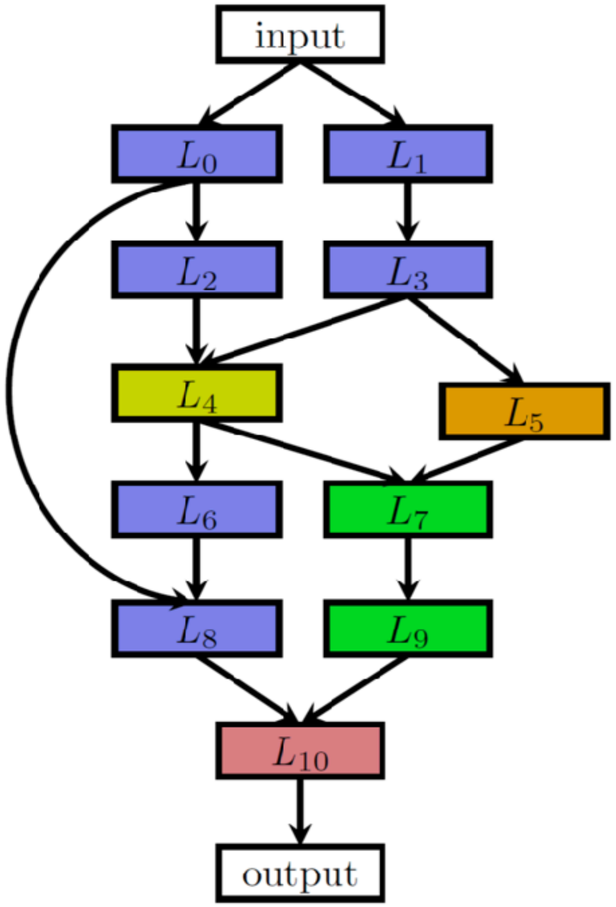
Choose:
- number of layers
- type of layers
  - dense
  - convolutional
  - max-pooling
  - …
- hyperparameters of layers

+ easier to search
- sometimes too simple

Parameterized Graph

Choose:
- branching
- joins
- skip connections
- types of layers
- hyperparameters of layers

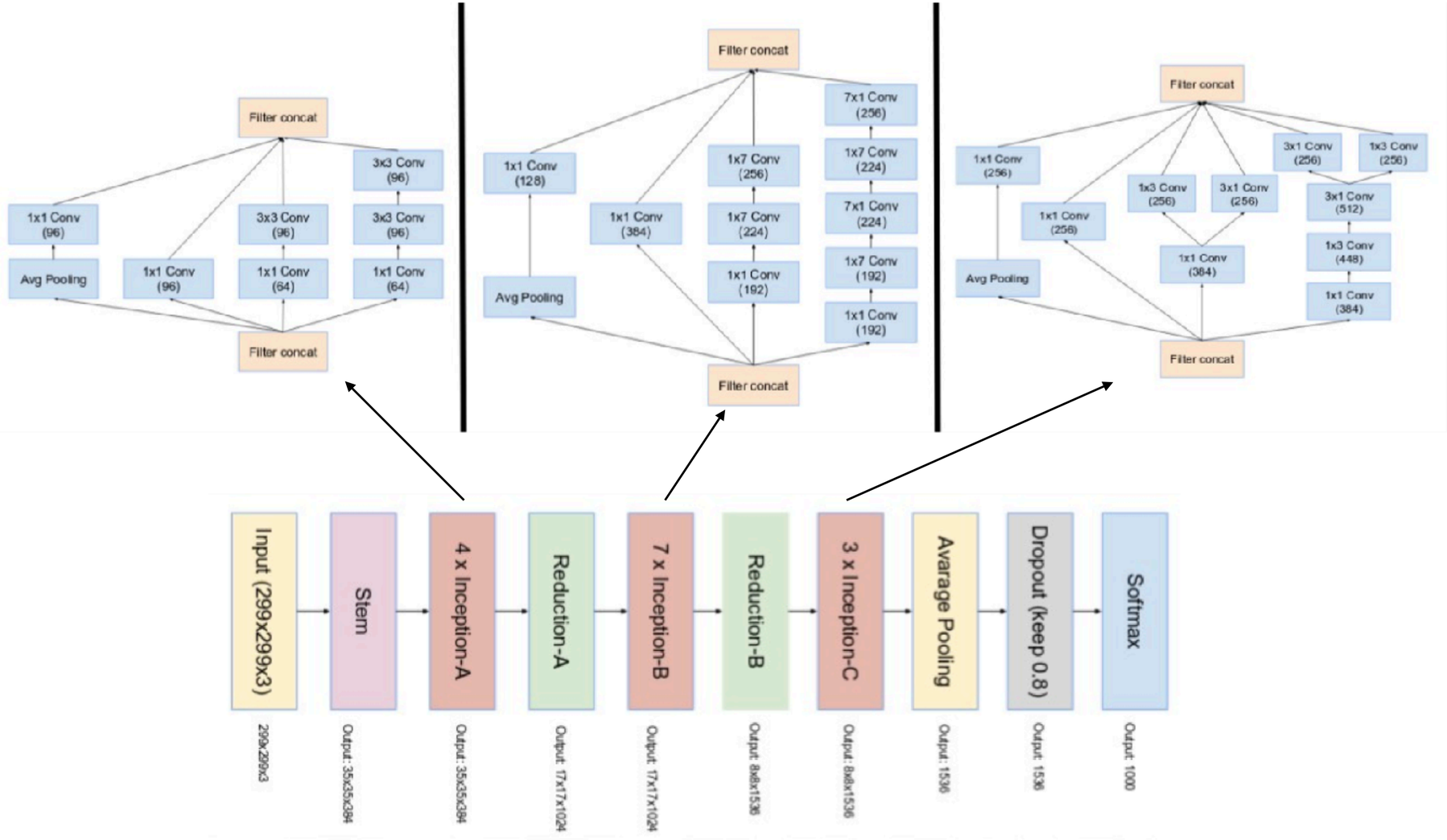
+ more flexible
- much harder to search



Cell search spaces

Manual bias: successful deep networks have repeated motifs (cells)

e.g. Inception v4:



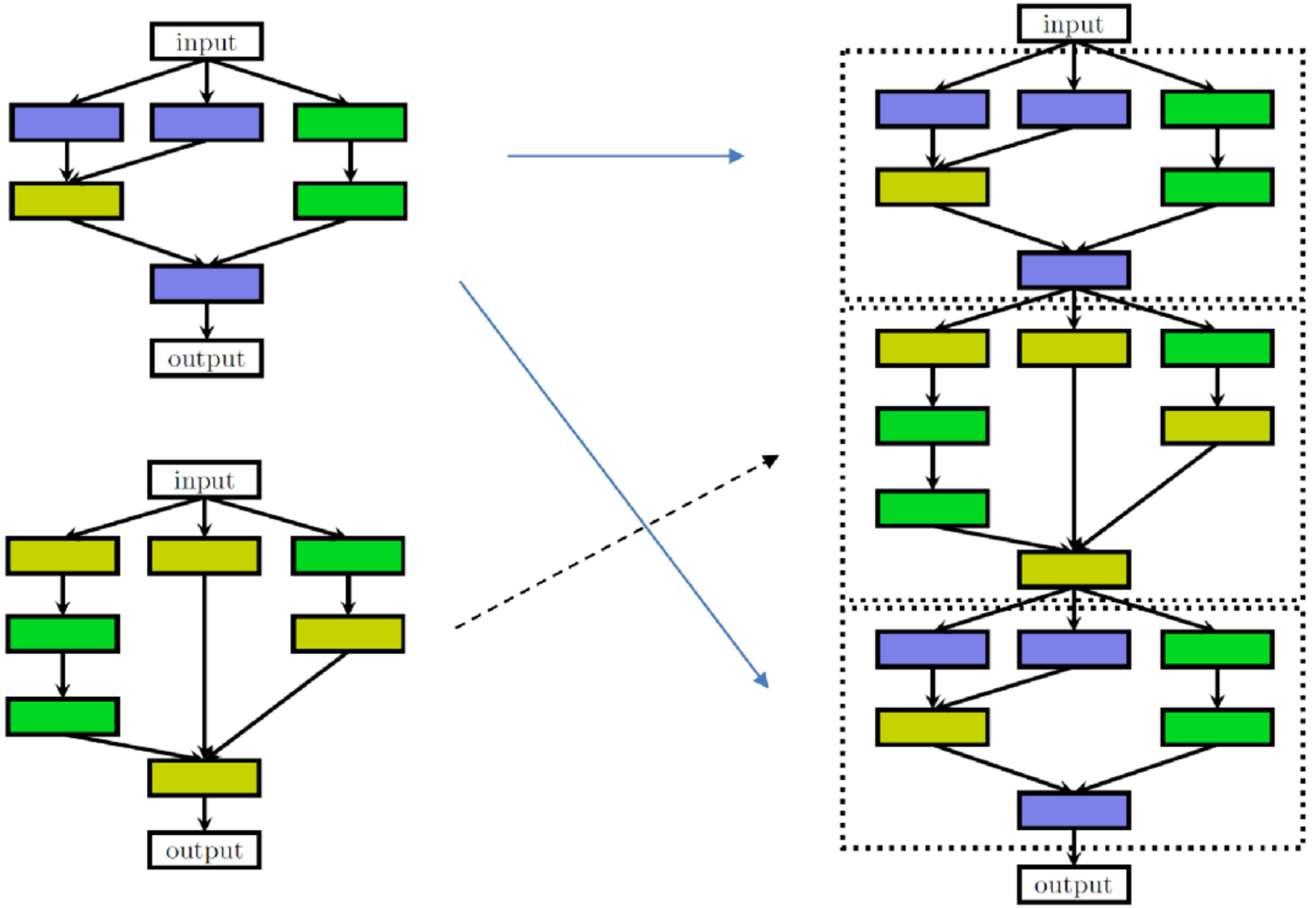
Cell search spaces

Google NASNet Zoph et al 2018

Compositionality: learn hierarchical building blocks to simplify the task

Cell search space

- learn parameterized building blocks *(cells)*
- stack cells together in macro-architecture

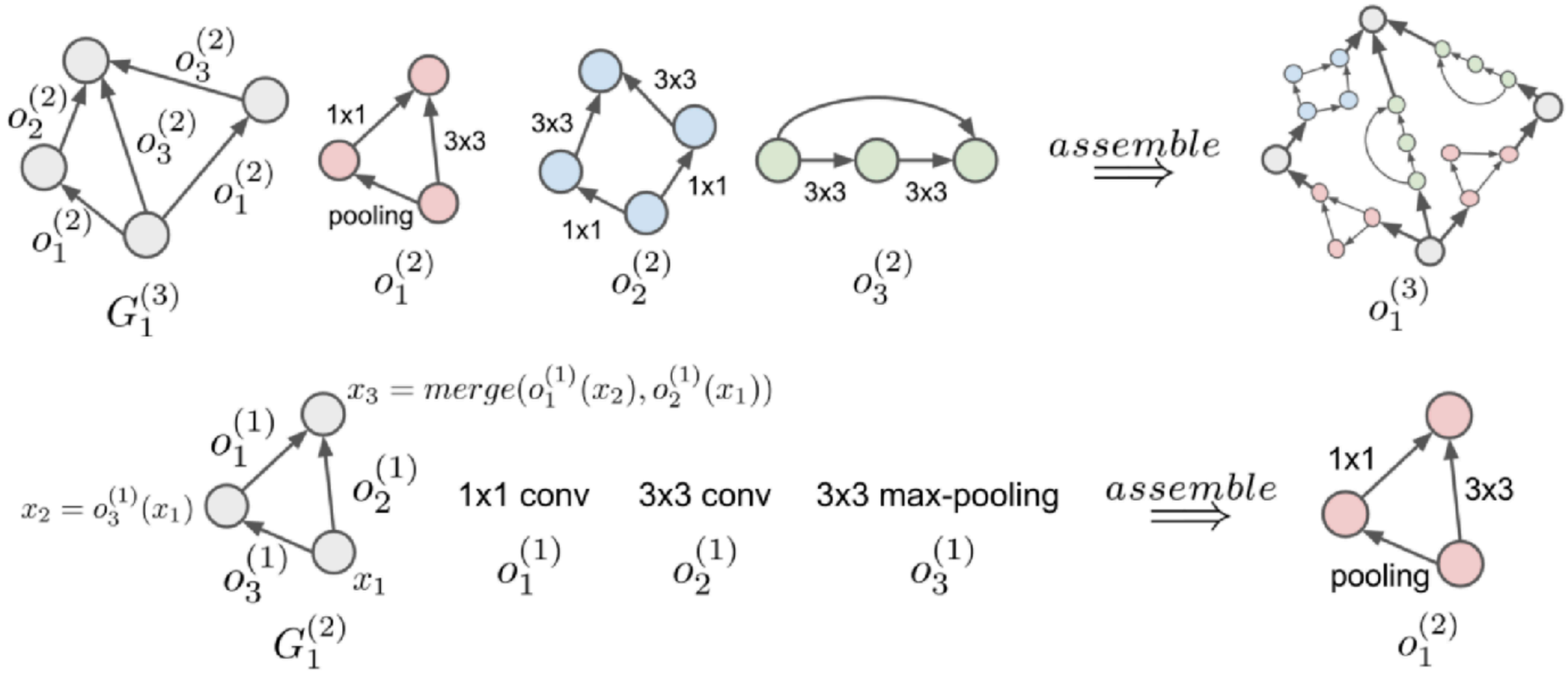


\+ smaller search space
\+ cells can be learned on a small dataset & transferred
\- strong domain priors, doesn't generalize well

Hierarchical Search Spaces

Liu et al. 2018

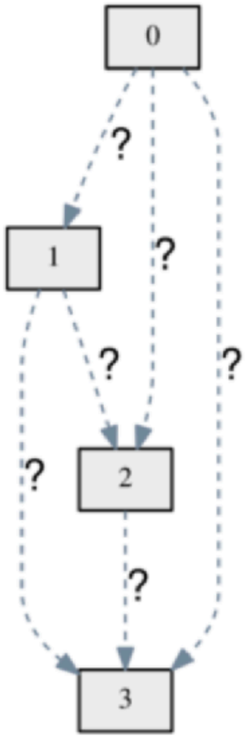
- Define a number of primitive operations (e.g. convolution, max-pooling,…)
- Build small motifs of primitives (o_1, o_2, o_3,…)
- Choose a graph structure G_i, replace edges with motifs, repeat



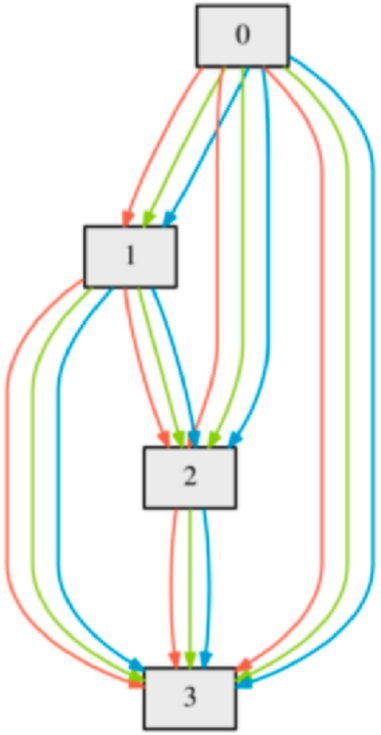
Differentiable Search Spaces (DARTS)

Liu et al. 2018

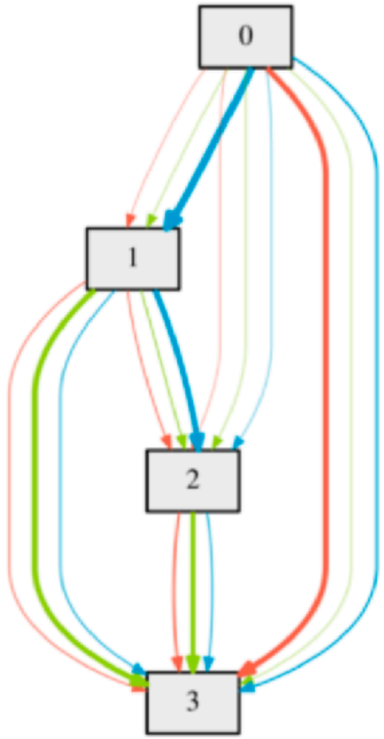
- Fixed (one-shot) structure, each edge can be any primitive operation
- Give all operators a weight α_i ,
- Optimize α_i and model weights ω_j using bilevel programming



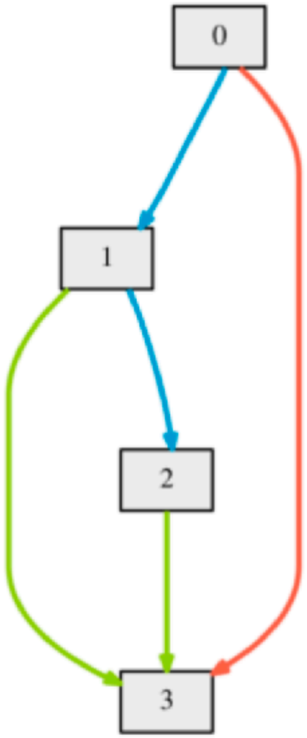
One-shot model



operator weights α_i

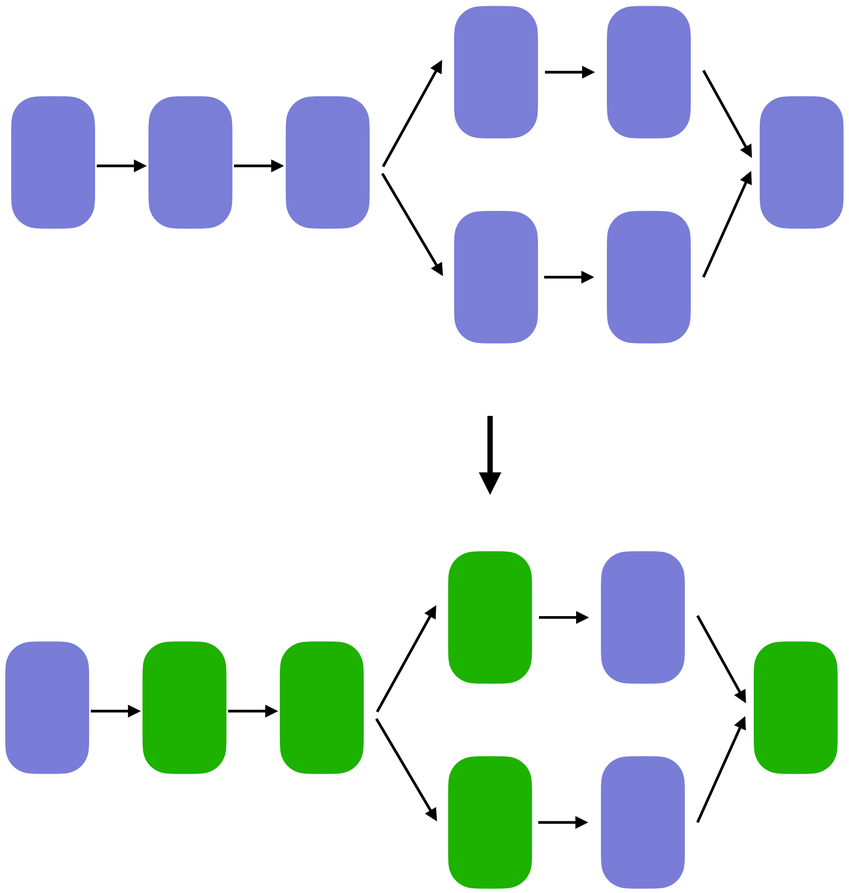


interleaved optimization of α_i and ω_j with SGD



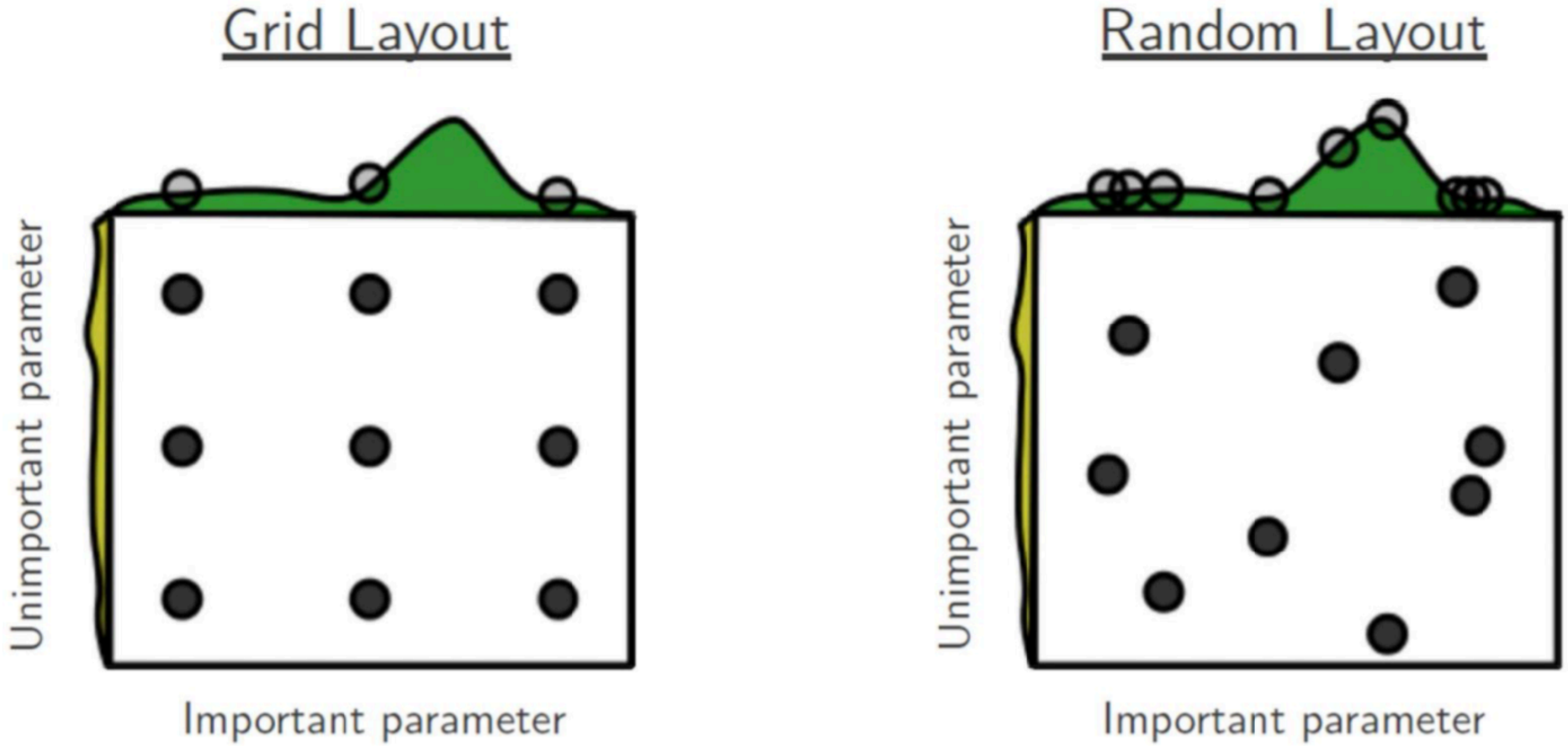
argmax α_i

Optimization



Random search

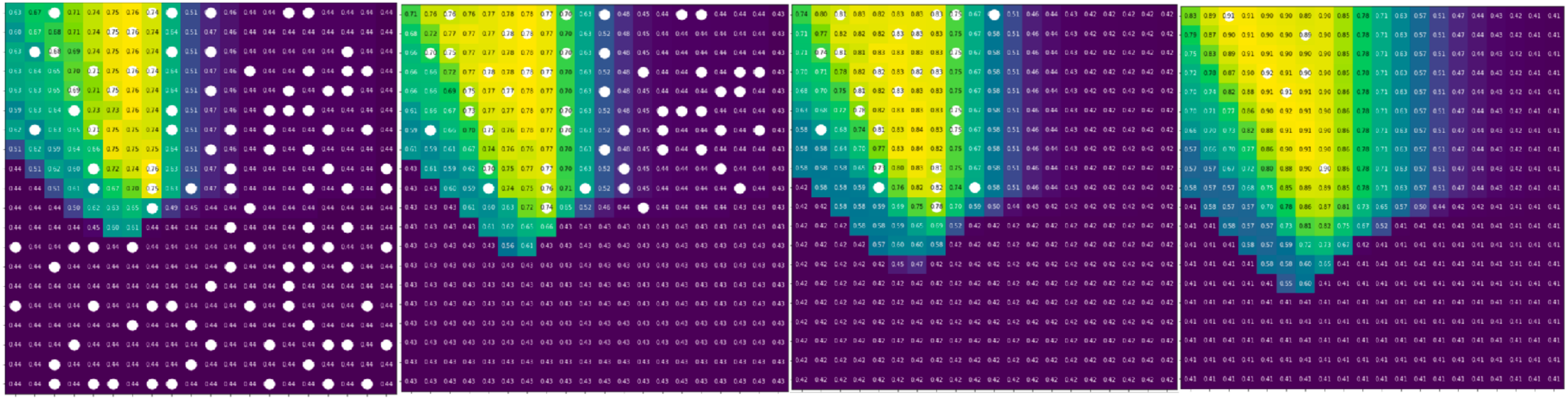
- Handles unimportant dimensions better than grid search
- Easily parallelizable, but uninformed (no learning)



Jamieson & Talwalkar 2016

Successive Halving

- Train on small data subsets, infer which regions may be interesting to evaluate in more depth
  - Randomly sample candidates and evaluate on a small data sample
  - Retrain the 50% best candidates on twice the data, repeat



1/16 1/8 1/4 1/2

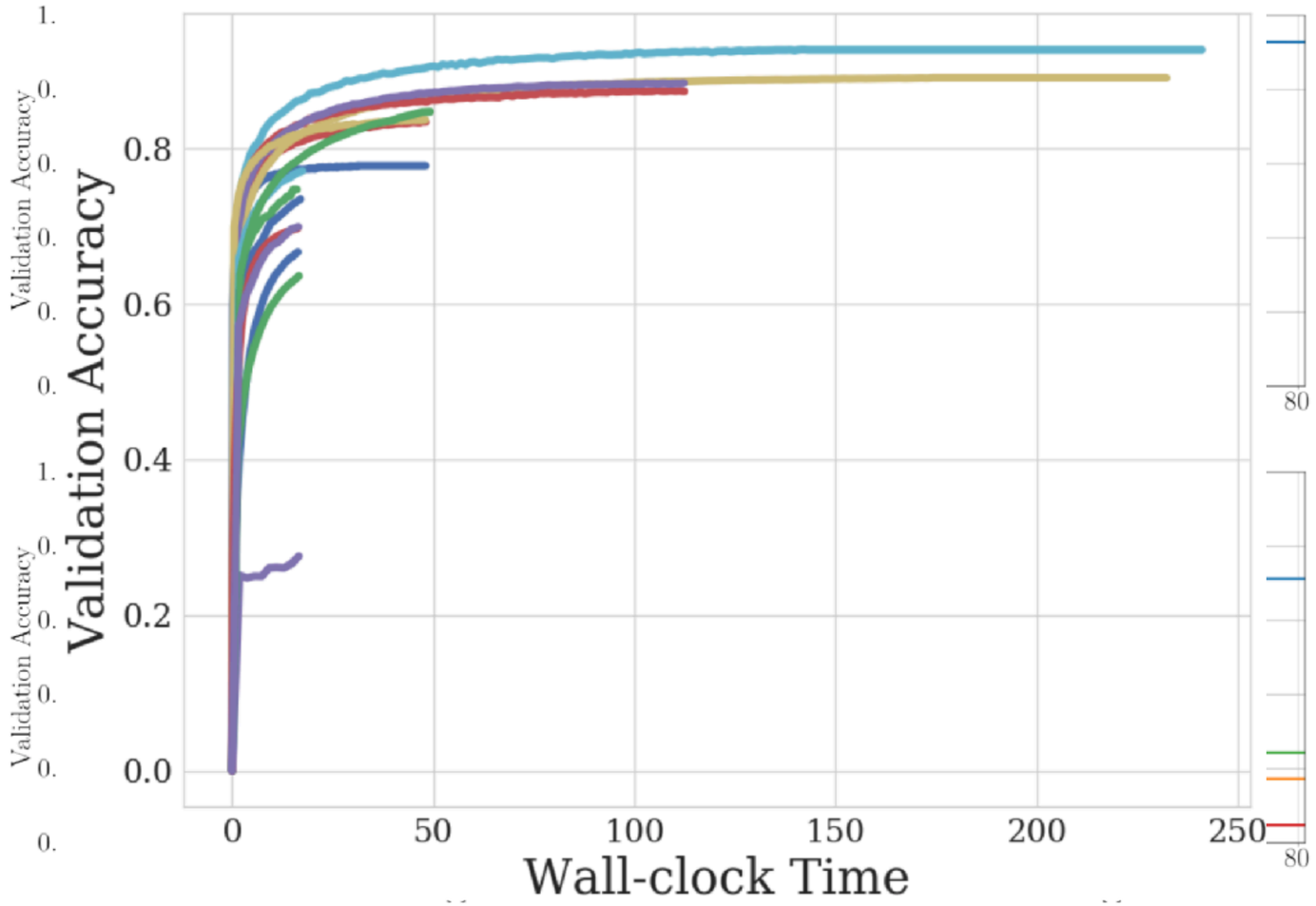
sample size

Li et al. 2017

Hyperband

Successive halving risks killing good models too early

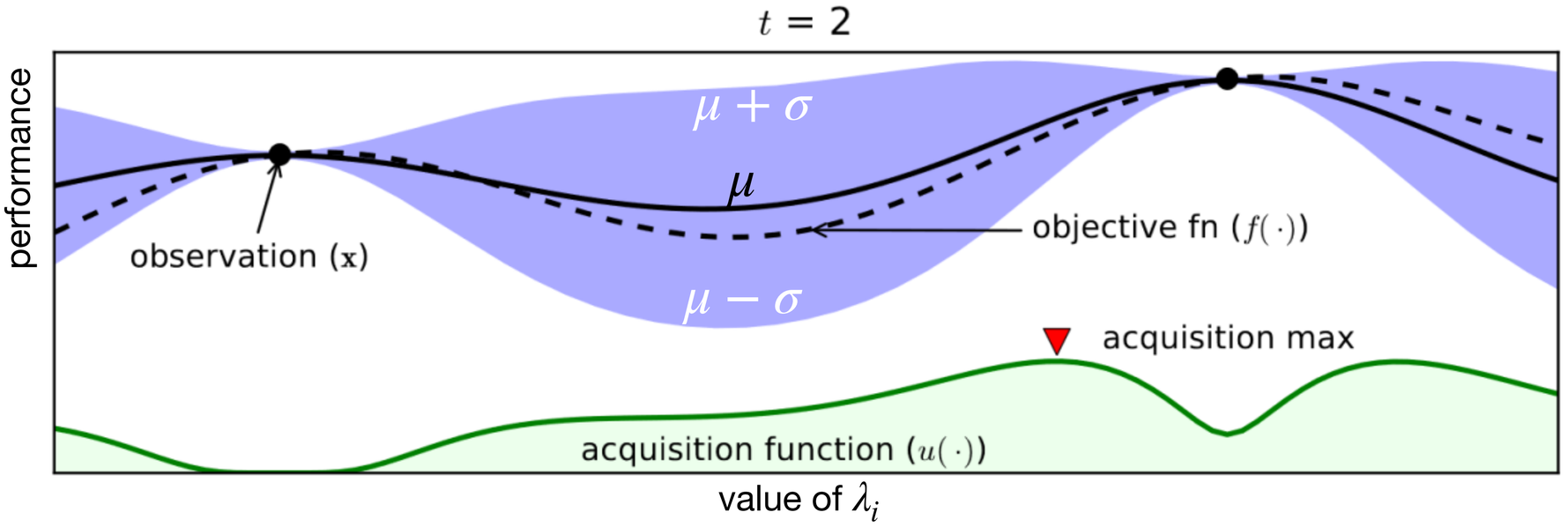
- Repeat in multiple, decreasingly aggressive iterations (brackets)
- Strong anytime performance, easy to implement, scalable, parallelizable



Mockus [1974]

Bayesian Optimization

- Start with a few (random) hyperparameter configurations
- Fit a ***surrogate model*** to predict other configurations
- Probabilistic regression (e.g. Gaussian Processes): mean μ and standard deviation σ (blue band)
- Use an ***acquisition function*** to trade off exploration and exploitation, e.g. Expected Improvement (EI)
- Sample for the best configuration under that function



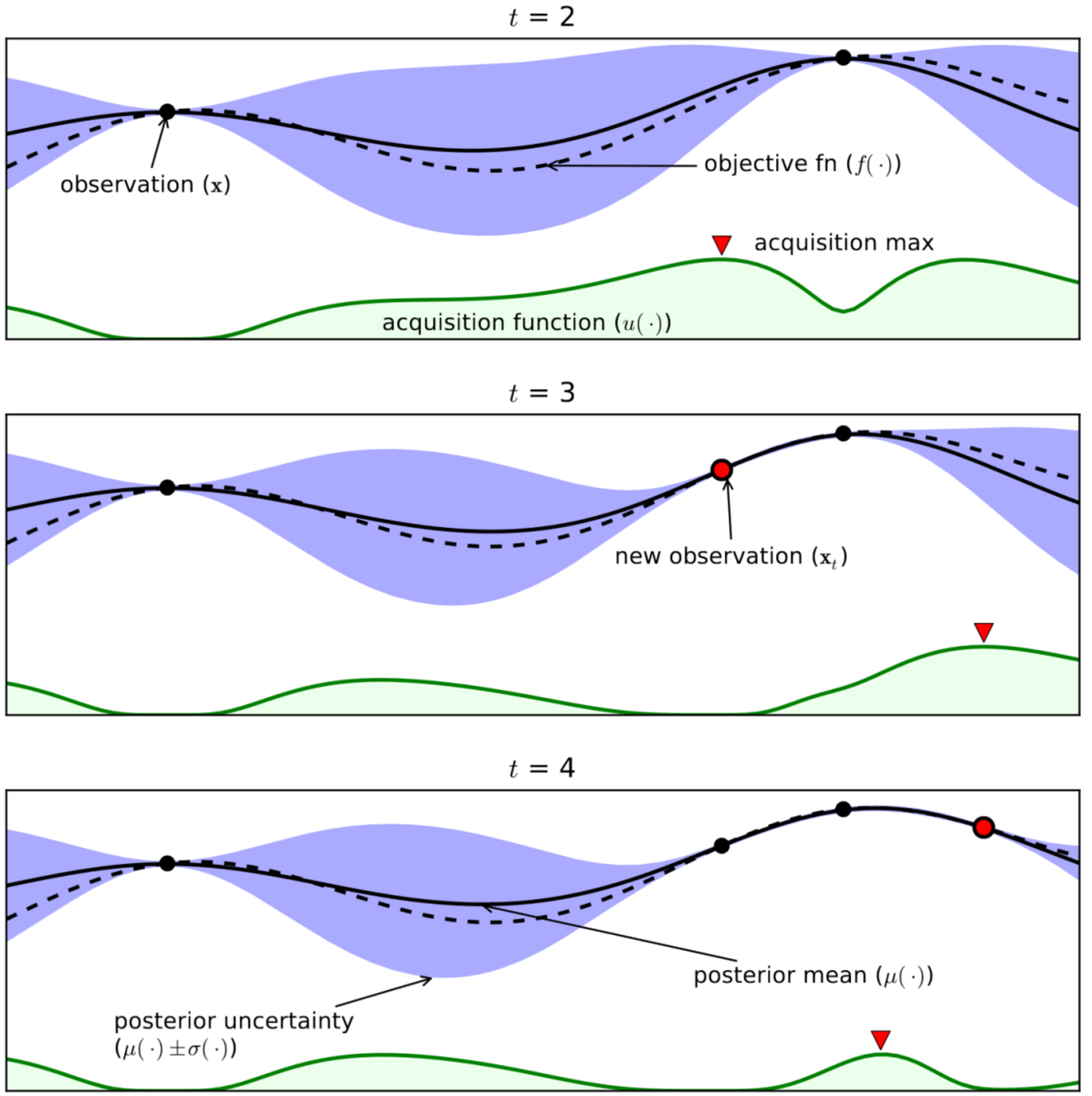
Bayesian Optimization

- Repeat until some stopping criterion:
  - Fixed budget
  - Convergence
  - EI threshold

- Theoretical guarantees

  Srinivas et al. 2010, *Freitas et al. 2012*, *Kawaguchi et al. 2016*

- Also works for non-convex, noisy data
- Used in AlphaGo

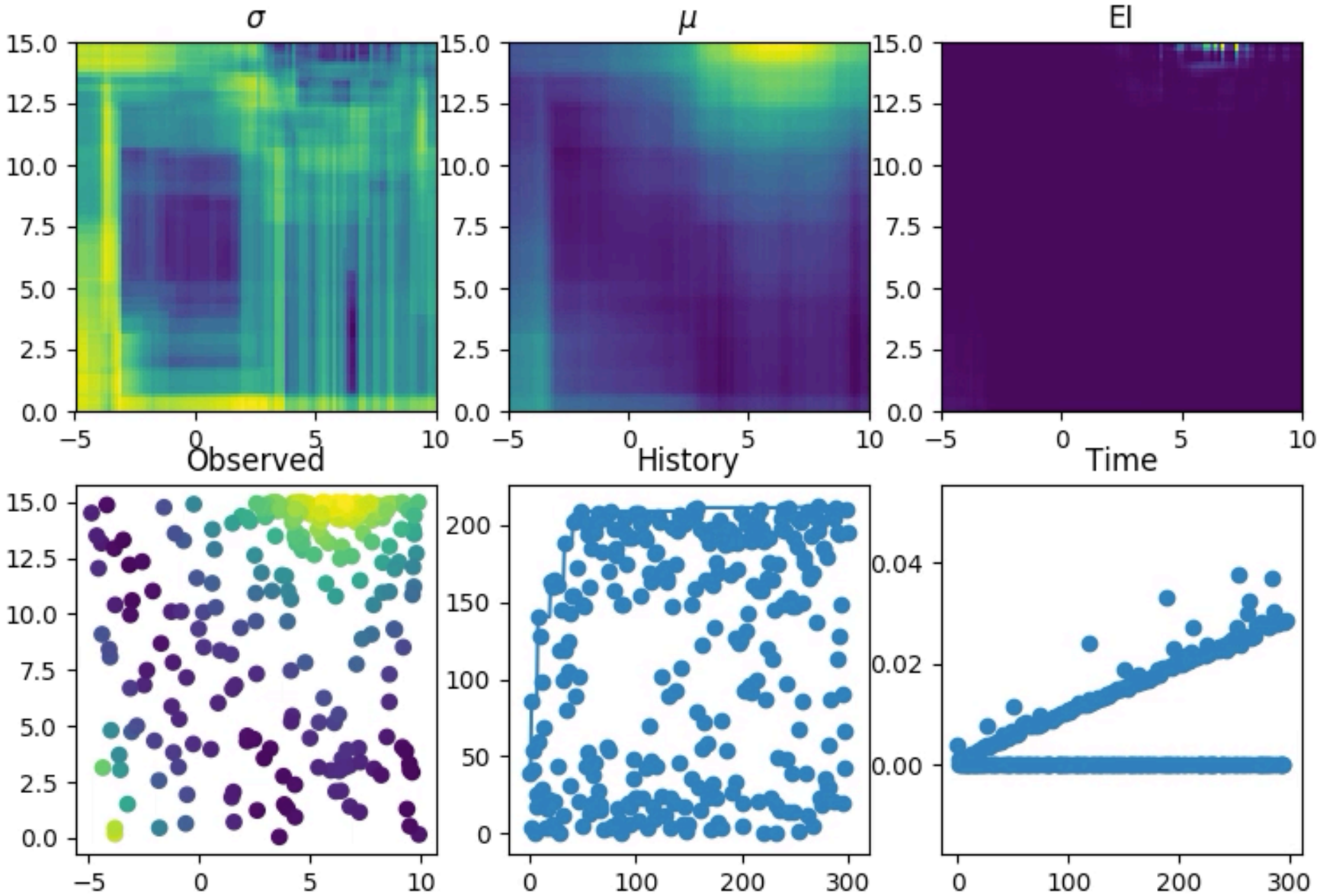


Bayesian Optimization (surrogate: random forests)

Hutter et al. 2011

Feurer et al. 2015

- Use random forest variance
- Scales well to many hyperparameters
- Used in Auto-sklearn

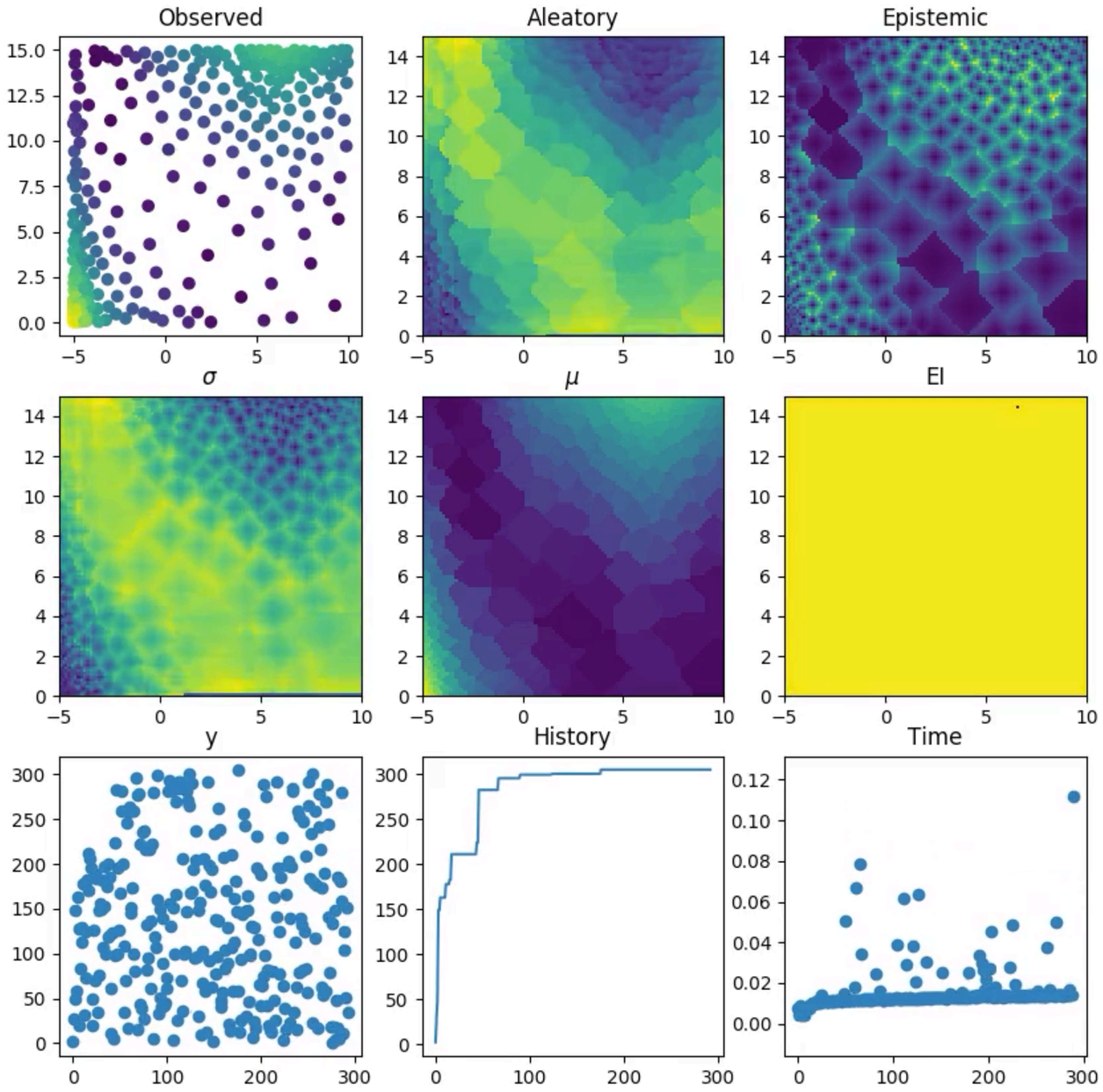


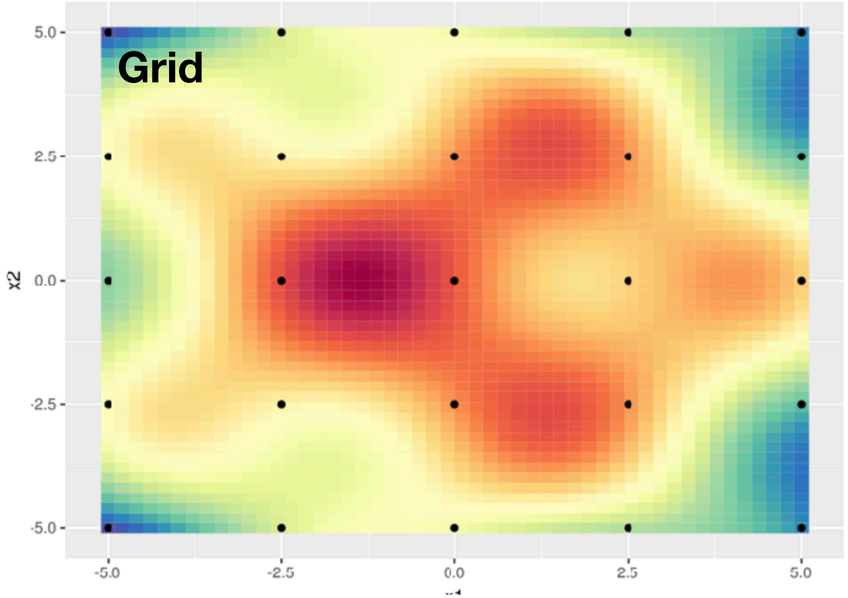
animation by Jeroen van Hoof

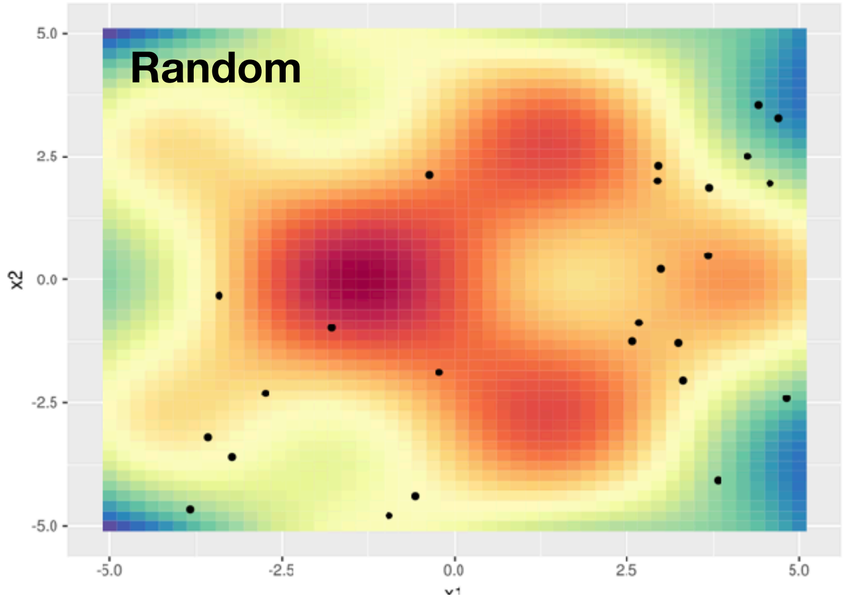
van Hoof & Vanschoren 2020

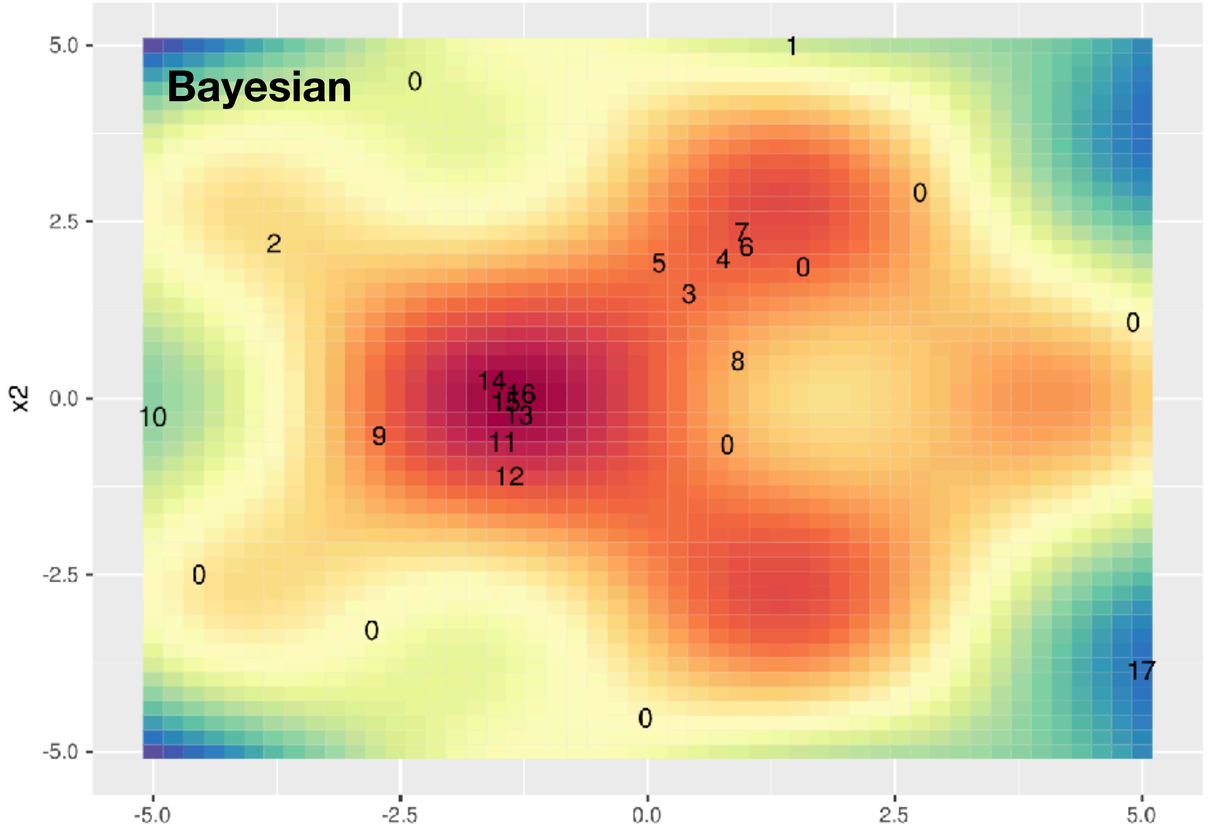
Bayesian Optimization (surrogate: gradient boosting)

- Let gradient boosting predict the mean + upper and lower quantile
  - Faster
  - More robust to concept drift




Grid
x2
5.0
2.5
0.0
-2.5
-5.0
-5.0
-2.5
0.0
2.5
5.0


Random
x2
5.0
2.5
0.0
-2.5
-5.0
-5.0
-2.5
0.0
2.5
5.0

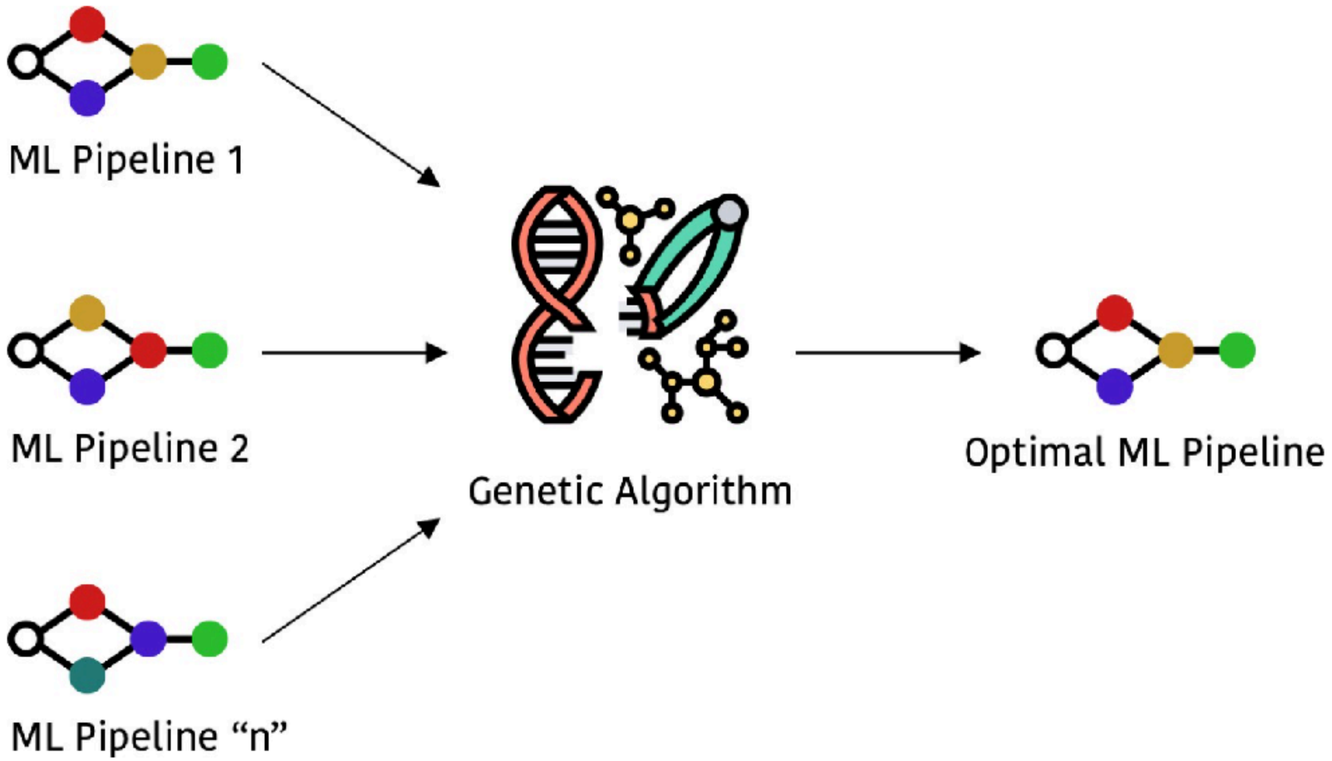

Bayesian
x2
5.0
2.5
0.0
-2.5
-5.0
-5.0
-2.5
0.0
2.5
5.0
1
0
0
2
7
6
5
4
0
3
0
8
14
16
15
13
10
9
11
0
12
0
0
17
0

Evolution

TPOT - *Olson & Moore 2016*

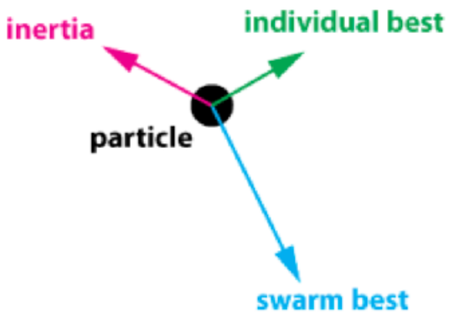
GAMA - *Gijsbers & Vanschoren 2018*

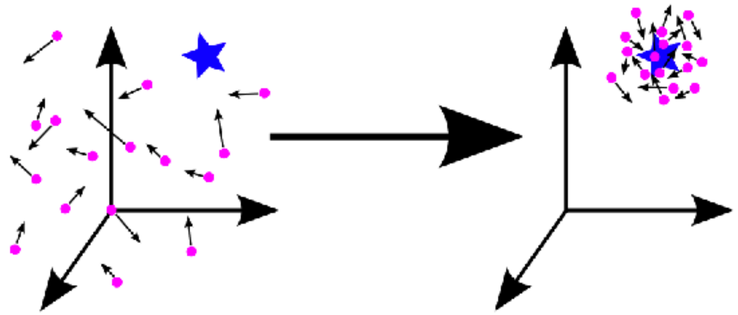
- Start with initial pipeline
- Best pipelines *evolve*: cross-over or mutation
- No fixed pipeline length: adapts to complexity of the problem
- Used in GAMA, TPOT,…

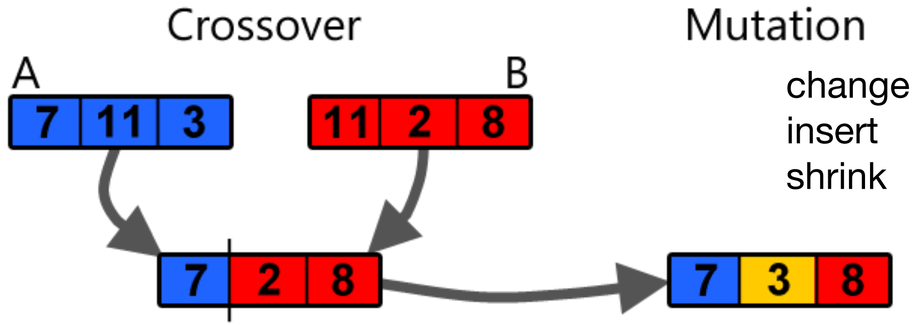


Evolution

- Genetic programming *Olson, Moore 2016, 2019*
  - Mutations: add, mutate/tune, remove HP
- Particle swarm optimization *Mantovani et al 2015*

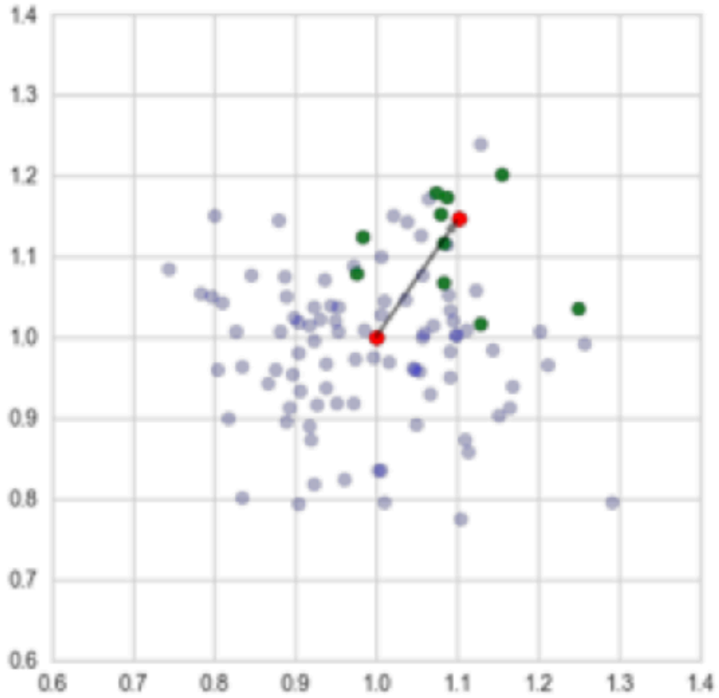






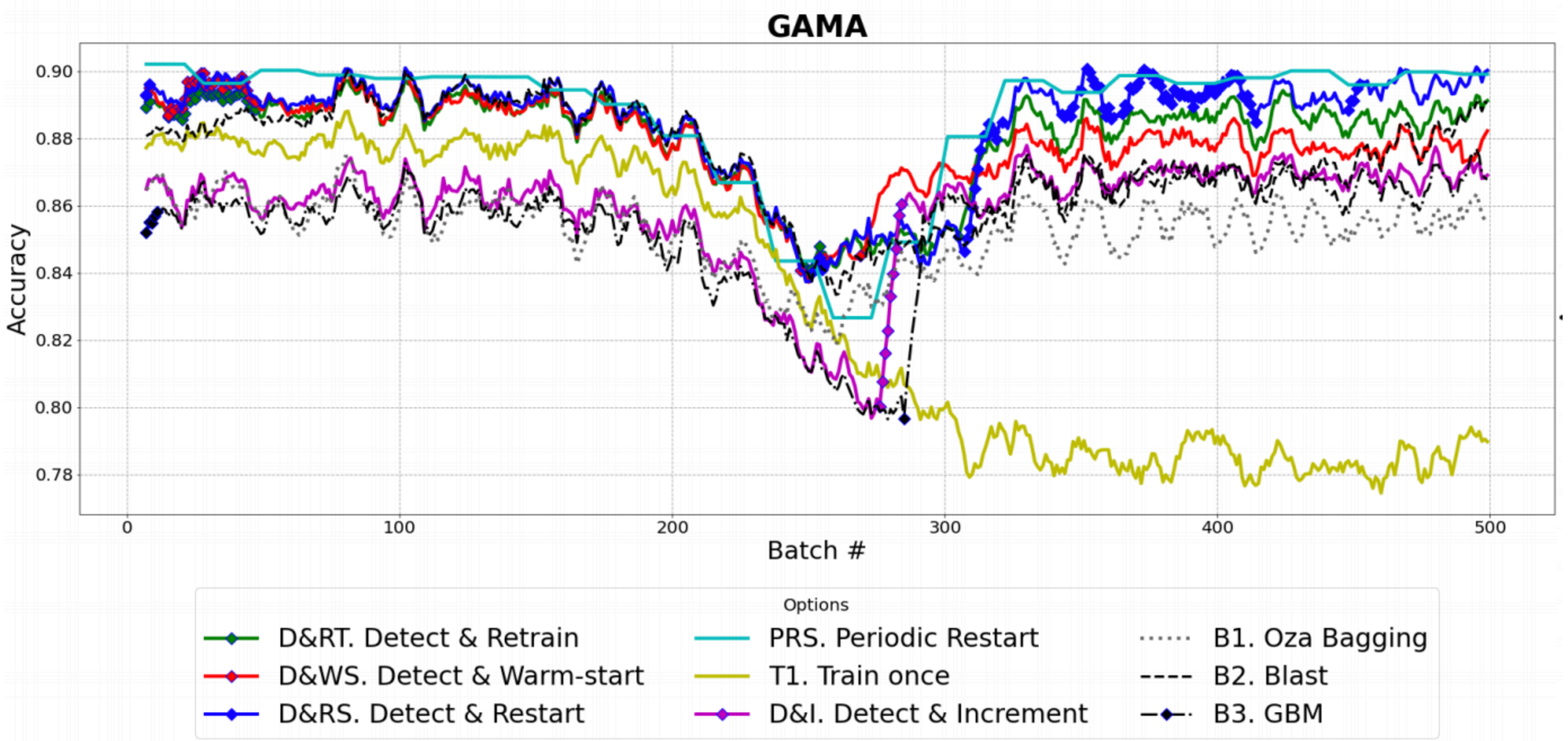
- Covariance matrix adaptation evolution (CMA-ES) *Hansen 2015*
  - Purely continuous, expensive
  - Competitive to optimize deep neural nets

[Loshilov, Hutter 2016]



Evolution

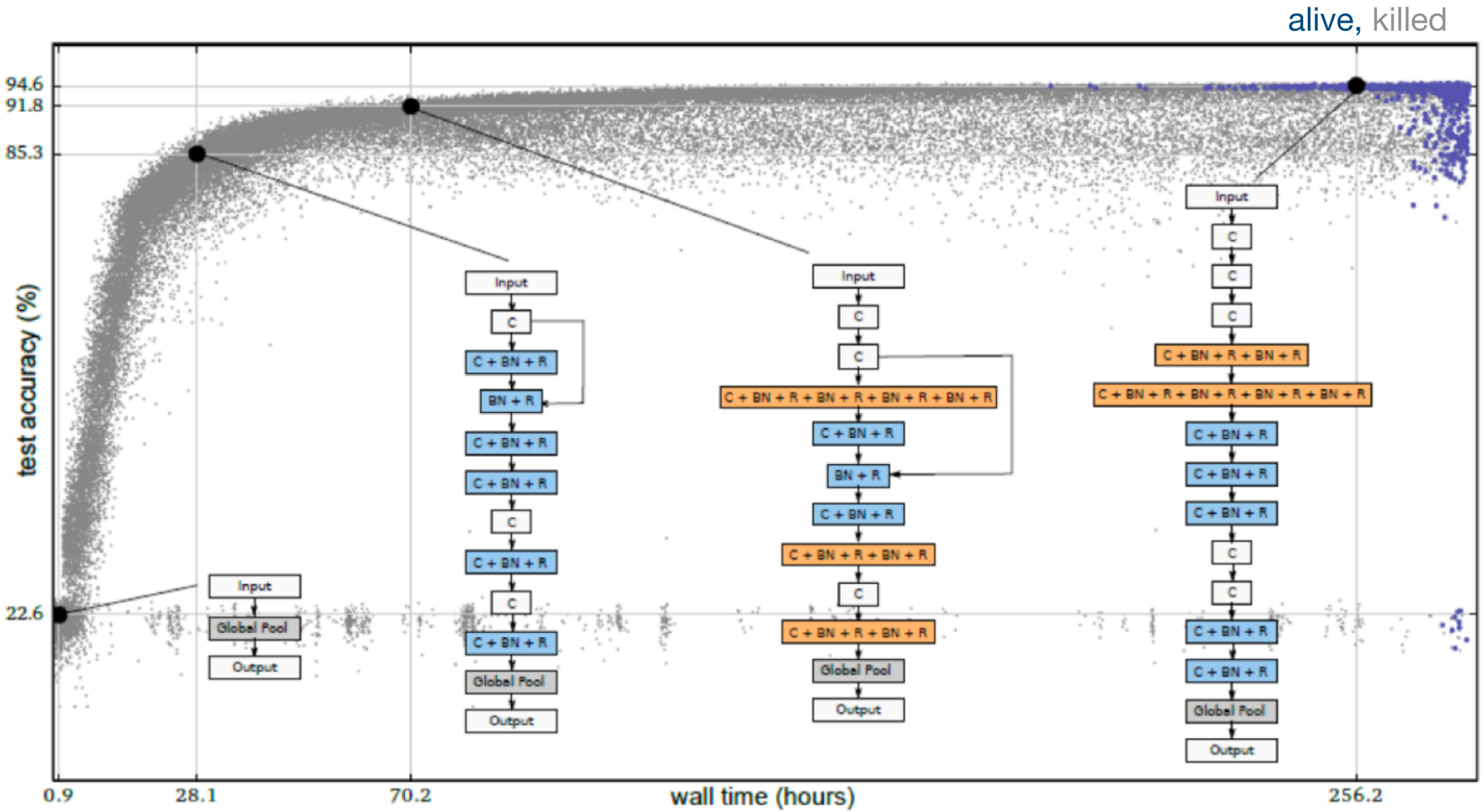
- Less sample efficient, but easy to parallelize, and quickly adapts to changes in the data
  - Allows warm-starting: if the data changes, re-start AutoML but start with best prior pipelines



Neuro-evolution

Real et al 2017
Miikkulainen et al 2017
Wistuba et al 2018

- learn the neural architecture through evolution
- mutations: add, change, remove layer

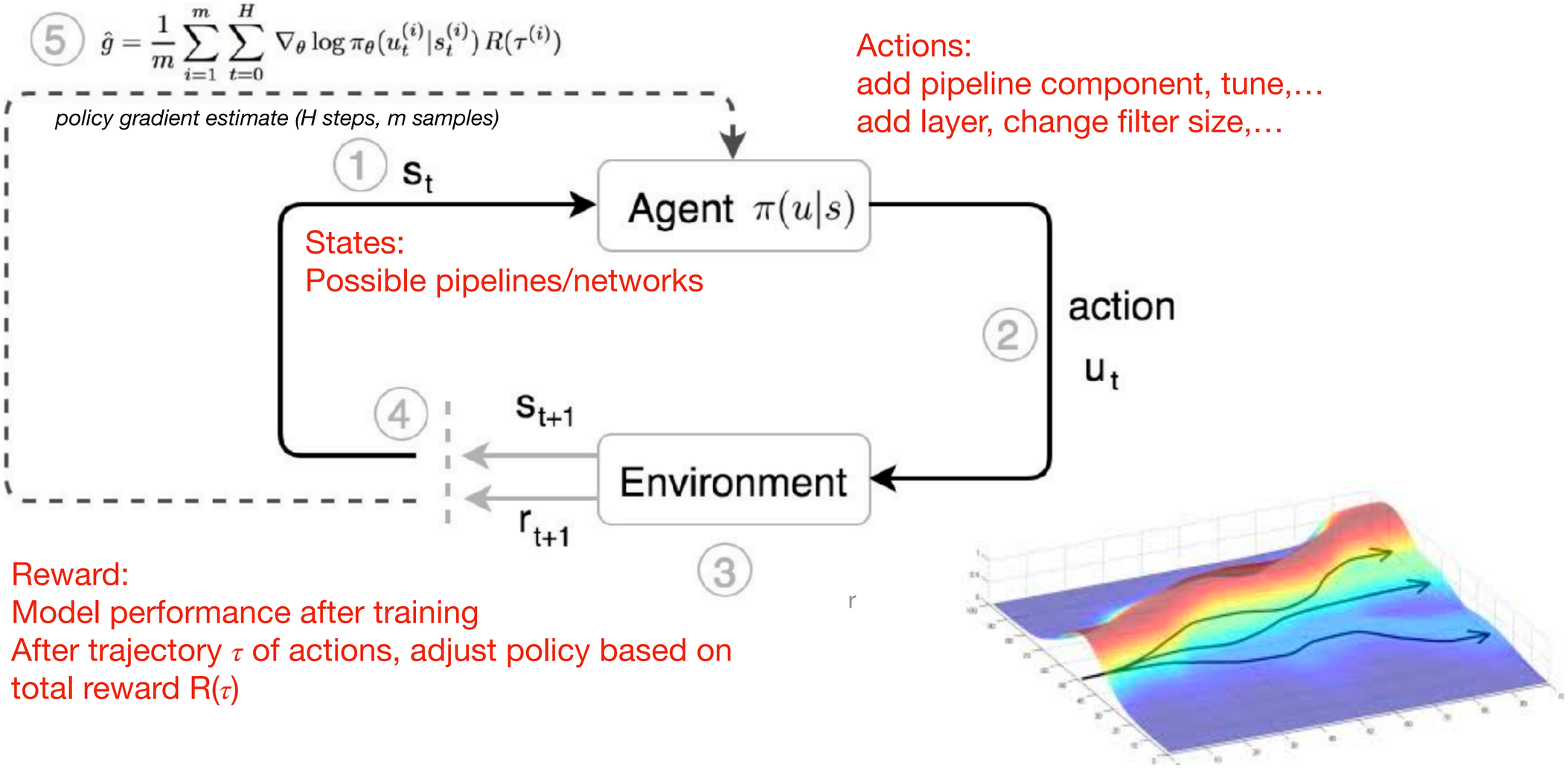


Optimization: Reinforcement learning

Zoph & Le 2017

Levine 2018

Build pipeline or network step-by-step, learn general strategy (policy)



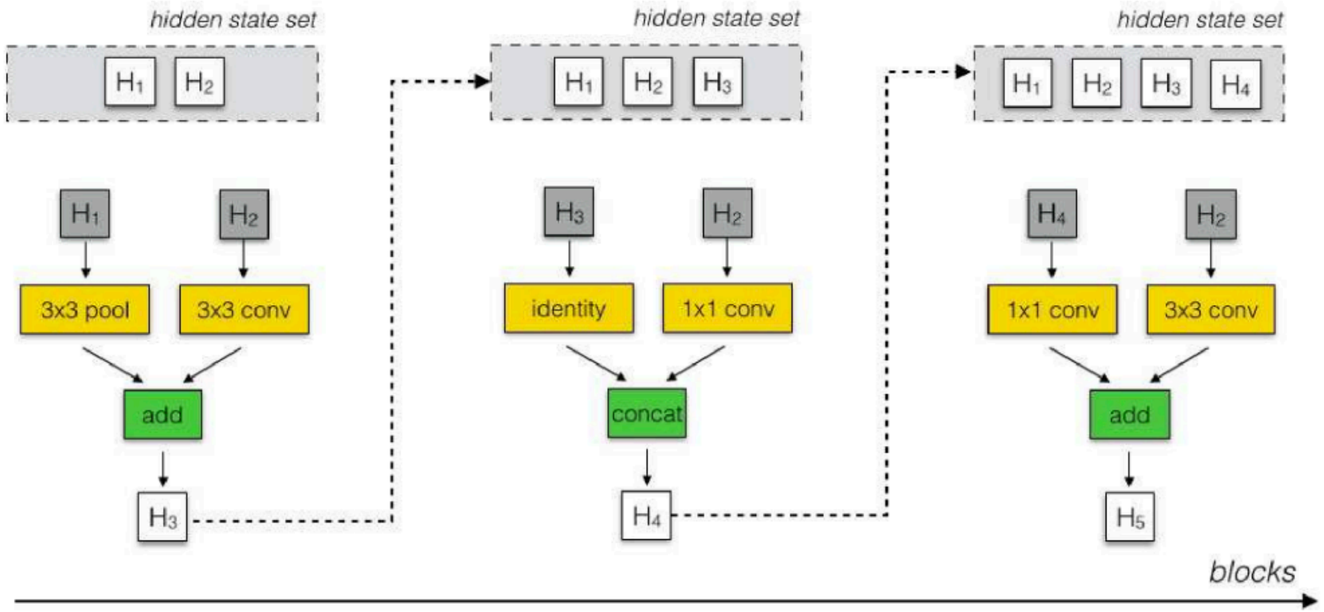
Zoph et al 2018

NAS with Reinforcement learning

1-layer LSTM (PPO), cell space search

- State of the art on ImageNet
- 450 GPUs, 3-4 days, 20000 architectures

- Cell construction:
  - Select existing layers (hidden states, e.g. cell input) H_i to build on
  - Add operation (e.g. 3x3conv) on H_i
  - Combine into new hidden state (e.g. concat, add,…)
  - Iterate over *B* blocks



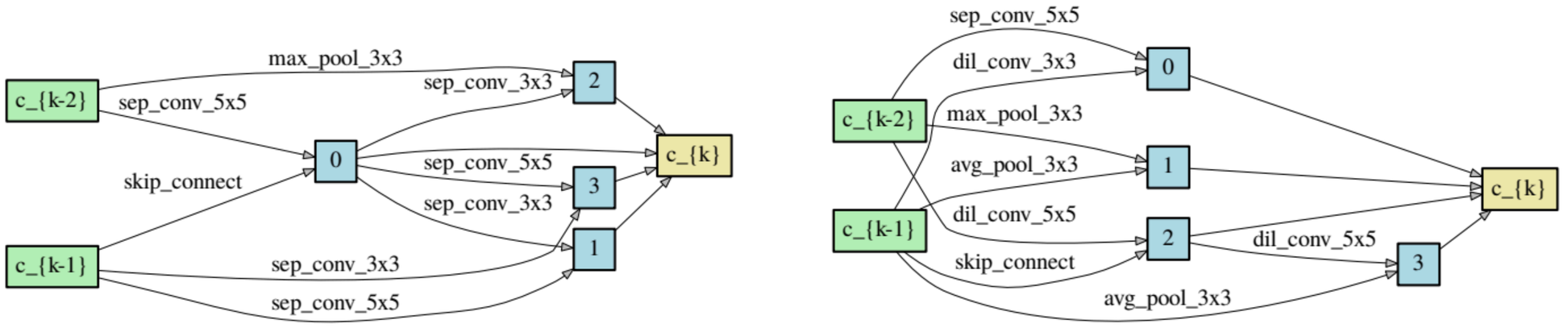
Sidenote: NAS with random search?

Li & Talwalkar 2019

Yu et al. 2019

Real et al. 2019

If you constrain the search space enough, you can get SOTA results with random search!

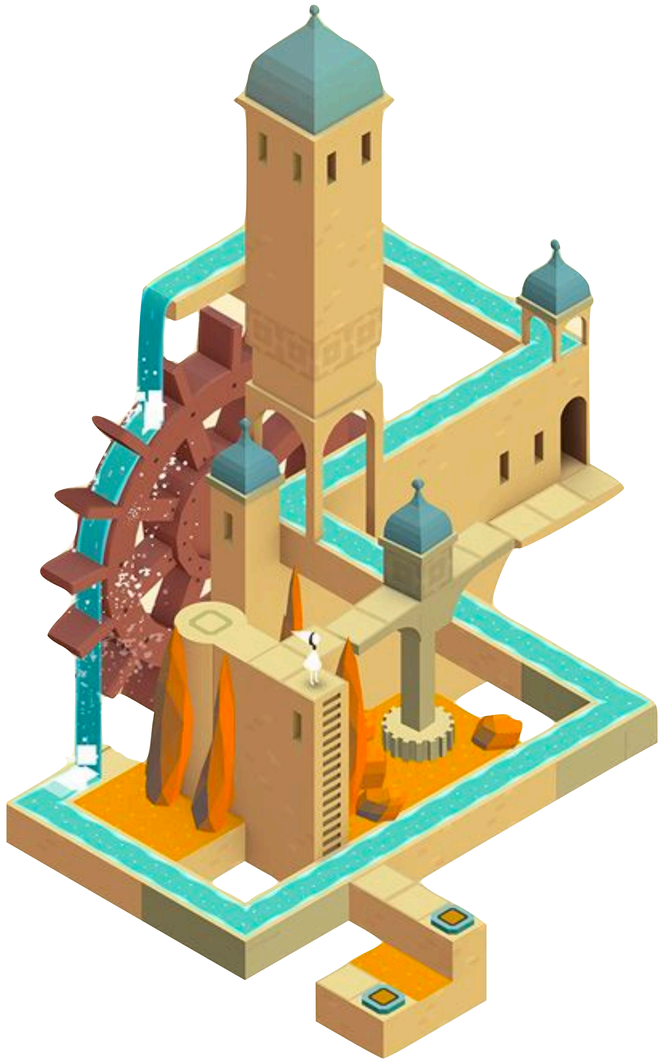


(a) Normal Cell

(b) Reduction Cell

Convolutional Cells on CIFAR-10 Benchmark: Best architecture found by random search with weight-sharing.

Overview



Part 3: Learning how to do AutoML

Closing the loop

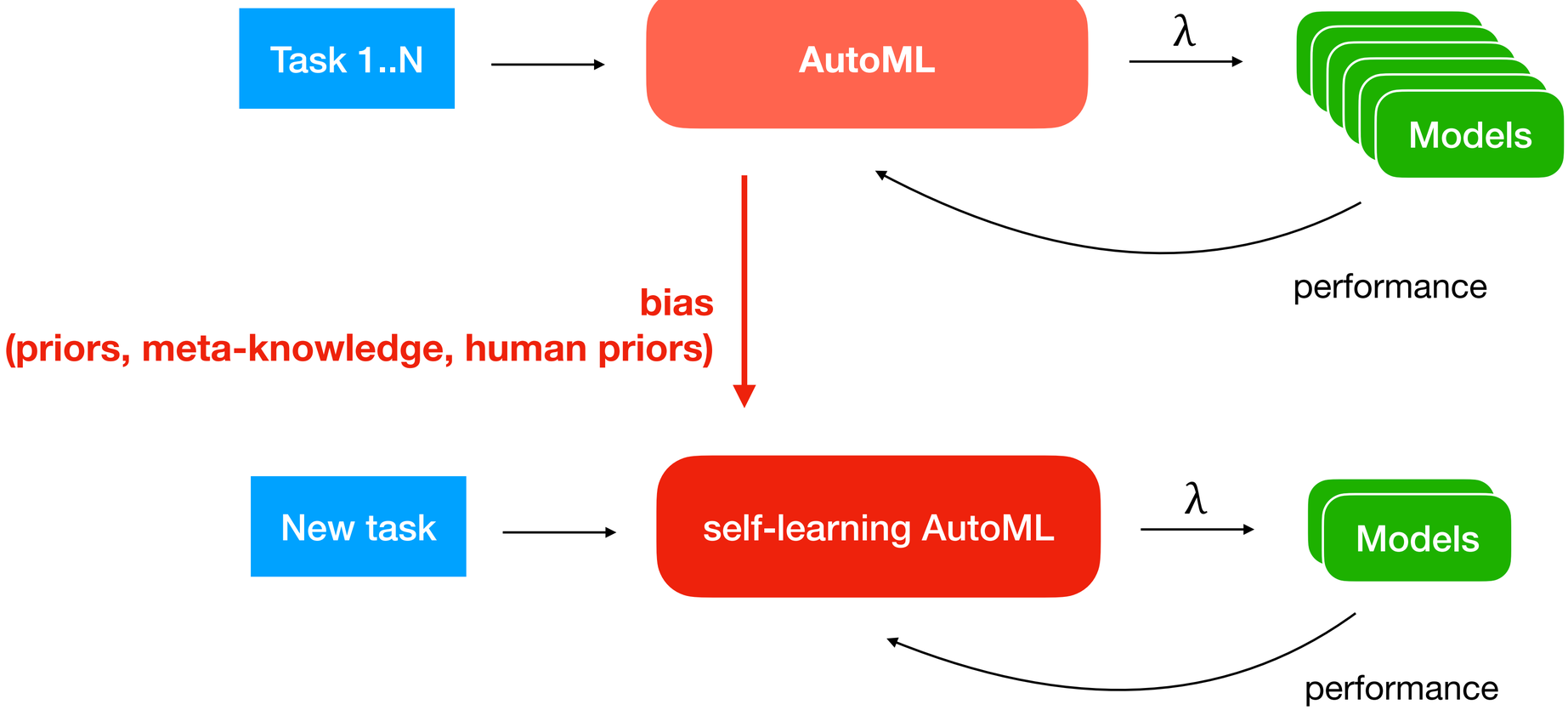
Hutter et al. 2019

AutoML + meta-learning

Meta-learn how to design architectures/pipelines and tune hyper parameters

Human data scientists also learn from experience

Search space can be huge!

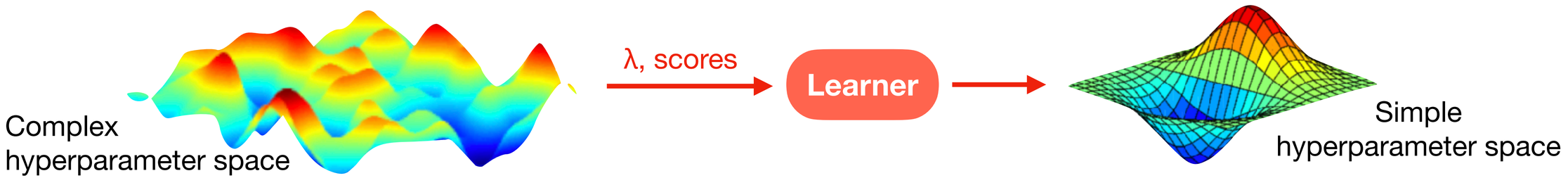


Vanschoren 2018

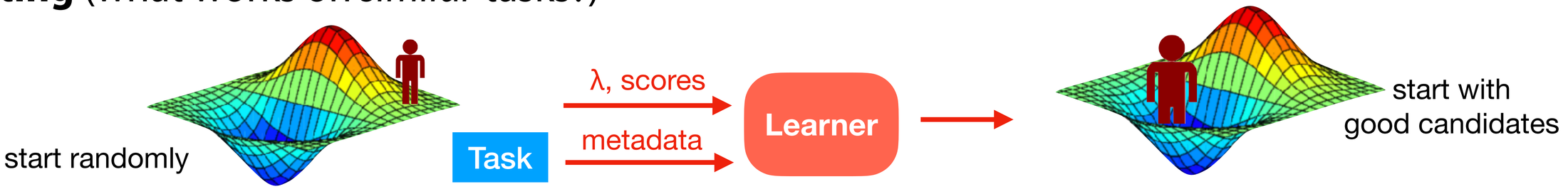
Meta-learning for AutoML: how?

hyperparameters = architecture + hyperparameters

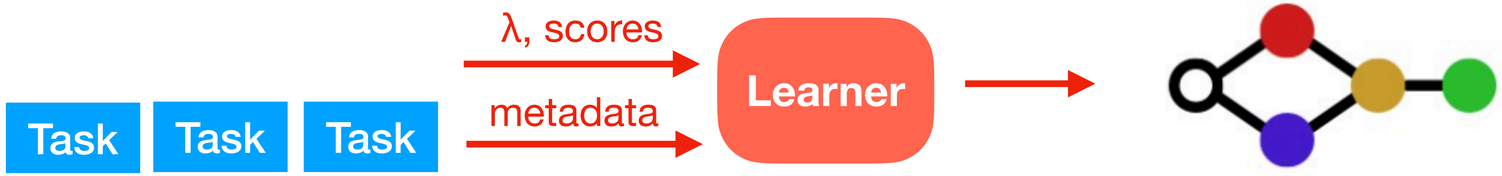
Learning hyperparameter priors



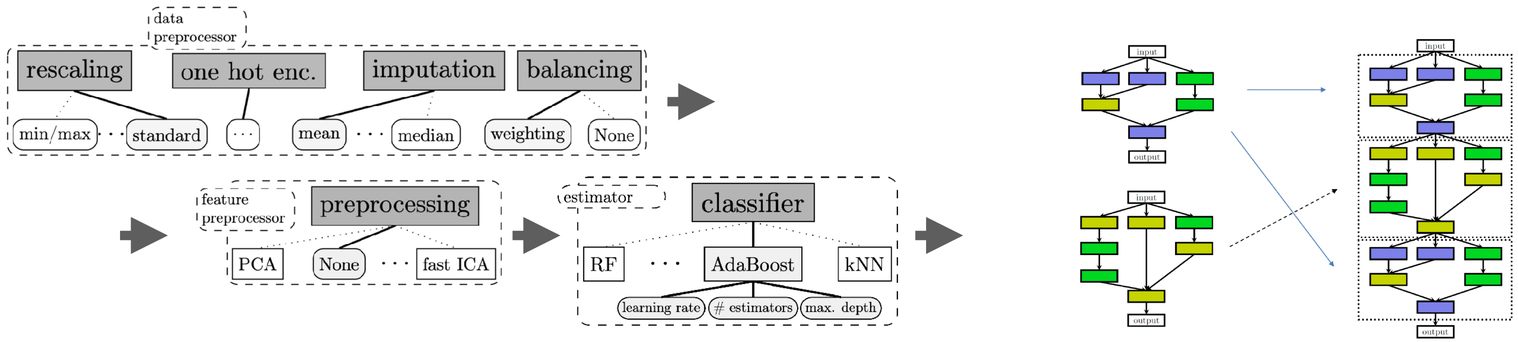
Warm starting (what works on *similar* tasks?)



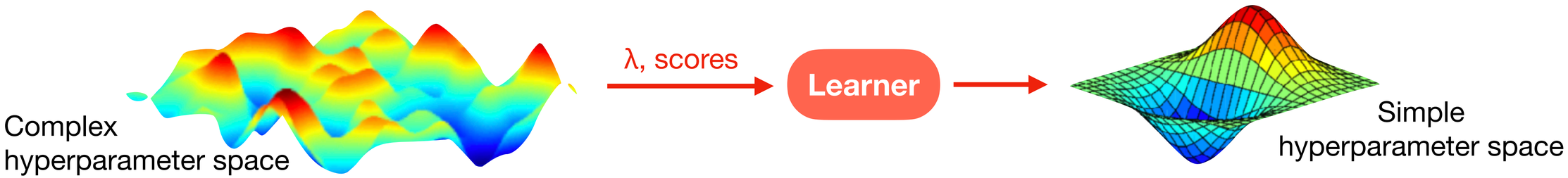
Meta-models (learn how to build models/components)



Observation:
current AutoML strongly depends on learned priors



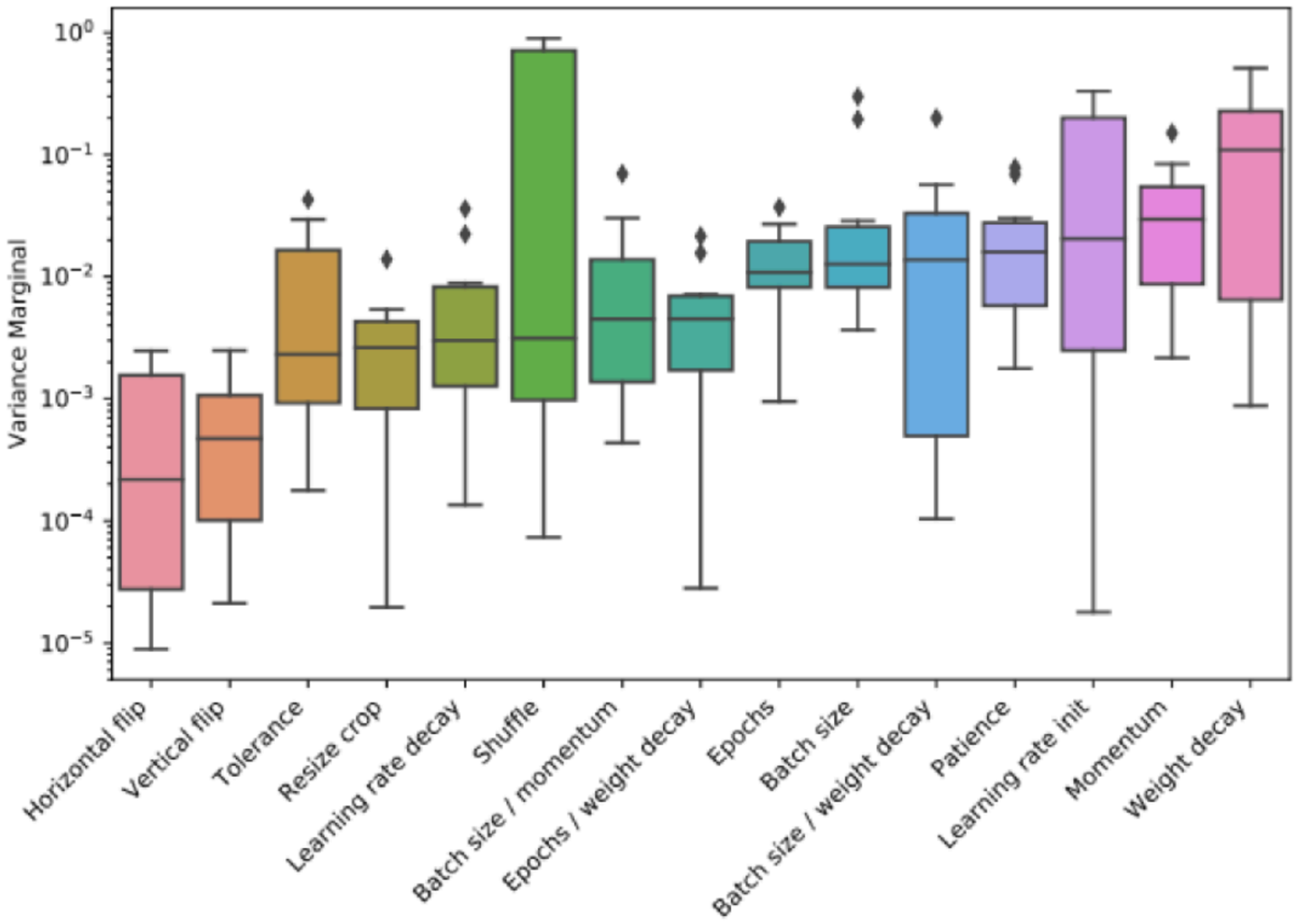
Can we *learn* good hyperparameter priors?



[1] van Rijn & Hutter 2018

Learn hyperparameter importance

- **Functional ANOVA** [1]
  - Select hyperparameters that cause variance in the evaluations.
  - Useful to speed up black-box optimization techniques



ResNets for image classification

Figure source: van Rijn & Hutter, 2018

Learn defaults + hyperparameter importance

[1] Probst et al. 2018

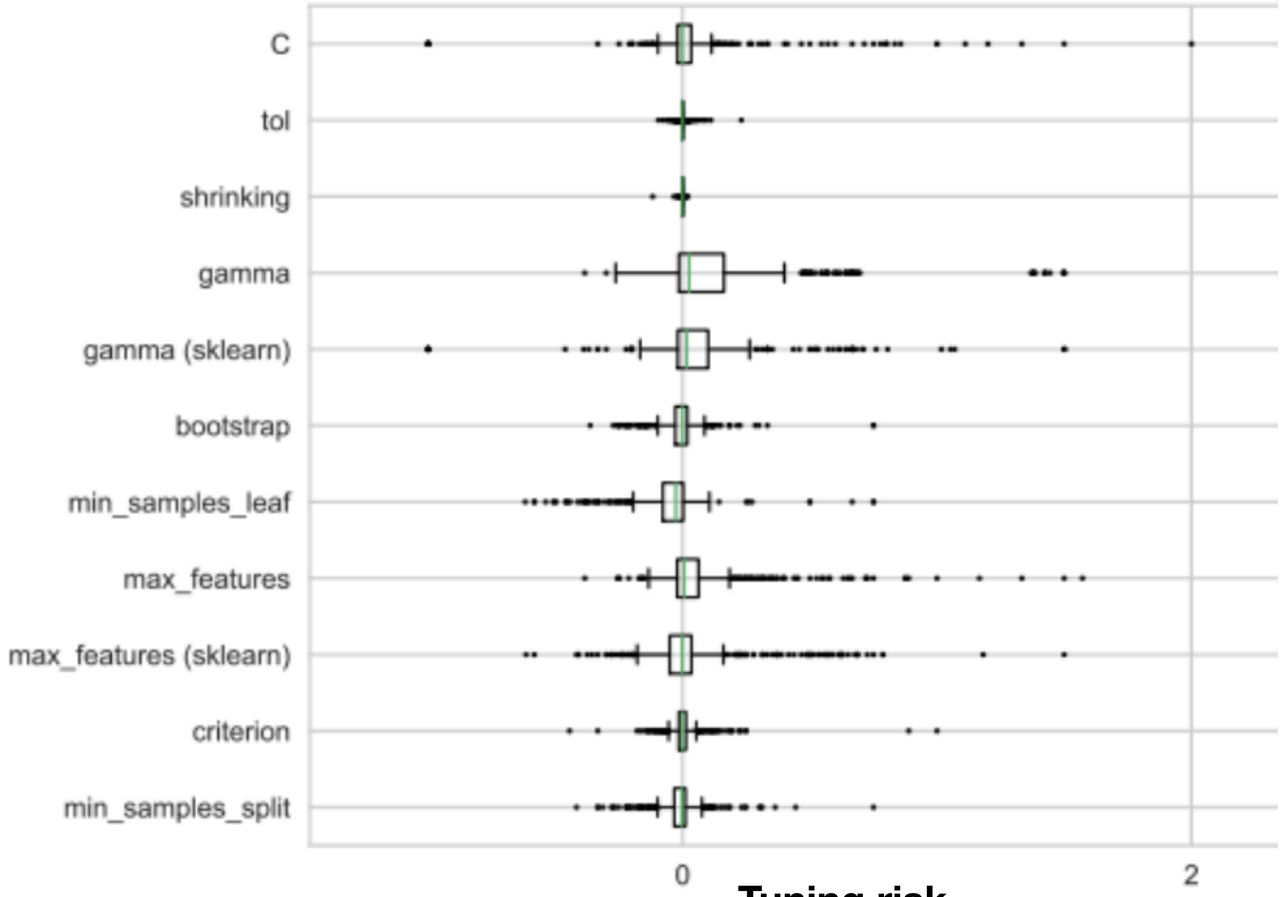
[2] Weerts et al. 2018

[3] van Rijn et al. 2018

- **Tunability** [1,2,3]
  Learn good defaults, measure importance as improvement via tuning

function
max_features
m = 0.16*p
m = p ^0.74
m = 1.15^sqrt(p)
m = sqrt(p)
gamma
m = 0.00574*p
m = 1/p
m = 0.006

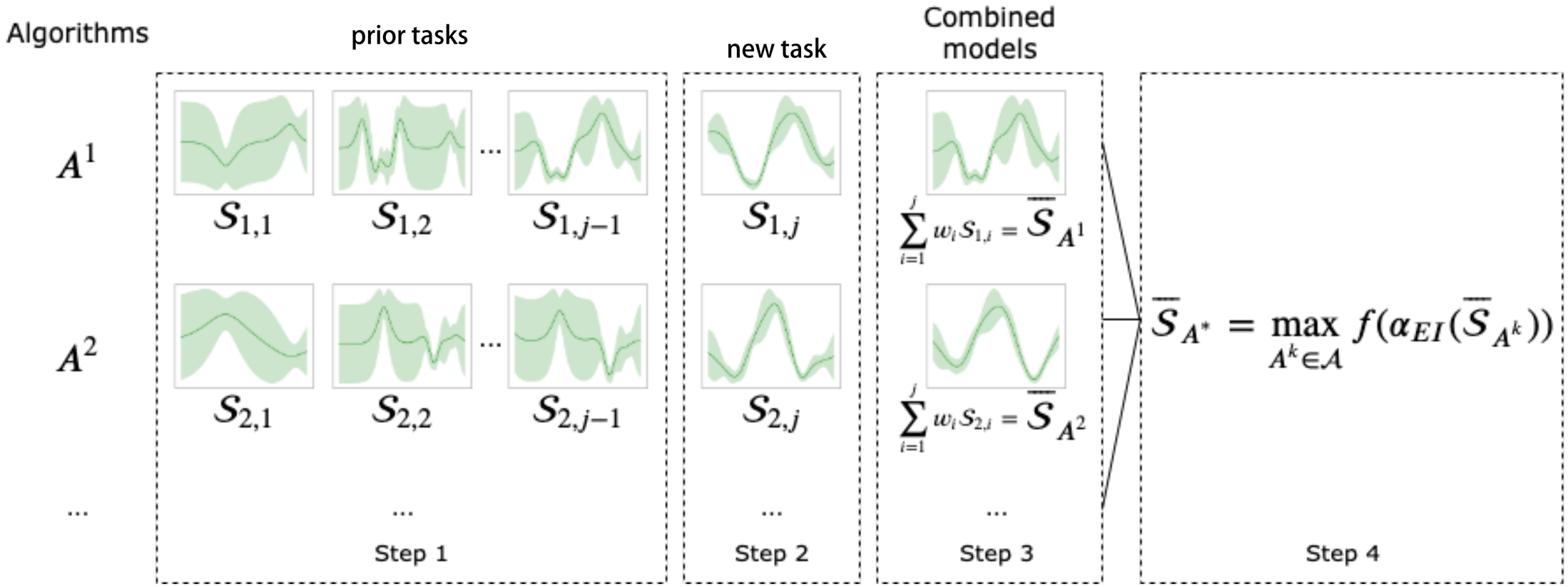
Learned defaults



Tuning risk

Surrogate model transfer

- If task j is *similar* to the new task, its surrogate model S_j will likely transfer well
- Sum up all S_j predictions, weighted by task similarity
- Build combined surrogate, *weighted by current performance* on new task[2]

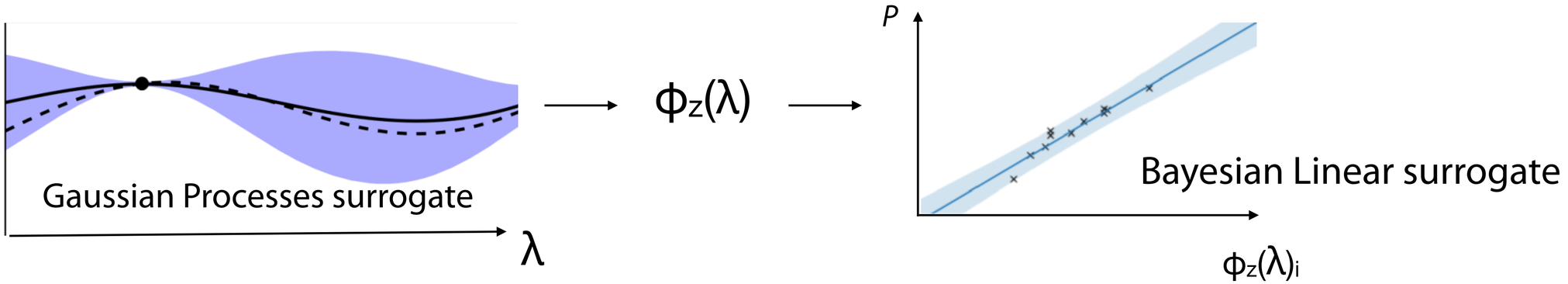


Perrone et al. 2018

Learn basis expansions for hyperparameters

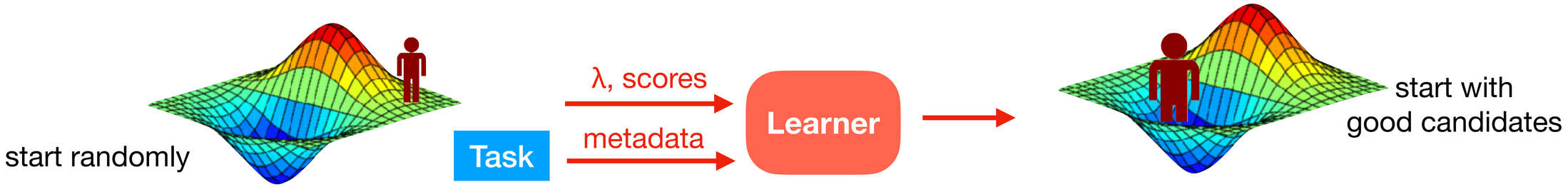
- Hyperparameters can interact in very non-linear ways
- Use a neural net to learn a suitable transform $\phi_z(\lambda)$ so that they behave linearly
- Used in SageMaker AutoML

Learn basis expansion on lots of data (e.g. OpenML)



Warm starting

(what works on *similar* tasks?)



How to measure task similarity?

- Hand-designed (statistical) meta-features that describe (tabular) datasets [1]
- Task2Vec: task embedding for image data [2]
- Optimal transport: similarity measure based on comparing probability distributions [3]
- Metadata embedding based on textual dataset description [4]
- Dataset2Vec: compares batches of datasets [5]
- Distribution-based invariant deep networks [6]

Figure source: Alvarez-Melis et al. 2020

[1] Vanschoren 2018
[2] Achille et al. 2019
[3] Alvarez-Melis et al. 2020
[4] Drori et al. 2019
[5] Jooma et al. 2020
[6] de Bie et al. 2020

Feurer et al. 2015

Warm-starting with kNN

- Find k most similar tasks, warm-start search with best λ_i
  - Auto-sklearn: Bayesian optimization (SMAC)
    - Meta-learning yield better models, faster
    - Winner of several AutoML Challenges

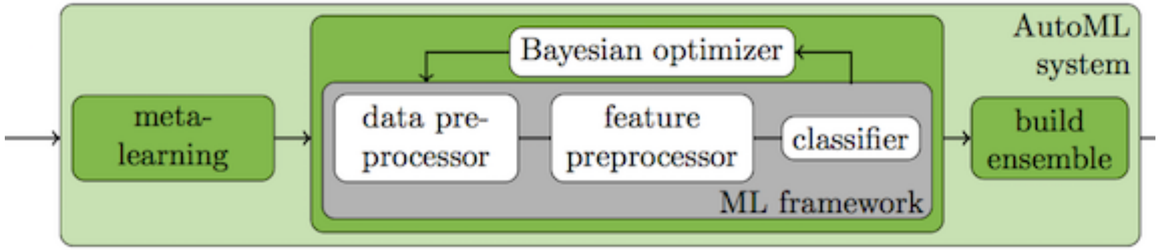
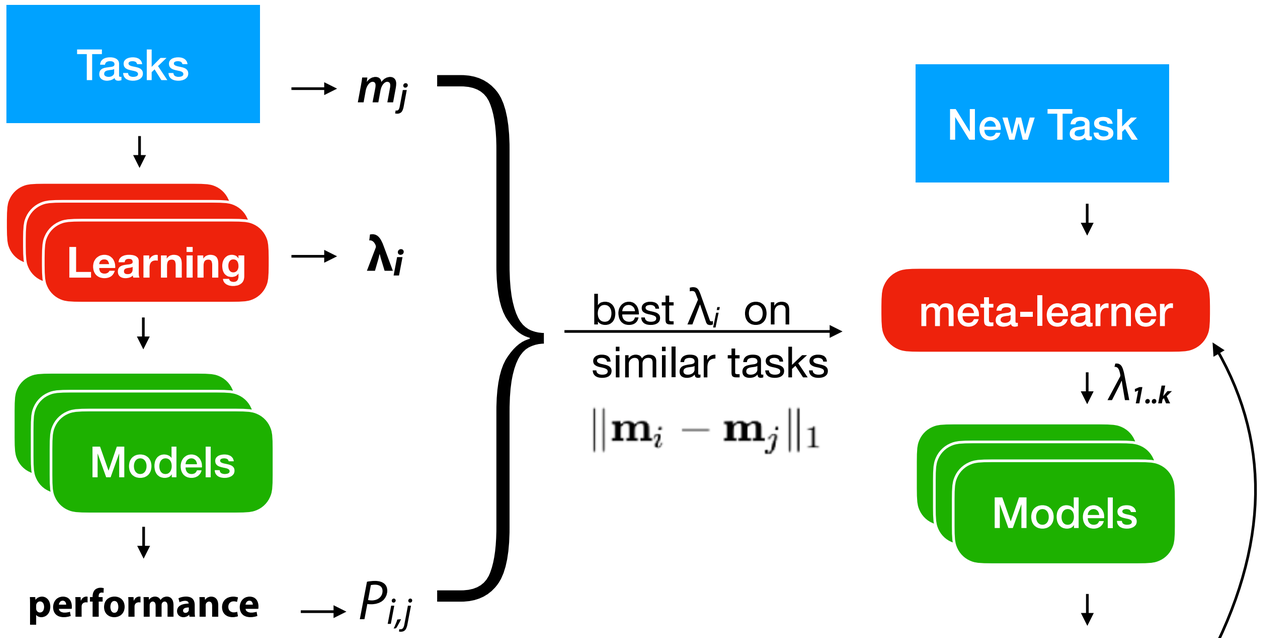
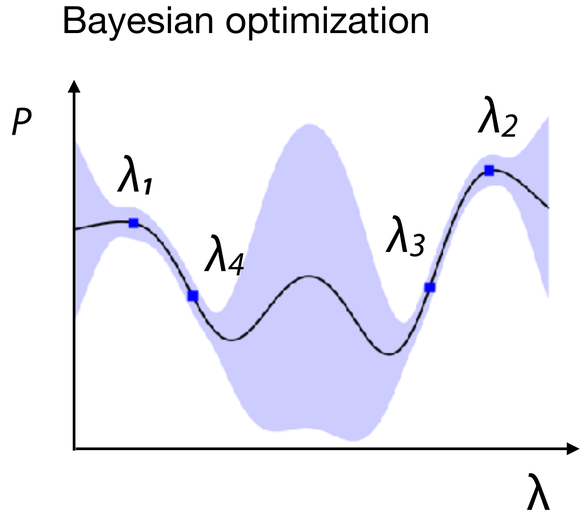


Figure source: Feurer et al., 2015



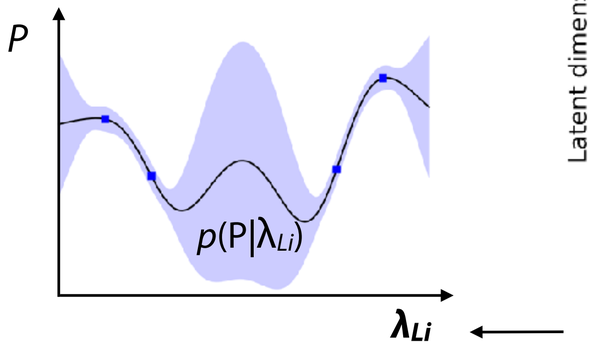


Probabilistic Matrix Factorization

Fusi et al. 2017

- Collaborative filtering: configurations λ_i are `rated' by tasks t_j

- Learn latent representation for tasks T and configurations λ
- Use meta-features to warm-start on new task
- Returns probabilistic predictions for Bayesian optitmization
- Used in Azure AutoML



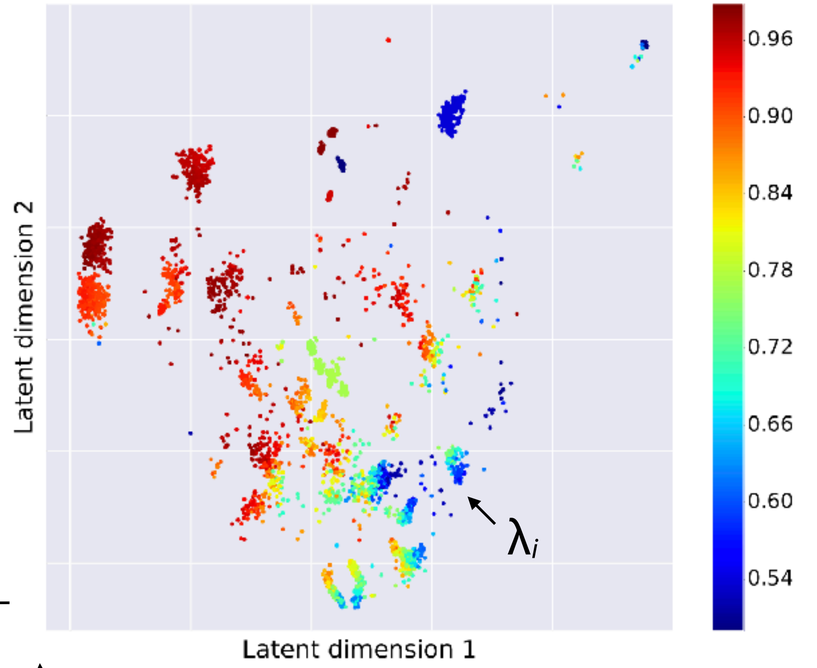
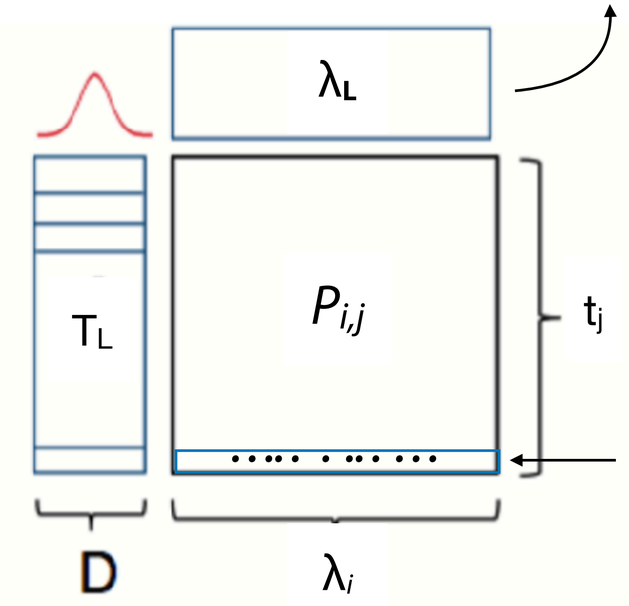


Figure source: Fusi et al., 2017

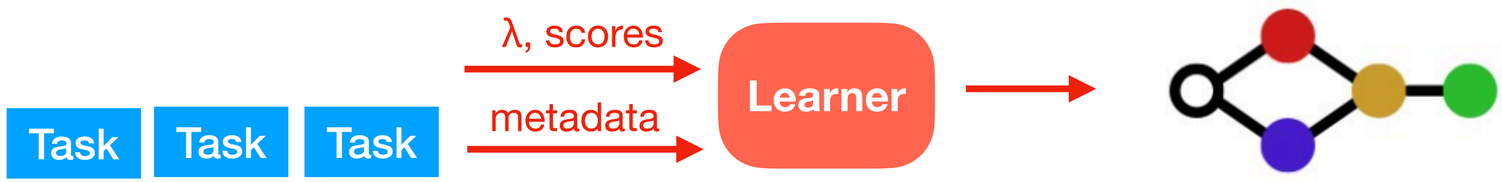


latent representation

t_{new} warm-started with $\lambda_{1..k}$

Meta-models

(learn how to build models/components)



Algorithm selection models

[1] Brazdil et al. 2009, Lemke et al. 2015

[2] Sun and Pfahringer 2013, Pinto et al. 2017

[3] Sanders and C. Giraud-Carrier 2017

[4] Yang et al. 2018

- *Learn* direct mapping between meta-features and $P_{i,j}$
  - Zero-shot meta-models: predict best λ_i given meta-features [1]

$$\boldsymbol{m_j} \rightarrow \text{meta-learner} \rightarrow \lambda_{best}$$

  - Ranking models: return ranking $\lambda_{1..k}$ [2]

$$\boldsymbol{m_j} \rightarrow \text{meta-learner} \rightarrow \lambda_{1..k}$$

  - Predict which algorithms / configurations to consider / tune [3]

$$\boldsymbol{m_j} \rightarrow \text{meta-learner} \rightarrow \boldsymbol{\Lambda}$$

  - Predict performance / runtime for given Θ_i and task [4]

$$\boldsymbol{m_j}, \lambda_i \rightarrow \text{meta-learner} \rightarrow P_{ij}$$

- Can be integrated in larger AutoML systems: warm start, guide search,…

[1] Ramachandran et al. 2017

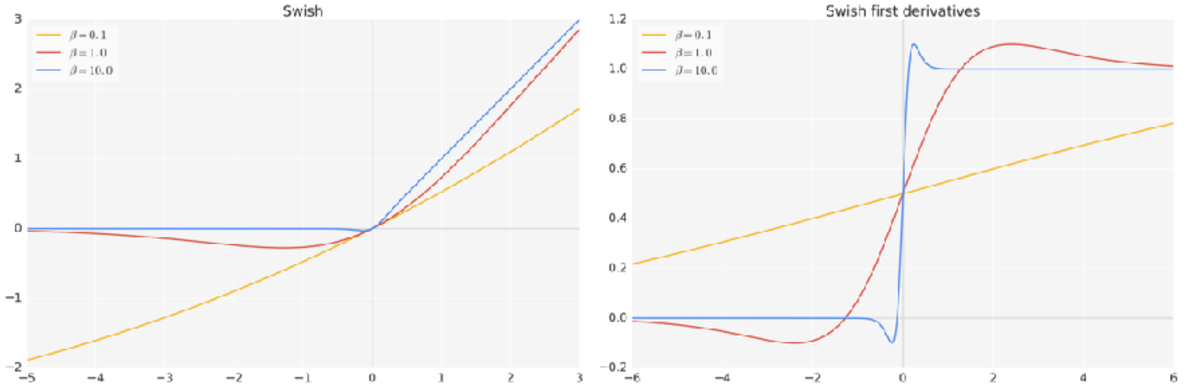
[2] Bello et al. 2017

[3] Volpp et al. 2020

Learning model components

- Learn nonlinearities: RL-based search of space of likely useful activation functions [1]
  - E.g. *Swish* can outperform ReLU

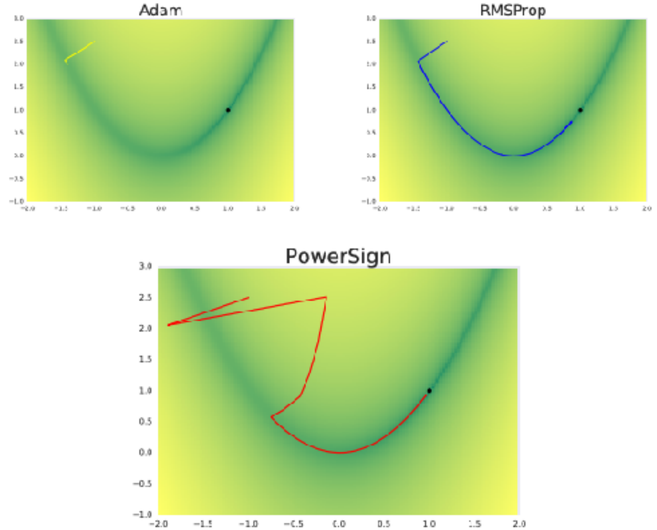
$$Swish : \frac{x}{1 + e^{-\beta x}}$$



- Learn optimizers: RL-based search of space of likely useful update rules [2]
  - E.g. PowerSign can outperform Adam, RMPprop

$$PowerSign : e^{sign(g)sign(m)}g$$

g: gradient, m:moving average



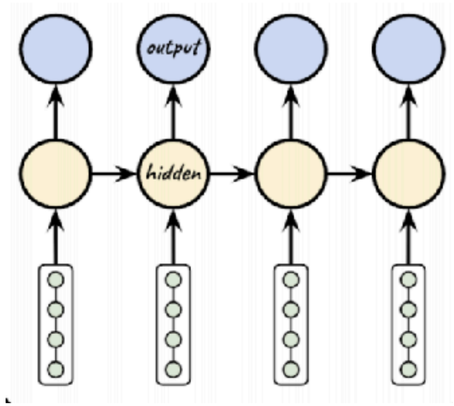
- Learn acquisition functions for Bayesian optimization [3]

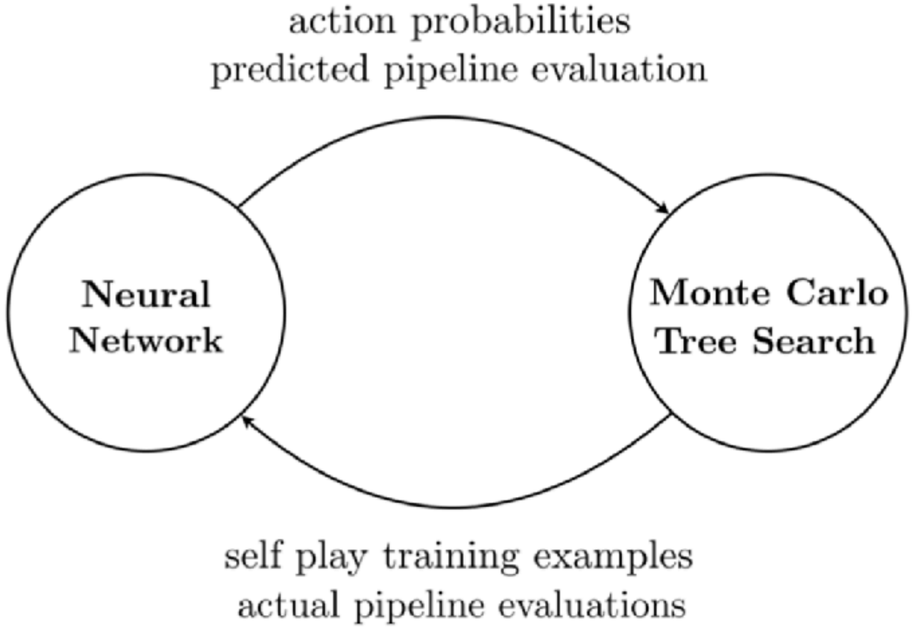
Figure source: Ramachandran et al., 2017 (top), Bello et al. 2017 (bottom)

MOSAIC [Rakotoarison et al. 2019]
AlphaD3M [Drori et al. 2019]

Monte Carlo Tree Search + reinforcement learning

- ***Self-play:***
  - Game actions: insert, delete, replace components in a pipeline
  - Monte Carlo Tree Search builds pipelines given action probabilities
  - Neural network (LSTM) Predicts pipeline performance





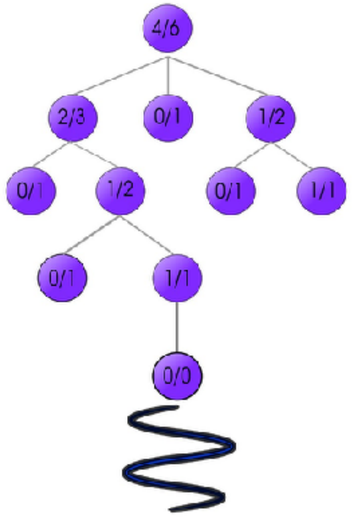


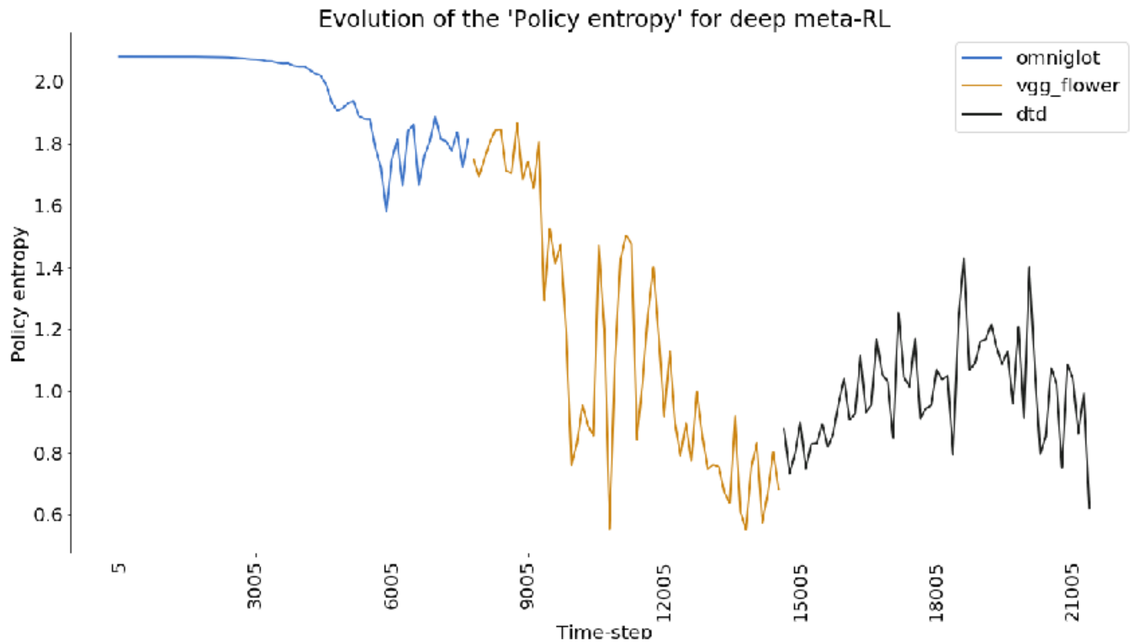
Figure source: Drori et al., 2019

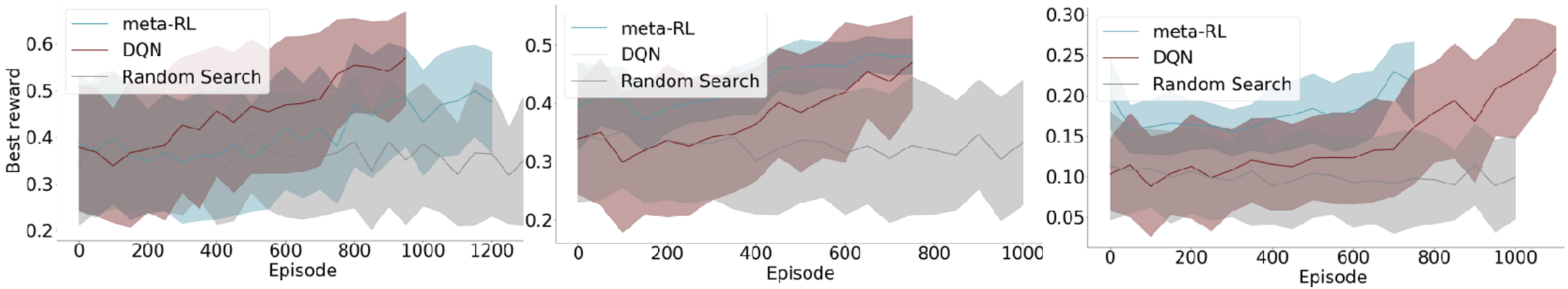
Gomez & Vanschoren, 2019

Meta-Reinforcement Learning for NAS

Results on increasingly difficult tasks:

- Initially slower than DQN, but faster after a few tasks
- Policy entropy shows learning/re-learning





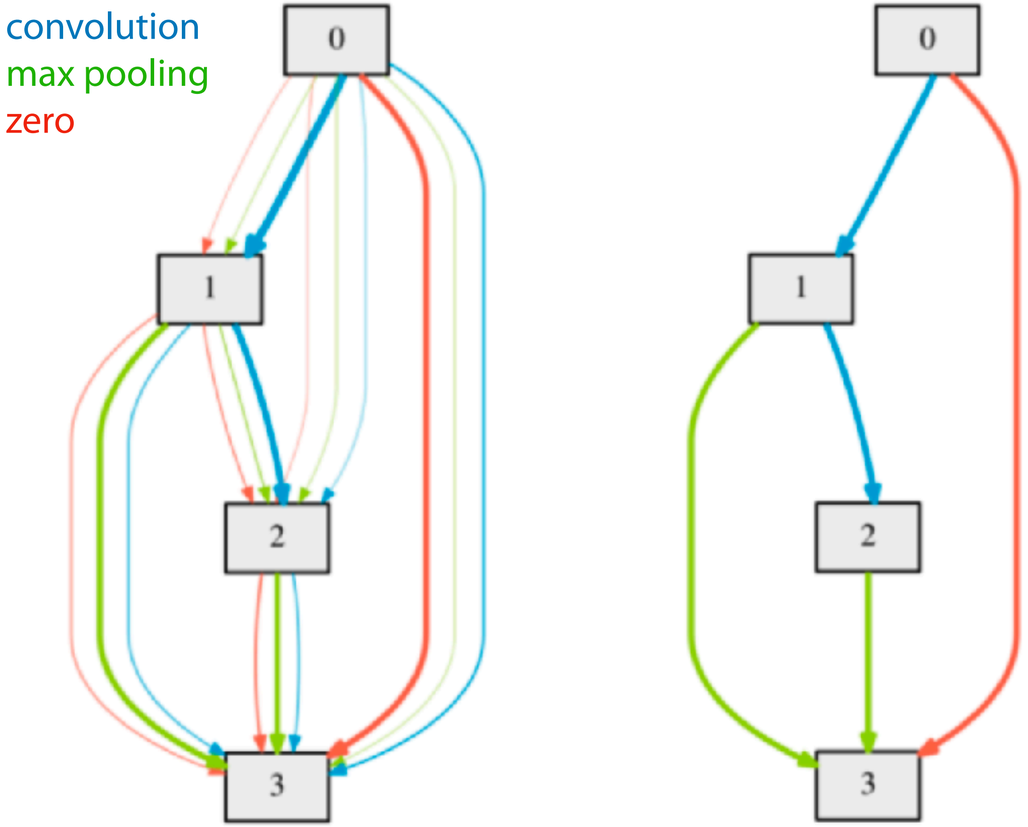
omniglot vgg_flower dtd

Elsken et al., 2020

MetaNAS: meta-learning + NAS

- Use meta-learning (MAML) to learn a good weight initialization for the network

Meta-learn initial operator weights α_i from previous tasks

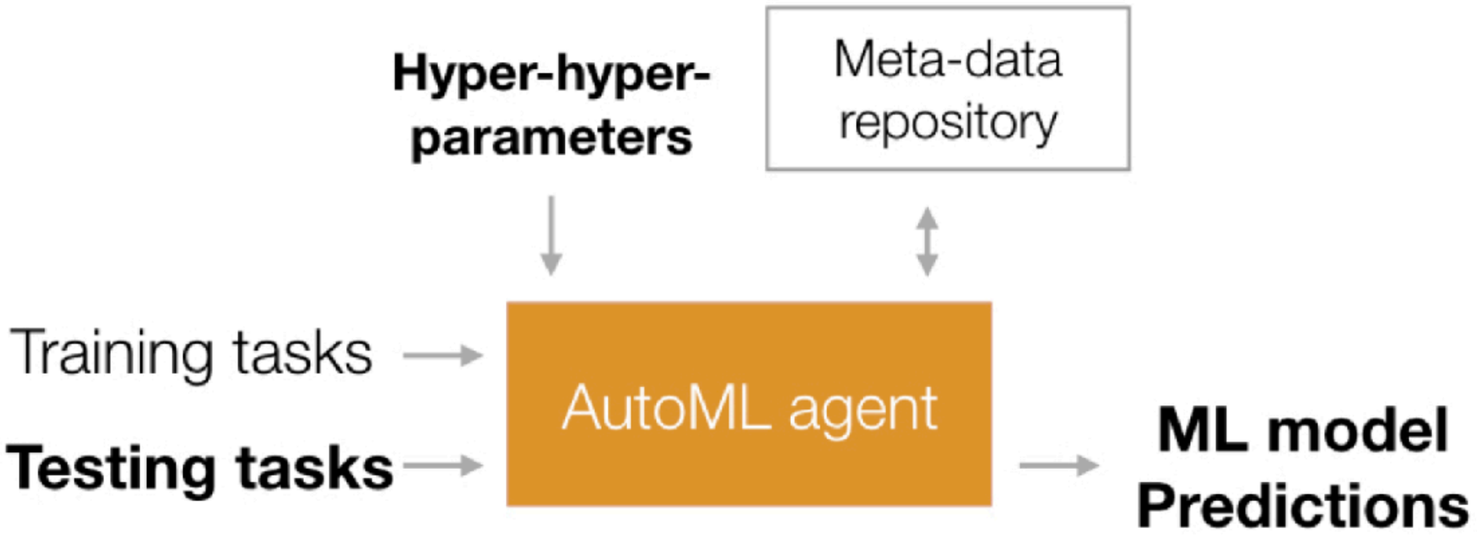


interleaved optimization of α_i and ω_j with SGD

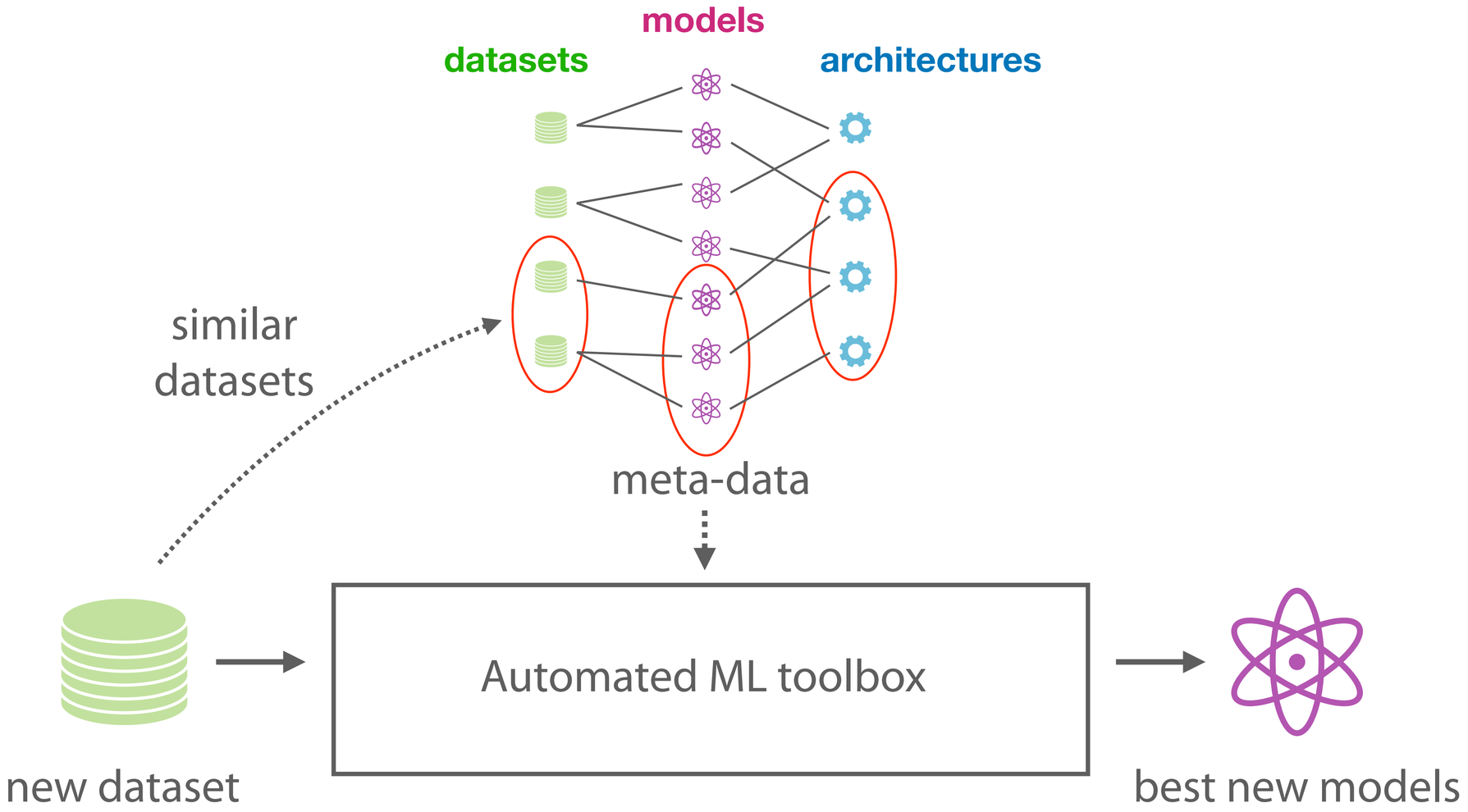
argmax α_i

Meta-learning AutoML in practice

- We need a meta-data repository of prior machine learning datasets (tasks) and experiments
  - e.g. OpenML.org
- Ideally, a *shared* memory that all AutoML tools can access



Meta-learning with OpenML



AutoML *open source* tools

	Architect. search	Operators	Hyperpar. search	Improvements	Metalearning
Auto-WEKA	Param. pipeline	WEKA	Bayesian Opt. (RF)		
auto-sklearn	Param. pipeline	sklearn	Bayesian Opt. (RF)	Ensemble	warm-start
mlr-mbo	Param. pipeline	mlr	Bayesian Opt.	multi-obj.	
BO-HB	Param. pipeline	sklearn	Tree of Parzen Estim.	Ensemble, HB	
hyperopt-sklearn	Param. pipeline	sklearn	Tree of Parzen Estim.		
skopt	Param. pipeline	sklearn	Bayesian Opt. (GP)		
TPOT	Evolving pipelines	sklearn	Population-based		
GAMA	Evolving pipelines	sklearn	Population-based	Ensemble, ASHA	
H2O AutoML	Param. pipeline	H2O	Random search	Stacking	
AutoGluon-Tabular	Param. pipeline	Sagemaker	Random search	multi-level Stacking	
OBOE	Single algorithms	sklearn	Low rank approx.	Ensembling	runtime pred
Auto-Keras	Param. NAS	keras	Bayesian Opt.	Net Morphisms	
Auto-pyTorch	Param. pipeline	pyTorch	BO-HB		
TensorFlow 2	/	keras	RS or HB		
Talos	/	keras	RS variants		

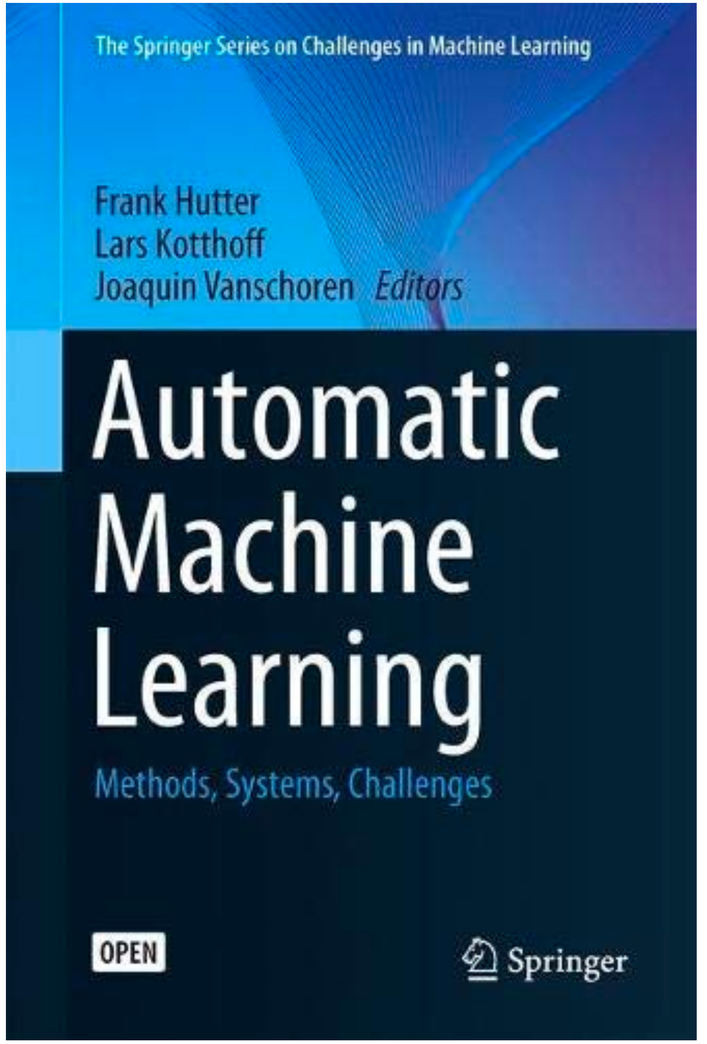
Many other tools for hyperparameter optimization alone

Further reading

Open access book

PDF (free): www.automl.org/book

www.amazon.de/dp/3030053172



Thank you!

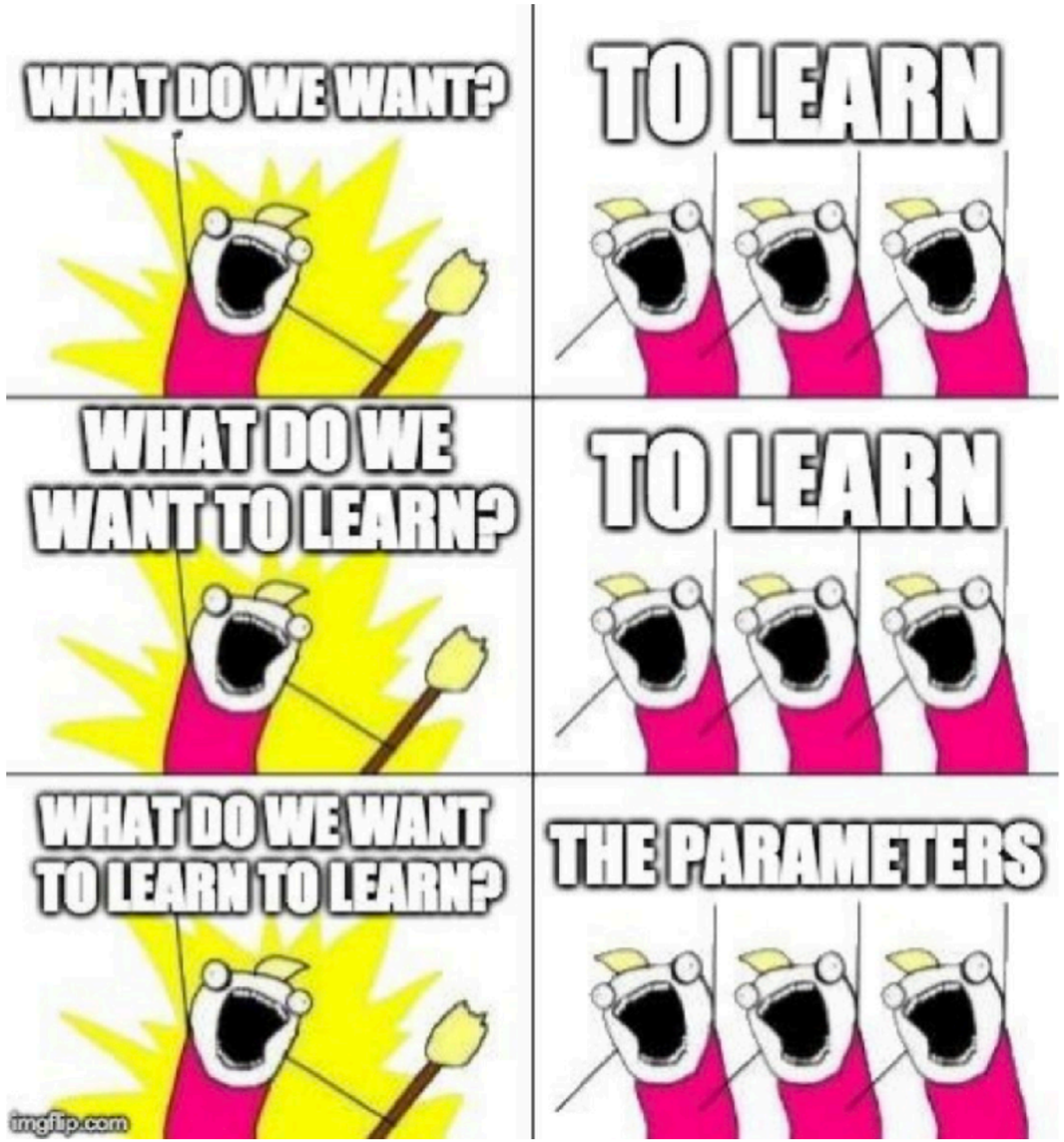


Image: Pesah et al. 2018