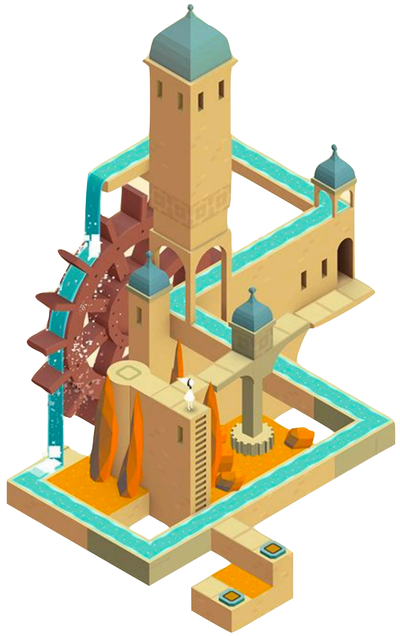


Meta-Learning for AutoML

ACDL 2021, part 2

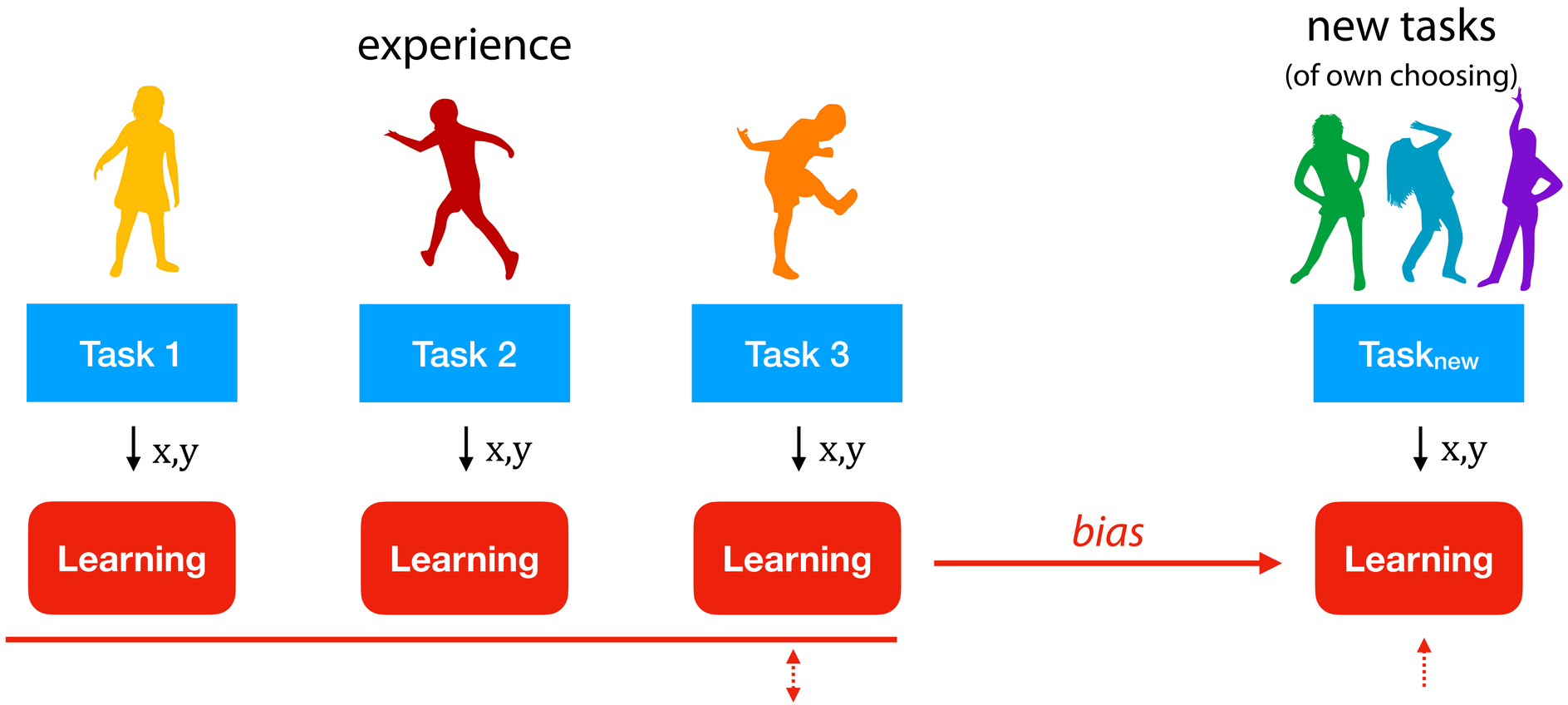
Joaquin Vanschoren (TU Eindhoven)

Cover art: MC Escher, Ustwo Games

Clune 2019

Recap: What can we learn to learn?

From hand-designed to learned learning algorithms … to AI-generating algorithms?



1. architectures / pipelines (hyperparameters, structures)

2. learning algorithms (priors, task embeddings,…)

3. learning environments (curricula, self-exploration)

Focus of this part

See part 2

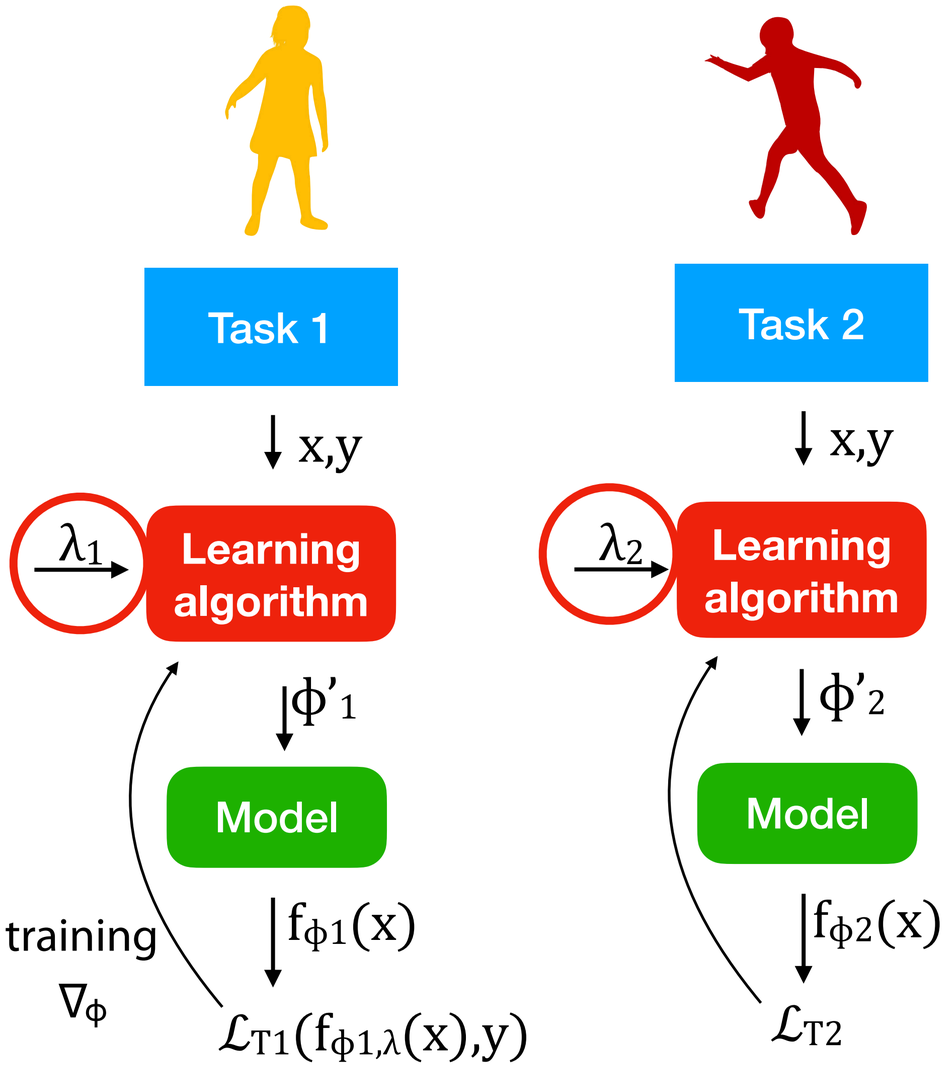
Machine Learning

Task: *distribution of samples* $q(x)$ *outputs* y, *loss* $\mathcal{L}(x,y)$

Learner: *model parameters* ϕ, *hyper-parameters* λ

↓

Neural architectures, pipelines, other hyperparameters, …



When the new task is quite similar, keep λ, ***(meta-)learn the model parameters*** ϕ

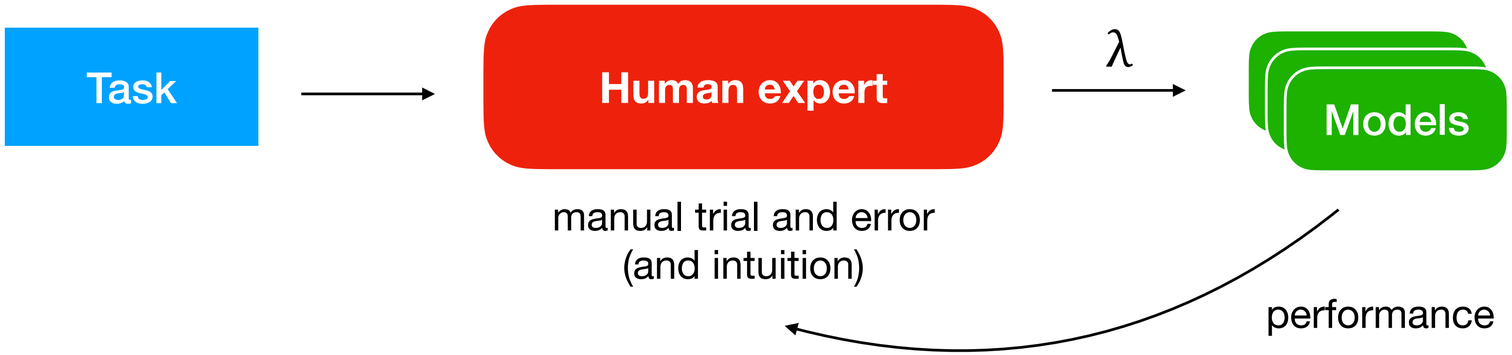
When the new task is quite different, ***(meta-)learn the hyper-parameters*** λ

Note: we can also learn λ *and* ϕ *at the same time (bilevel optimization)*

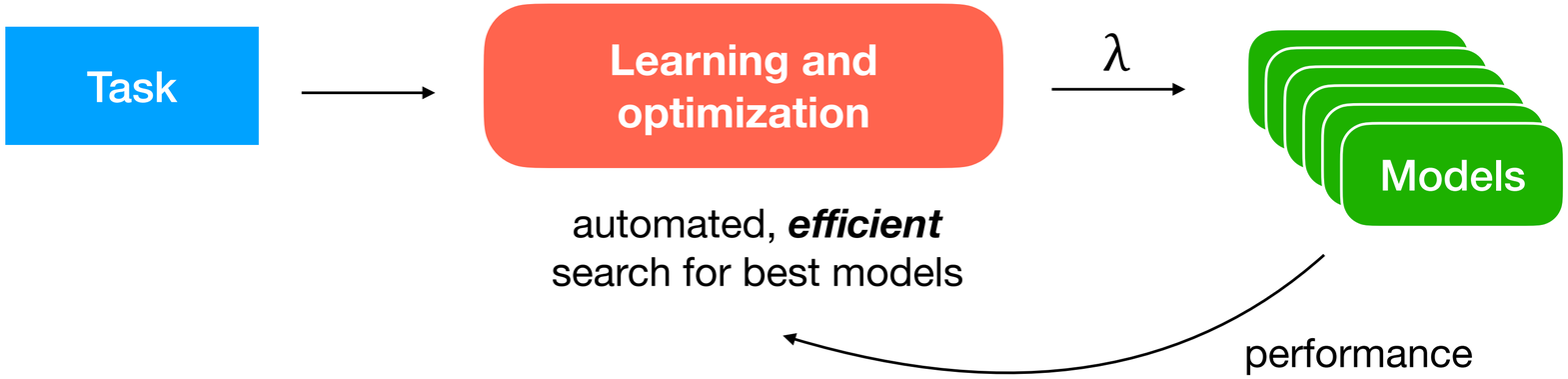
Hutter et al. 2019

Automatic Machine Learning (AutoML)

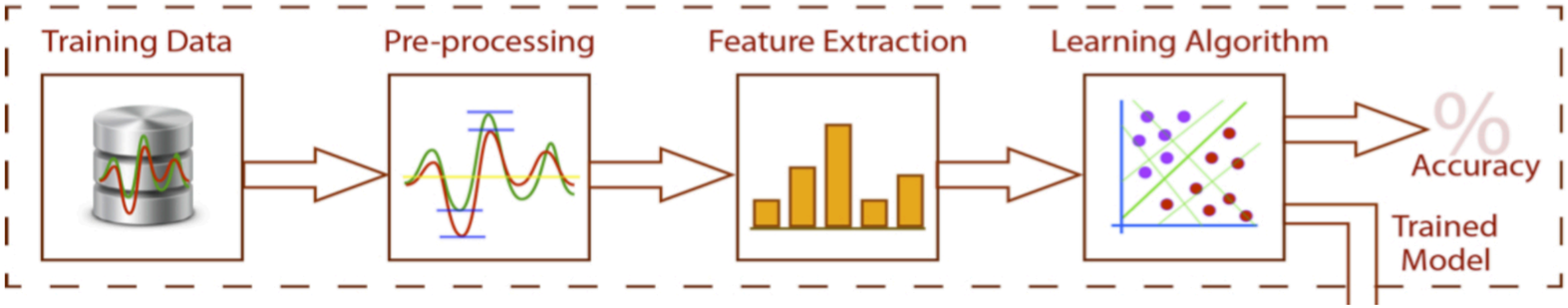
Manual machine learning



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



AutoML example: Pipeline synthesis



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

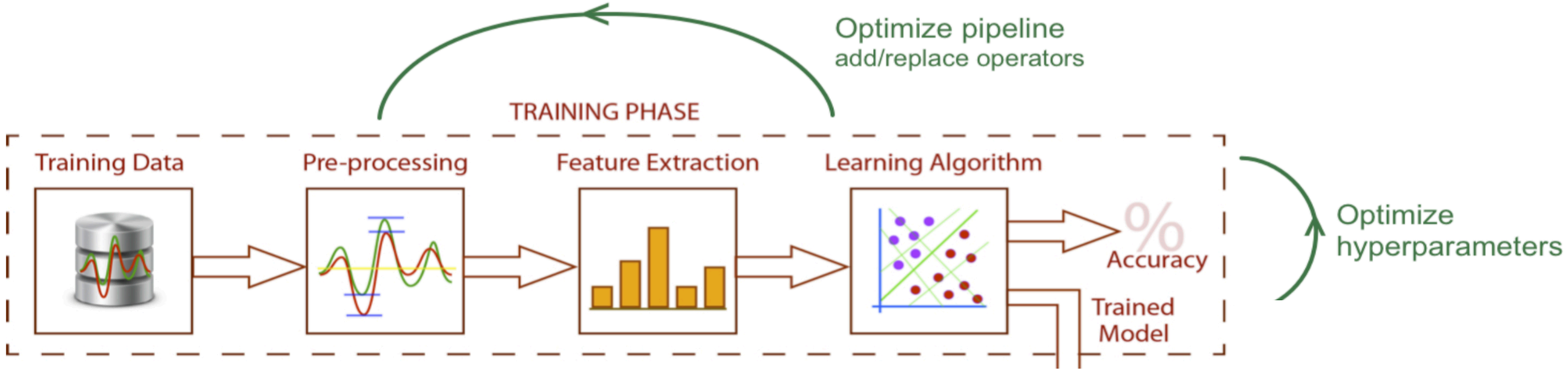
Figure source: Nick Gillian

AutoML example: Pipeline synthesis

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

Figure source: Nick Gillian

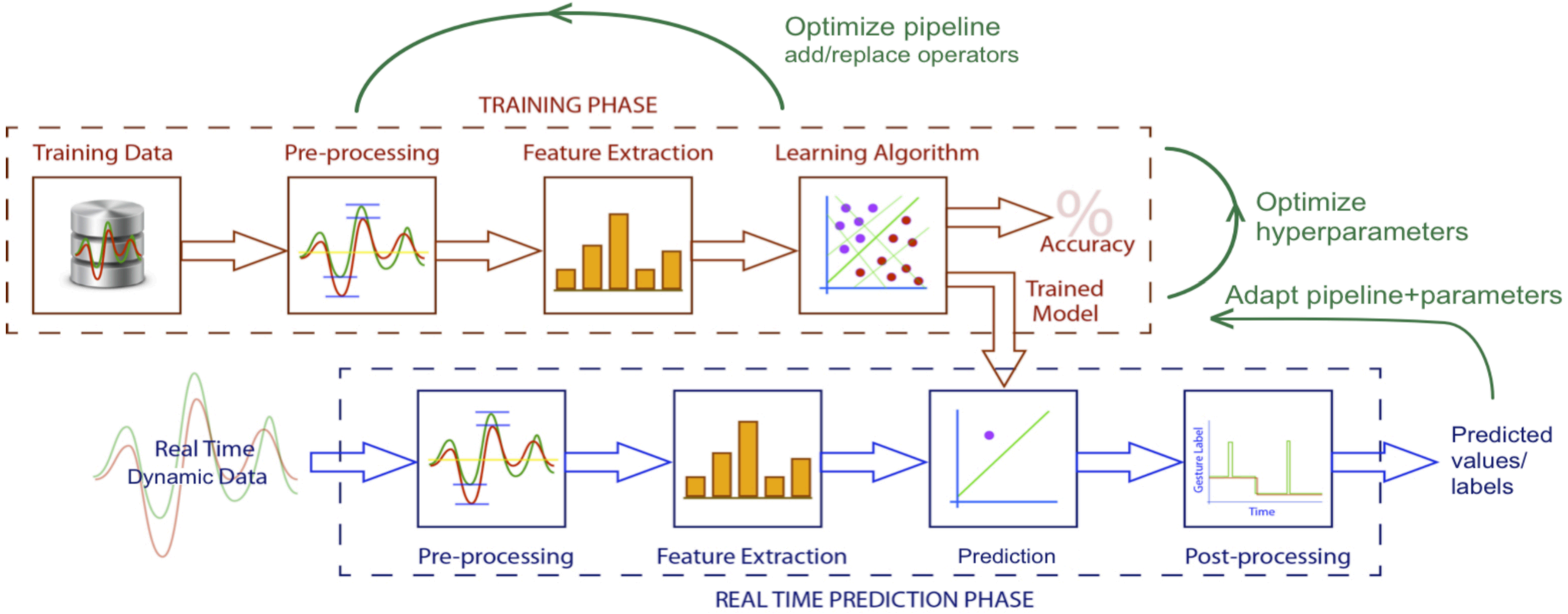
AutoML example: Pipeline synthesis



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

Figure source: Nick Gillian

AutoML example: Pipeline synthesis



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

Figure source: Nick Gillian

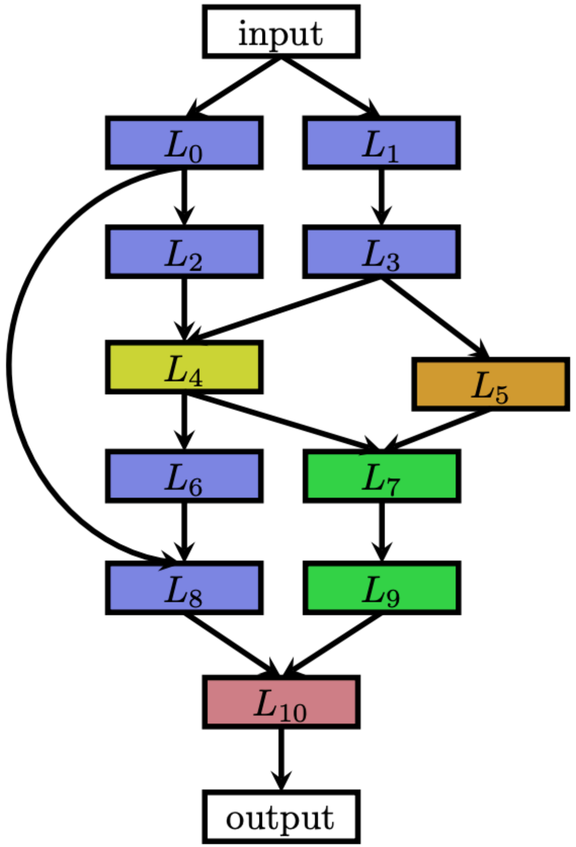
AutoML example: Neural Architecture Search

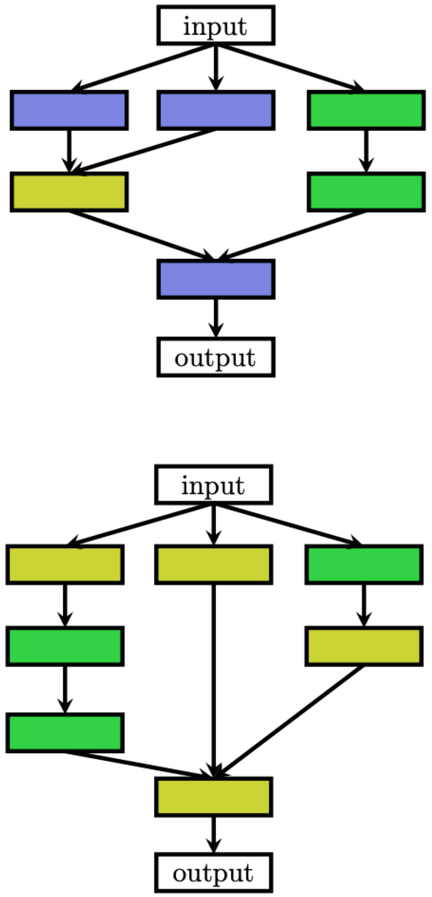
Architecture:

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

Optimization:

- Gradient descent hyperparameters
- Regularization
- …





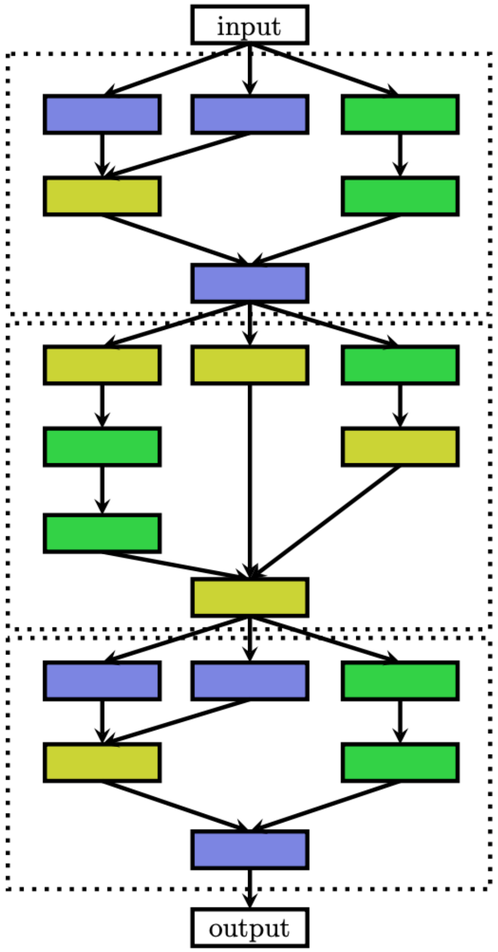


Figure source: Elsken et al., 2018

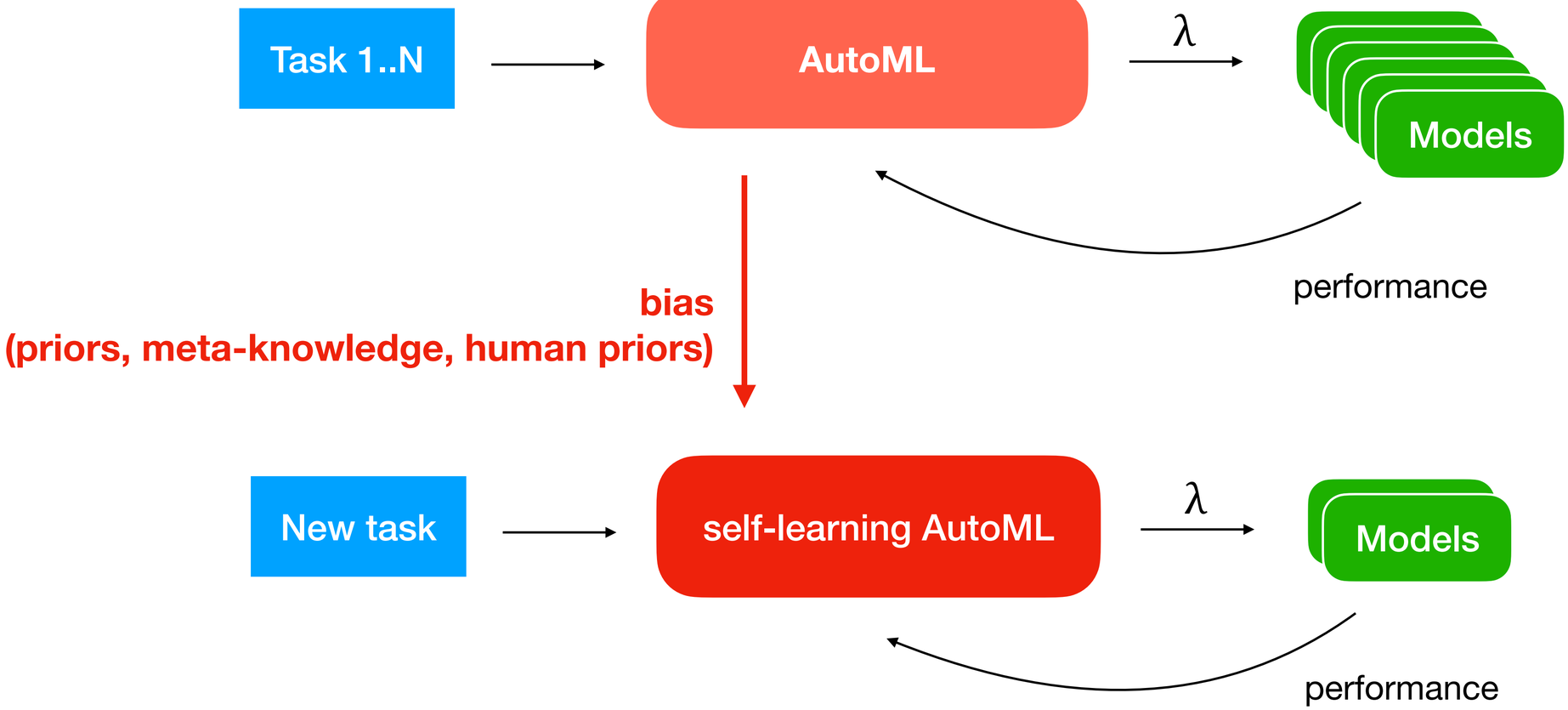
Hutter et al. 2019

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!

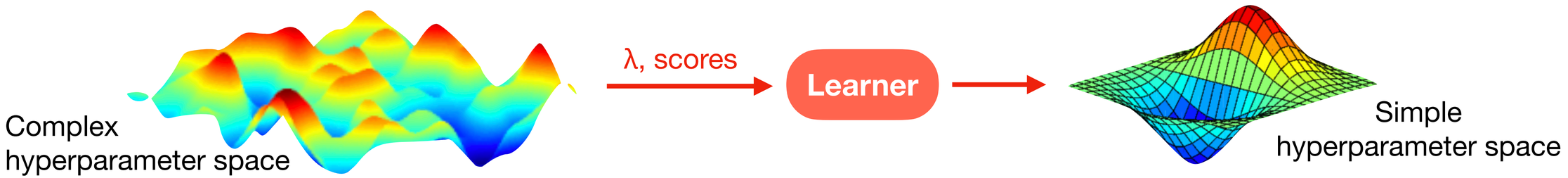


Vanschoren 2018

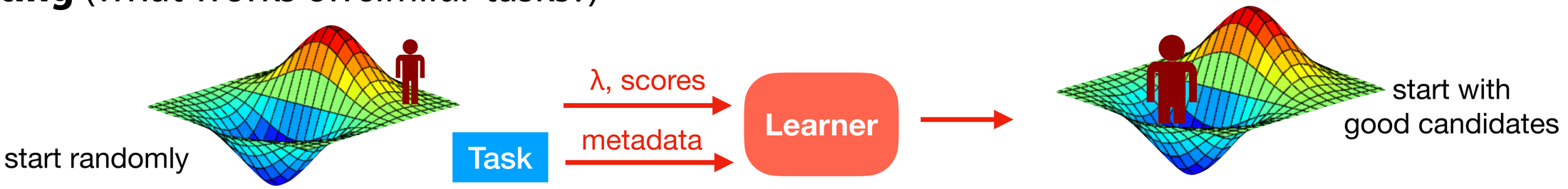
Meta-learning for AutoML: how?

hyperparameters = architecture + hyperparameters

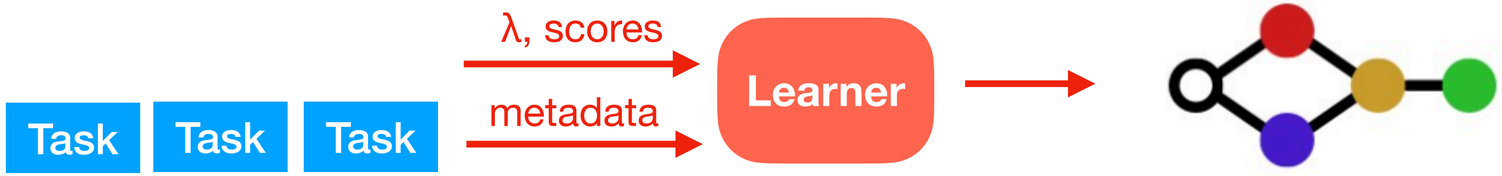
Learning hyperparameter priors



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



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



Observation:
current AutoML strongly depends on learned priors

Manual architecture priors

- Most successful pipelines have a similar structure

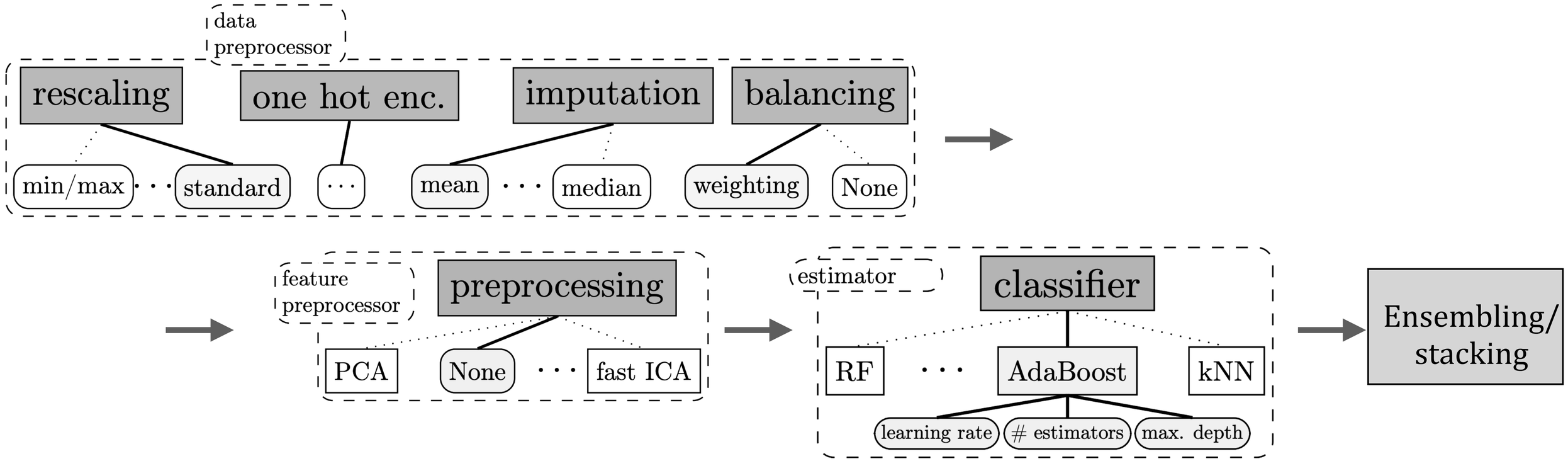
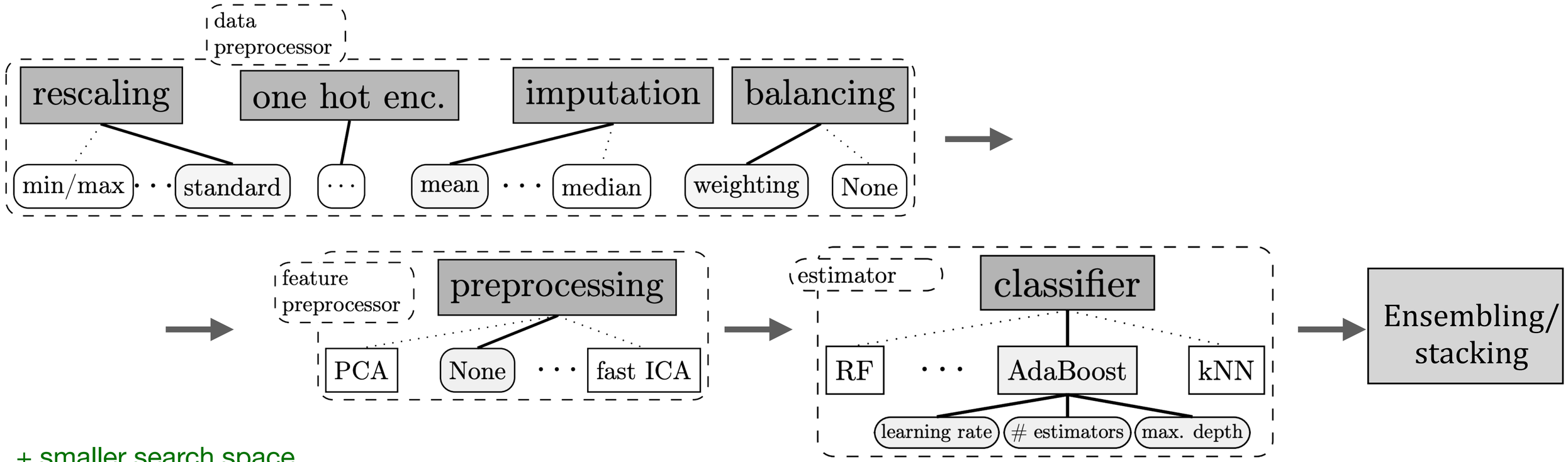


Figure source: *Feurer et al. 2015*

Manual architecture priors

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



+ smaller search space

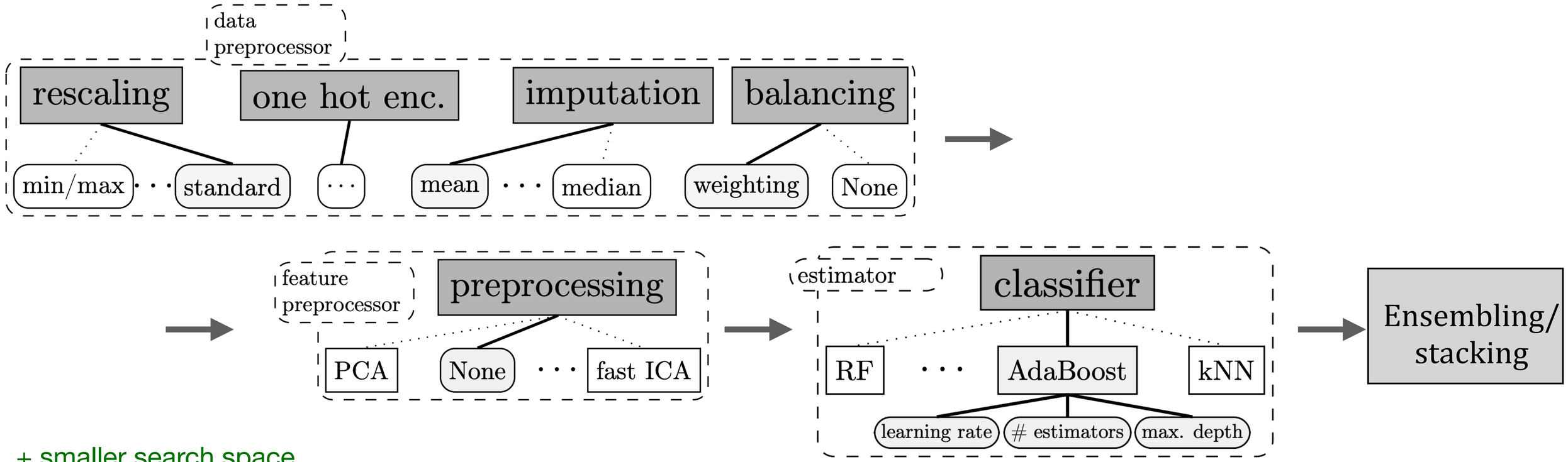
- you can't learn entirely new architectures

Figure source: *Feurer et al. 2015*

Manual architecture priors

autosklearn *Feurer et al. 2015*
autoWEKA *Thornton et al. 2013*
hyperopt-sklearn *Komer et al. 2014*
AutoGluon-Tabular *Erickson et al. 2020*

- 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

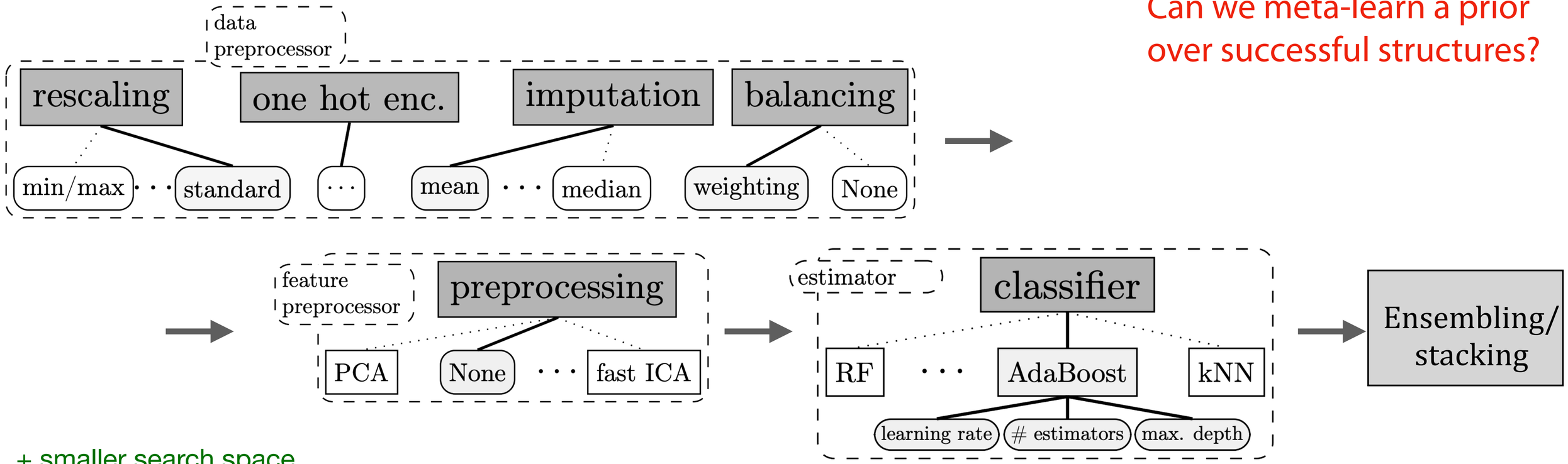
Figure source: *Feurer et al. 2015*

Manual architecture priors

autosklearn *Feurer et al. 2015*
autoWEKA *Thornton et al. 2013*
hyperopt-sklearn *Komer et al. 2014*
AutoGluon-Tabular *Erickson et al. 2020*

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

Can we meta-learn a prior over successful structures?



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

Figure source: *Feurer et al. 2015*

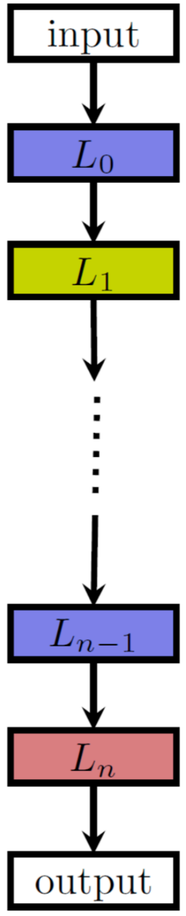
Elsken et al. 2019

Manual architecture priors

Parameterized Sequential

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

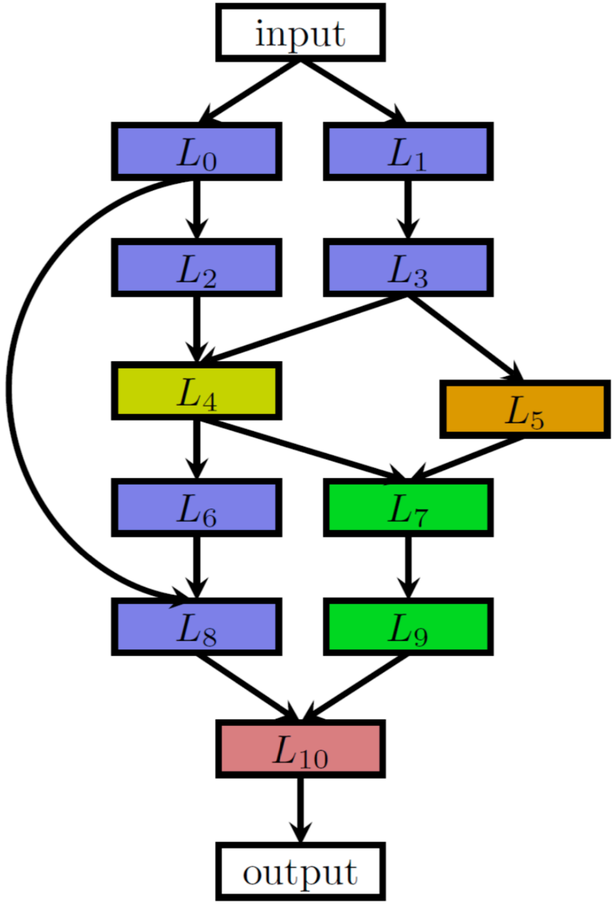
+ easier to search
- sometimes too simple



Parameterized Graph

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

+ more flexible
- much harder to search



Manual architecture priors

Successful deep networks often have repeated motifs (cells)

e.g. Inception v4:

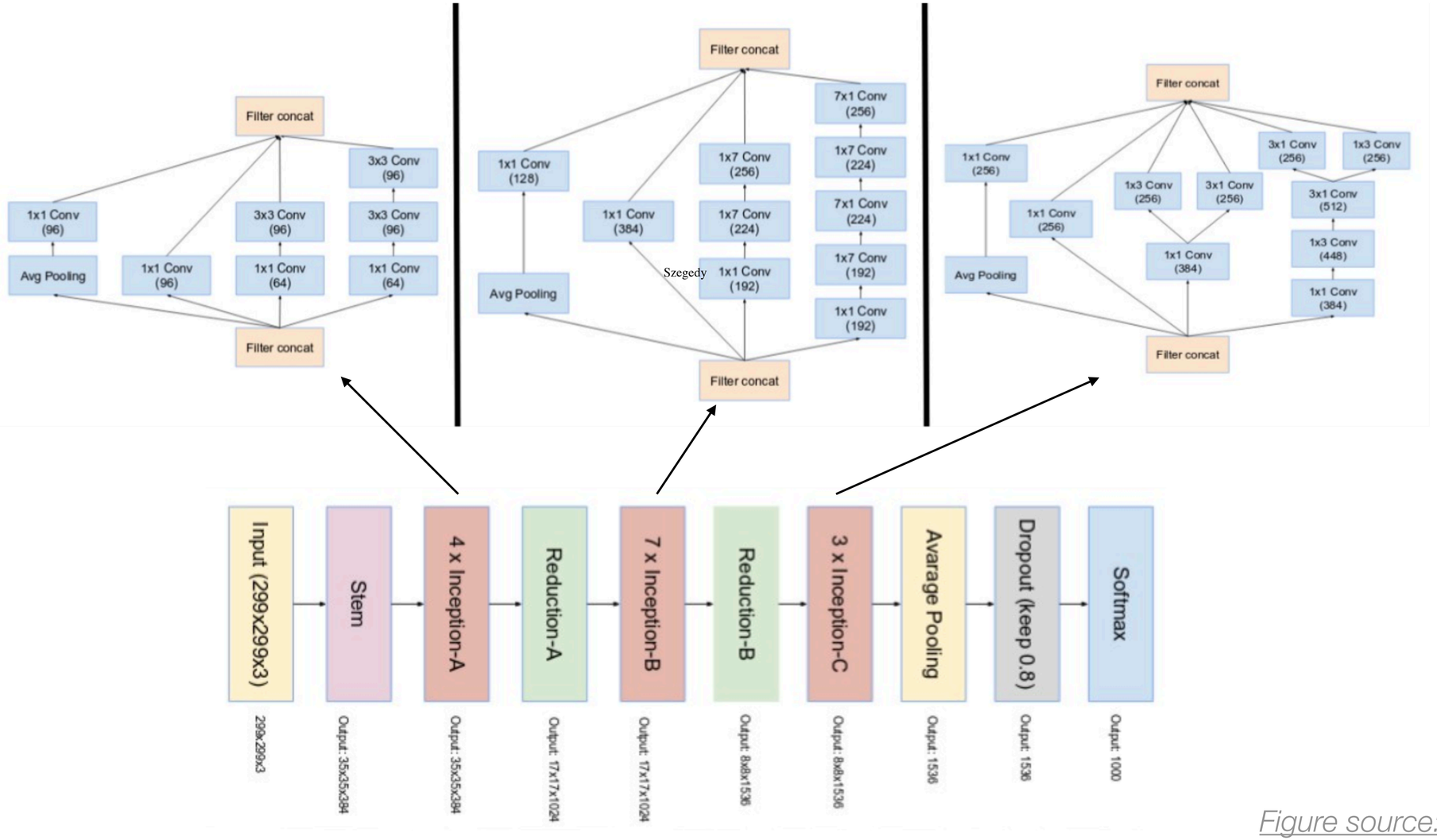


Figure source: Szegedy et al 2016

Google NASNet Zoph et al 2018

Cell search space prior

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 to a larger dataset
- strong domain priors, doesn't generalize well

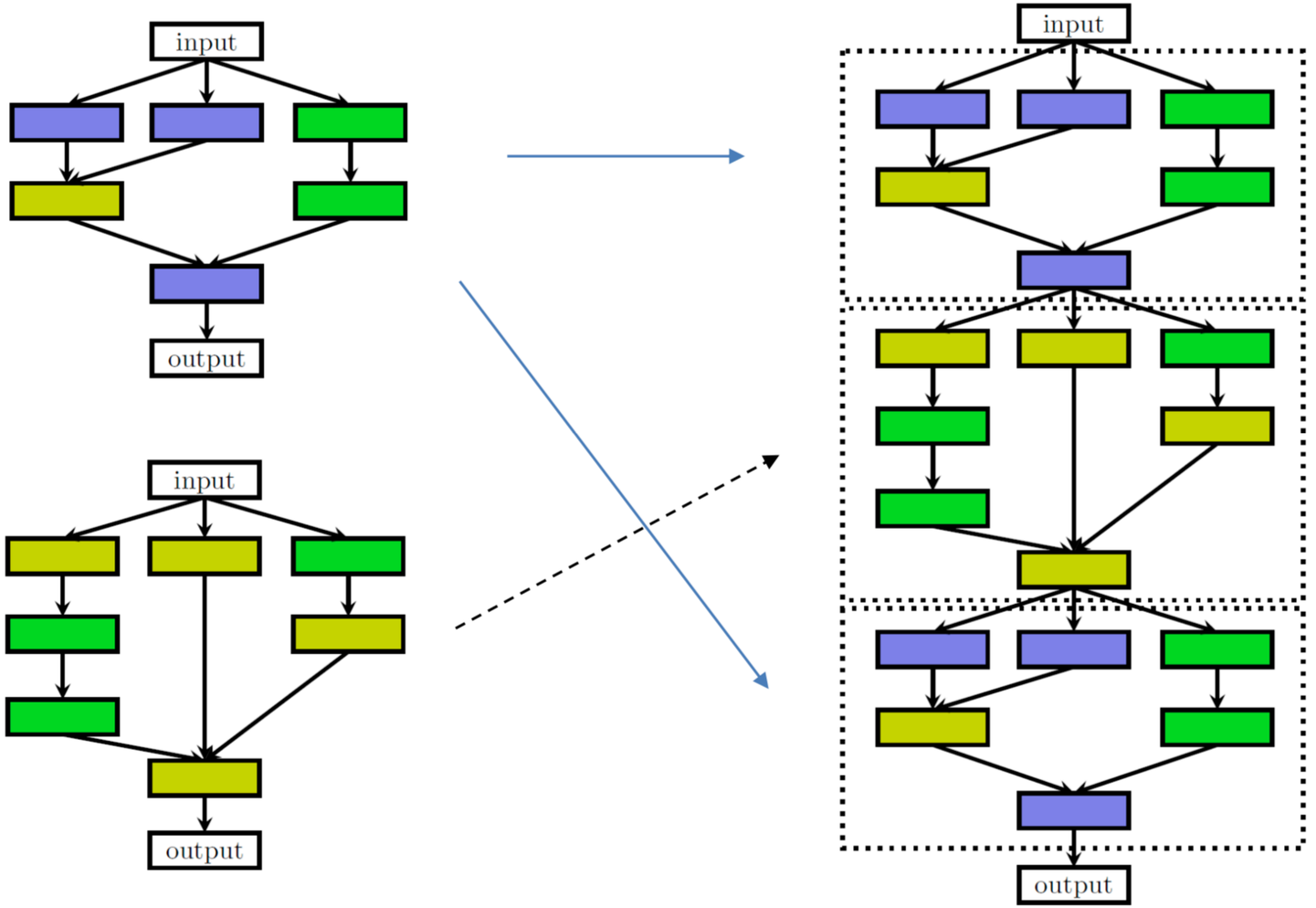


Figure source: Elsken et al., 2019

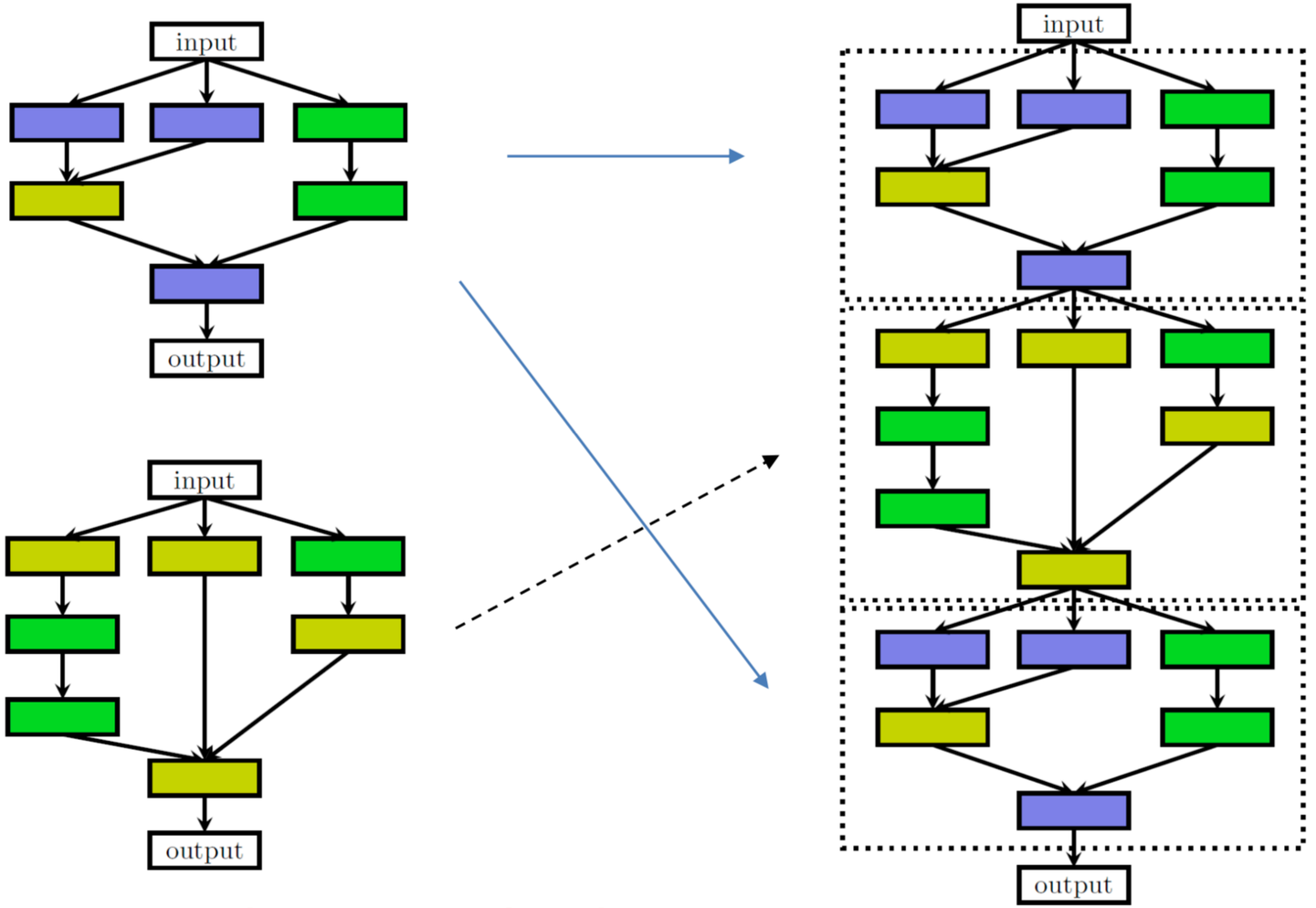
Google NASNet Zoph et al 2018

Cell search space prior

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 to a larger dataset
- strong domain priors, doesn't generalize well

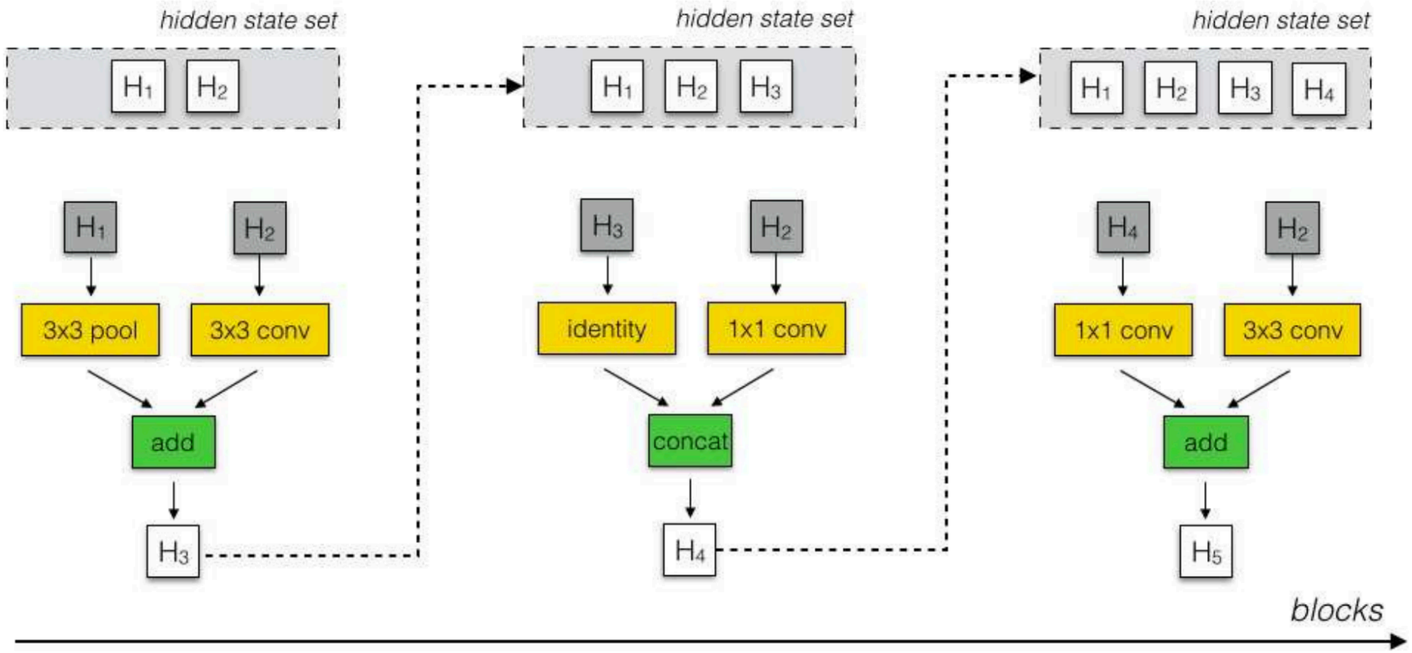
Can we meta-learn hierarchies / components that generalize better?

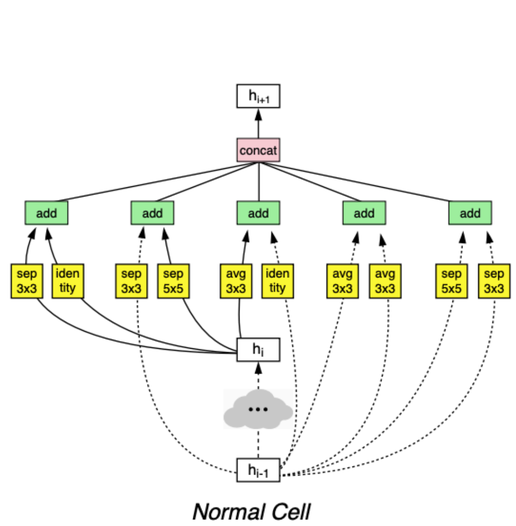
Figure source: Elsken et al., 2019

NASNet, Zoph et al 2018

Cell search space prior

- Cell construction with RL-based search (SOTA ImageNet, 450 GPUs, 3-4 days):
  - 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





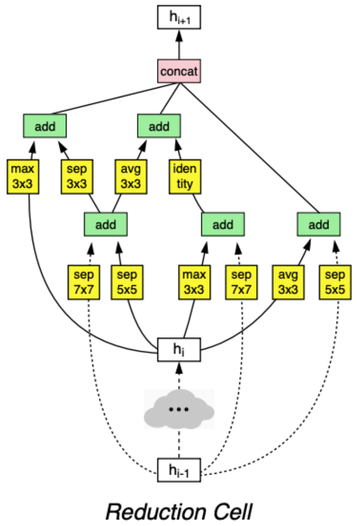
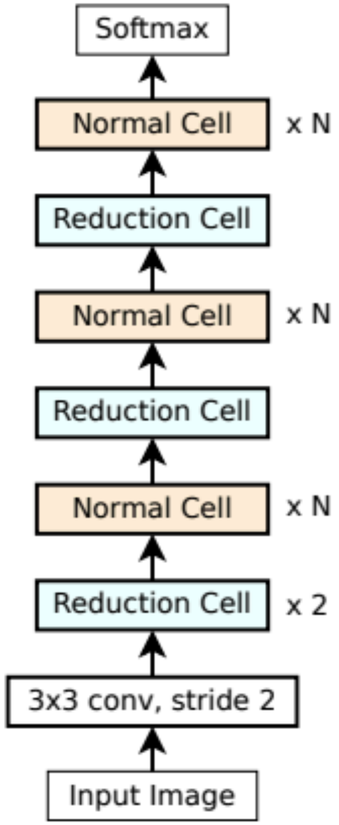


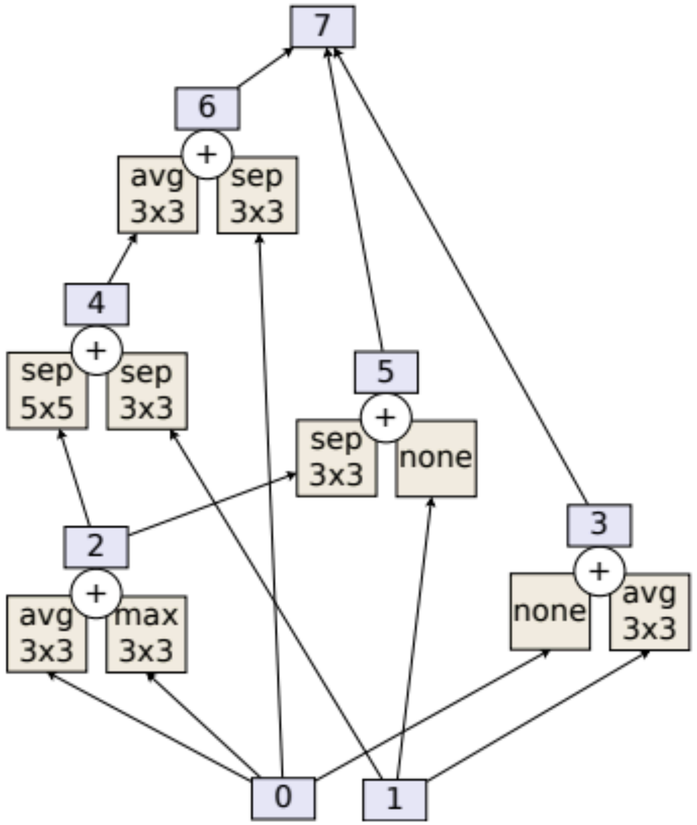
Figure source: Zoph et al., 2018

Cell search space prior

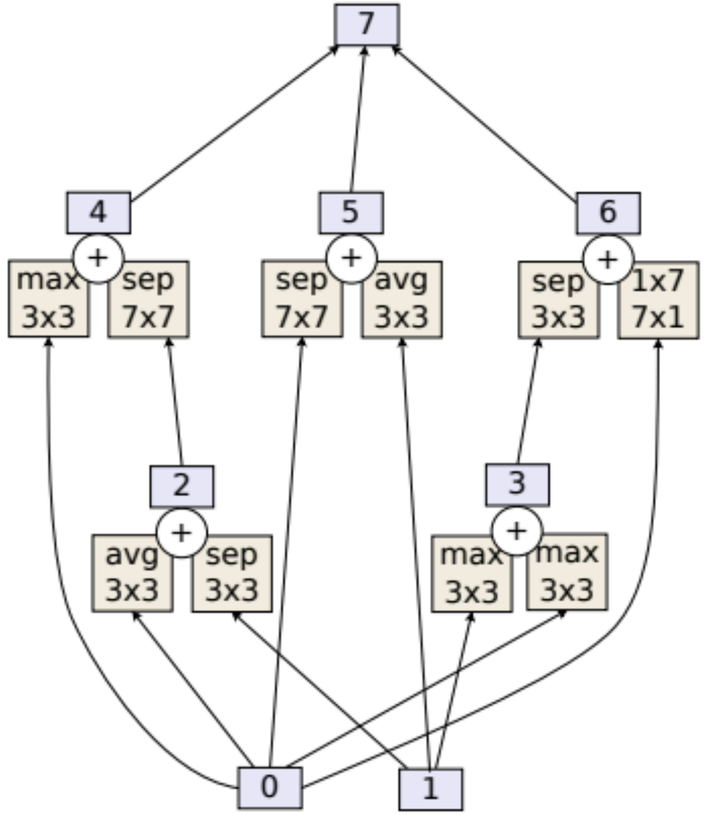
AmoebaNet, Real et al 2019

- Cell construction with neuro-evolution (SOTA ImageNet)





normal cell



reduction cell

Figure source: Real et al., 2019

Cell search space prior

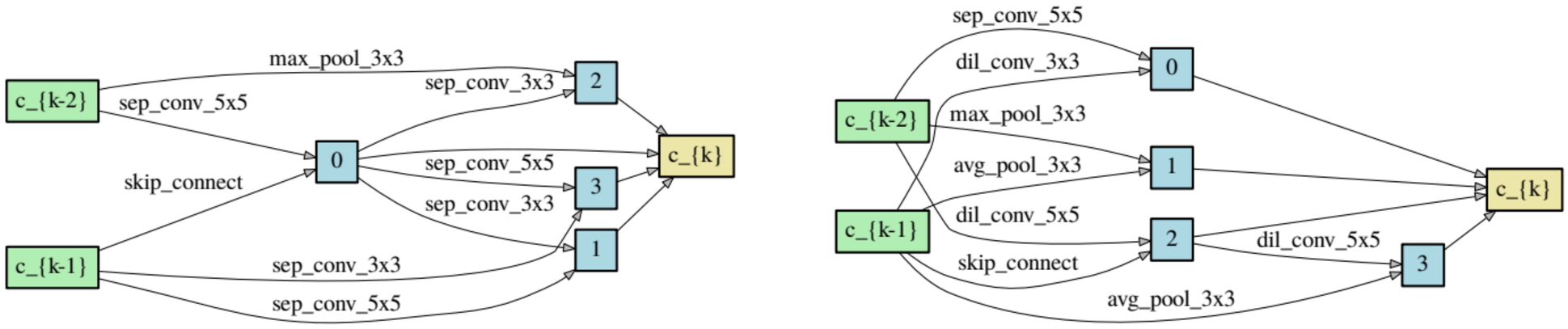
Li & Talwalkar 2019

Yu et al. 2019

Real et al. 2019

- Cell construction with multi-fidelity random search!

  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.

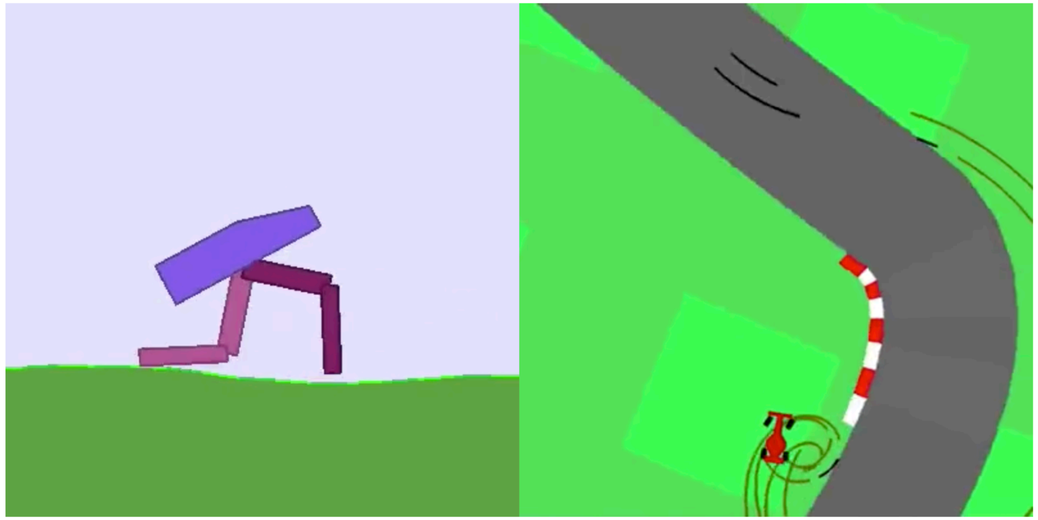
Figure source: Li & Talwalkar., 2019

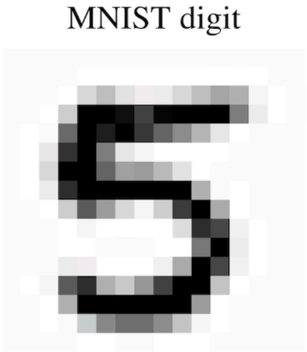
Gaier & Ha, 2019

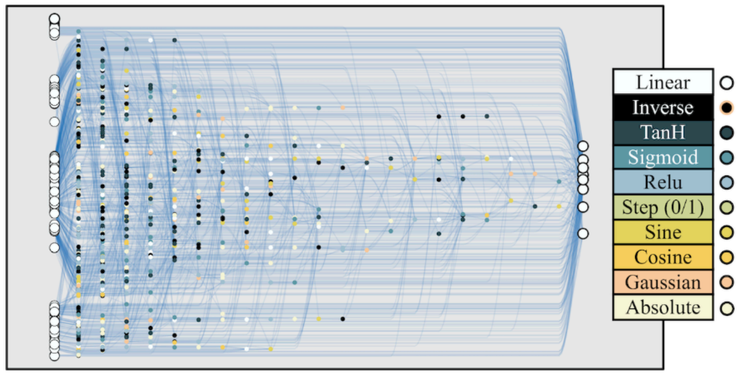
Manual priors: Weight sharing

Weight-agnostic neural networks

- ALL weights are shared
- Only evolve the architecture?
  - Minimal description length
  - Baldwin effect?







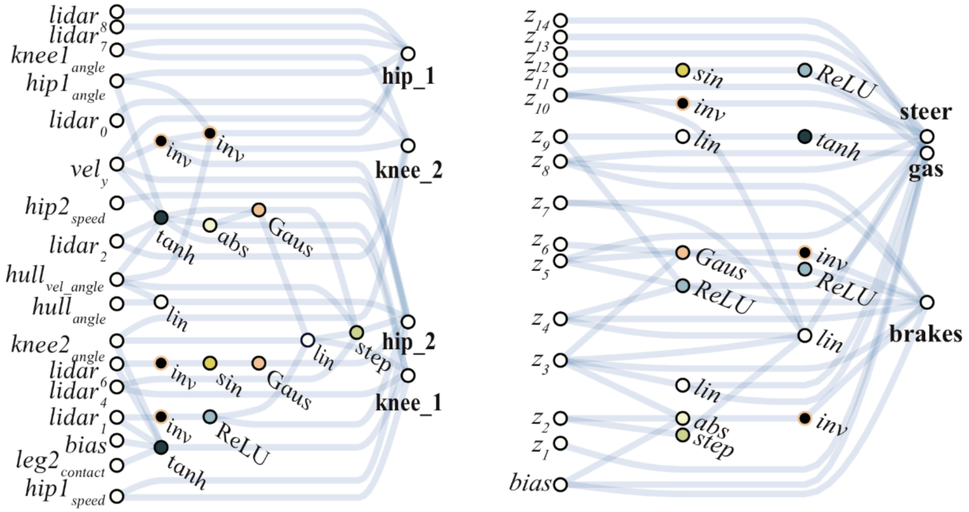
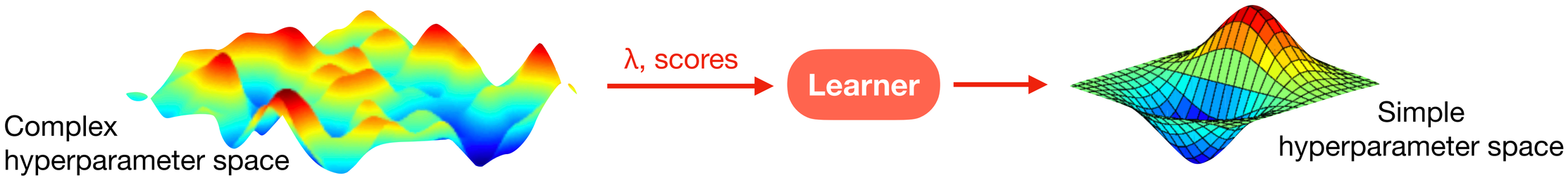


Figure source: Gaier & HA., 2019

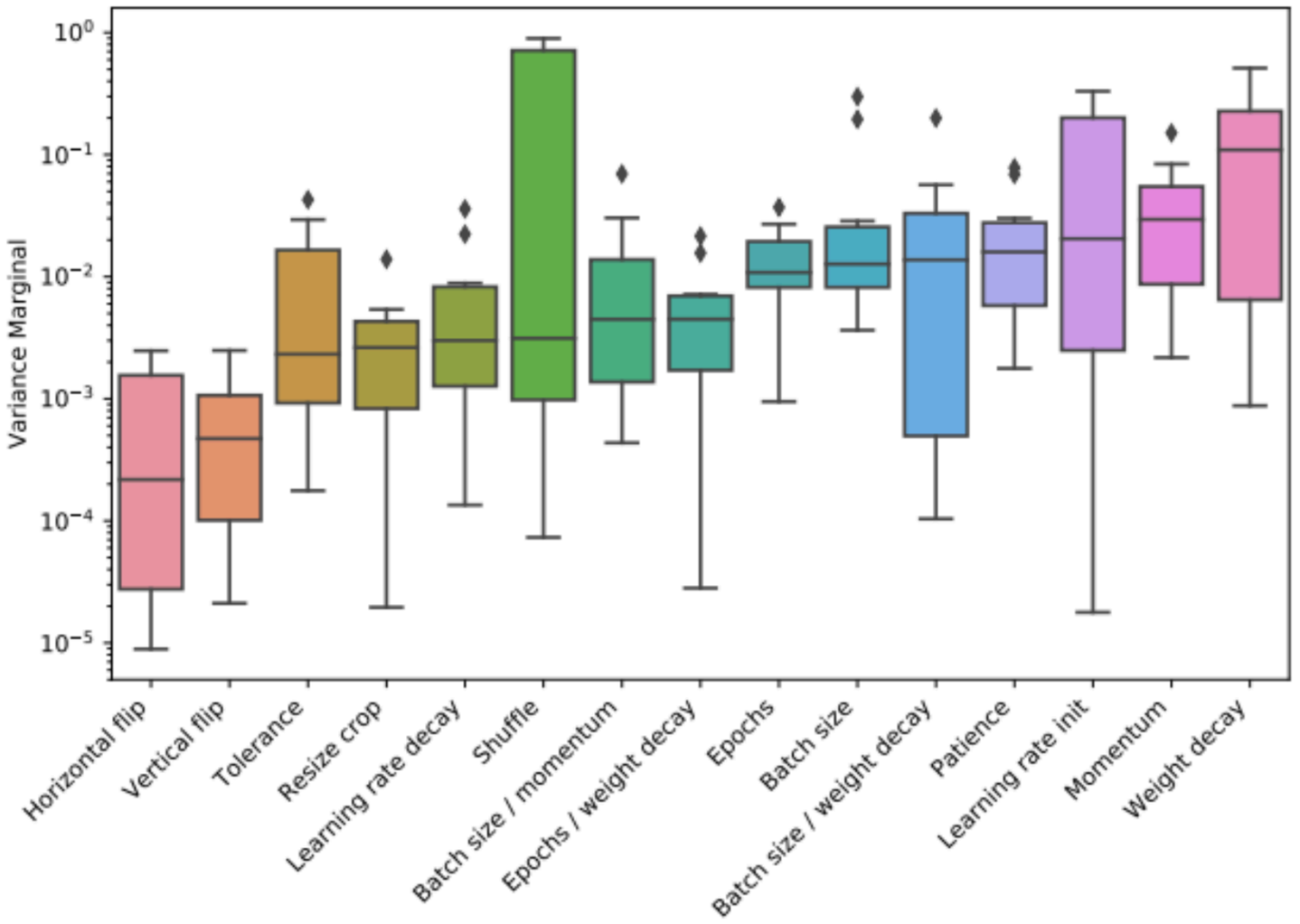
Learning hyperparameter priors



[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

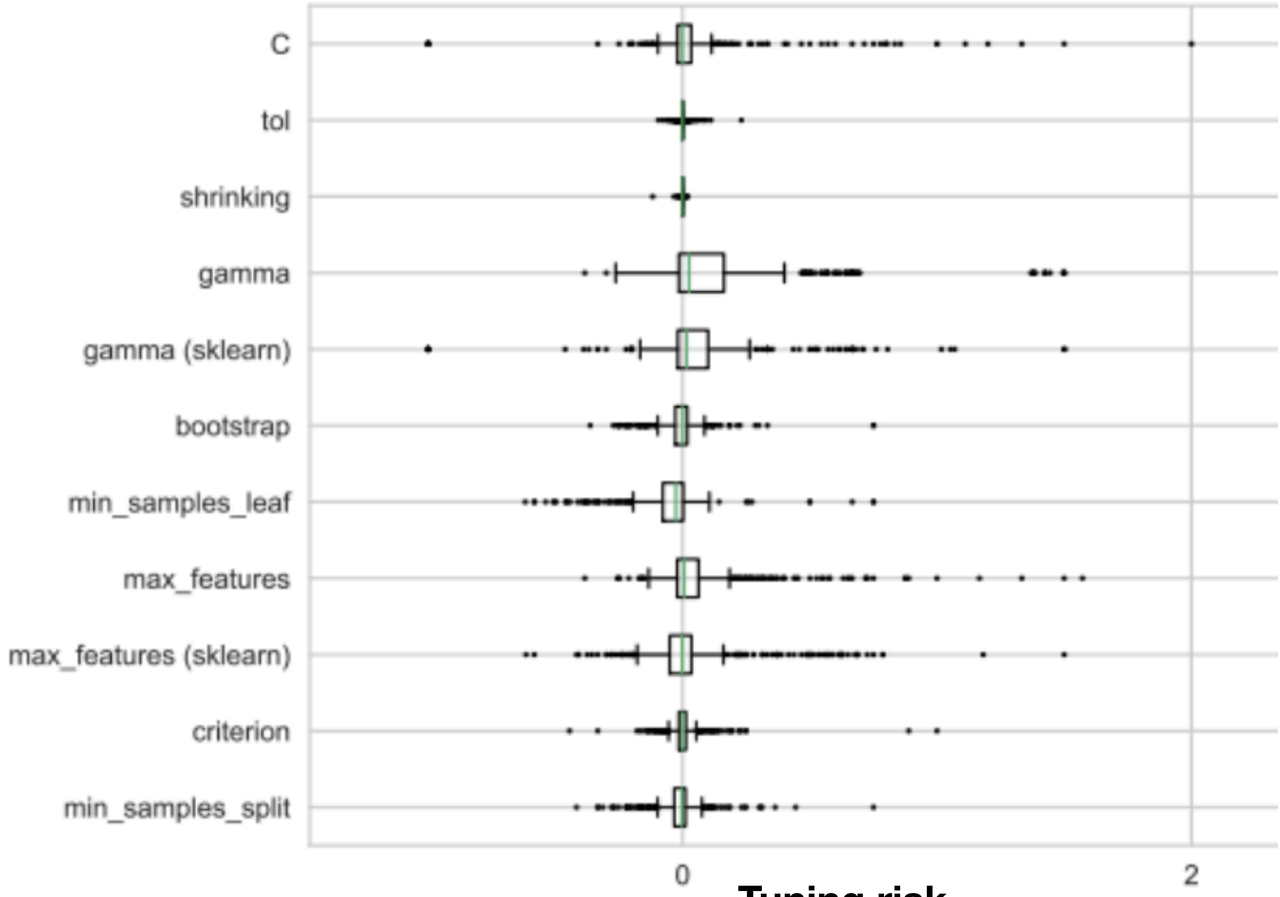
[2] Weerts et al. 2018

[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

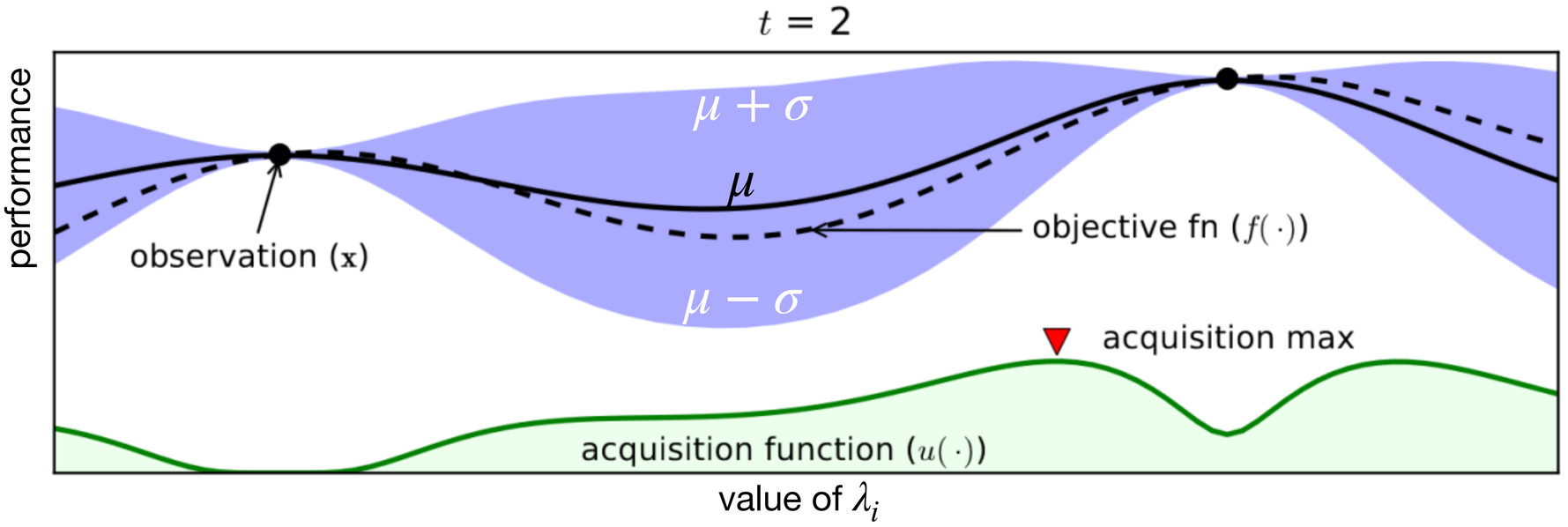


Tuning risk

Mockus, 1974

Bayesian Optimization (interlude)

- Start with a few (random) hyperparameter configurations
- Fit a ***surrogate model*** to predict other configurations
- Probabilistic regression: 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

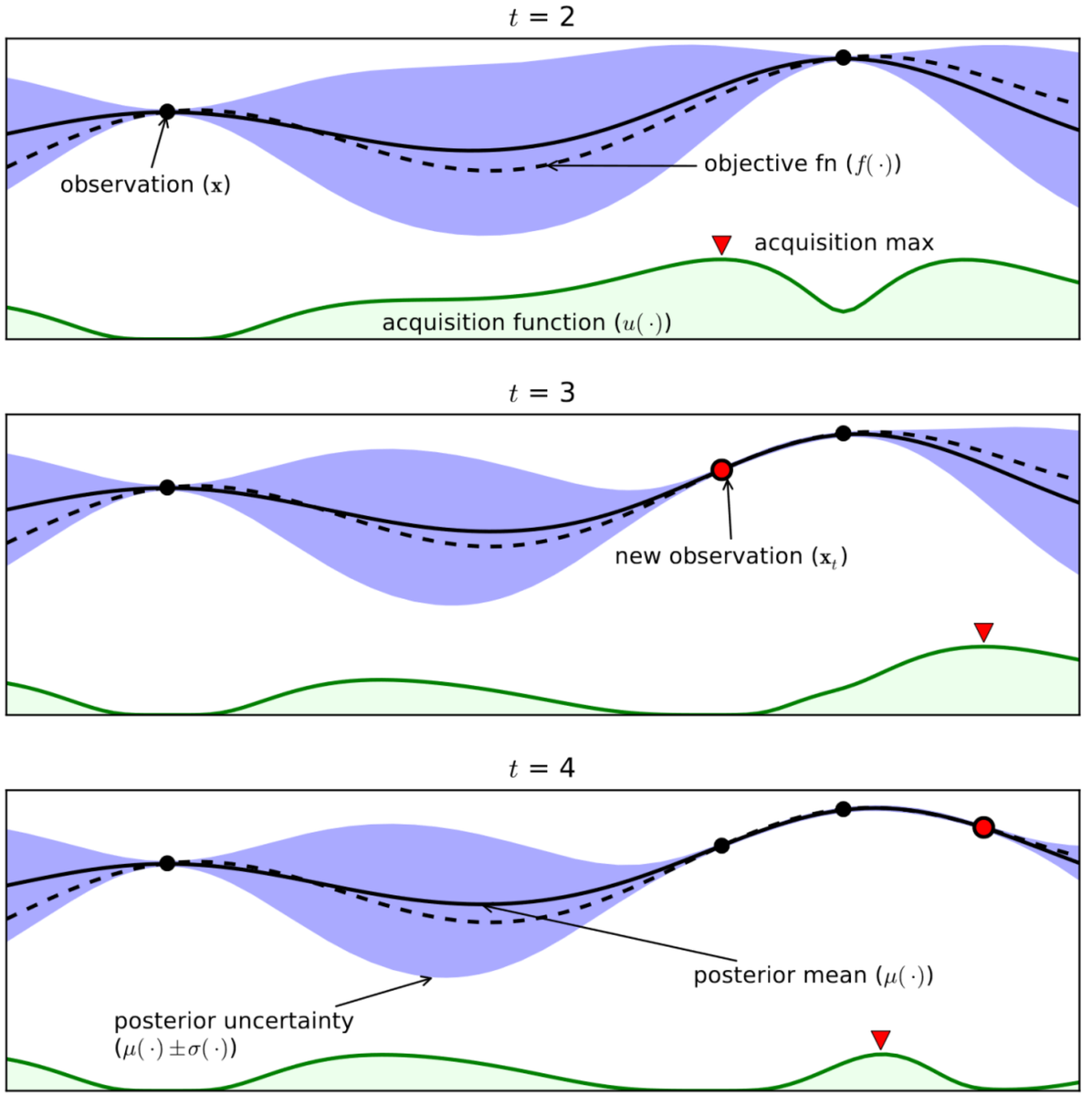


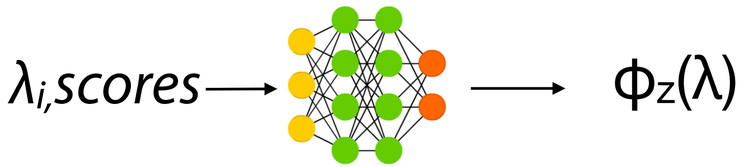
Figure source: Shahriari 2016

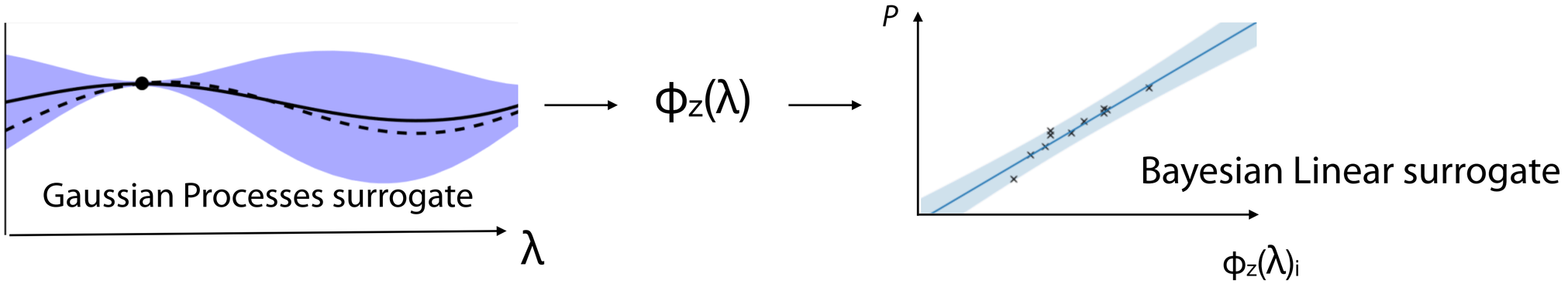
Perrone et al. 2018

Learn basis expansions for hyperparameters

- Hyperparameters can interact in very non-linear ways
- Use a neural net to learn a suitable basis expansion $\phi_z(\lambda)$ for all tasks
- You can use Bayesian linear models, transfers info on configuration space

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



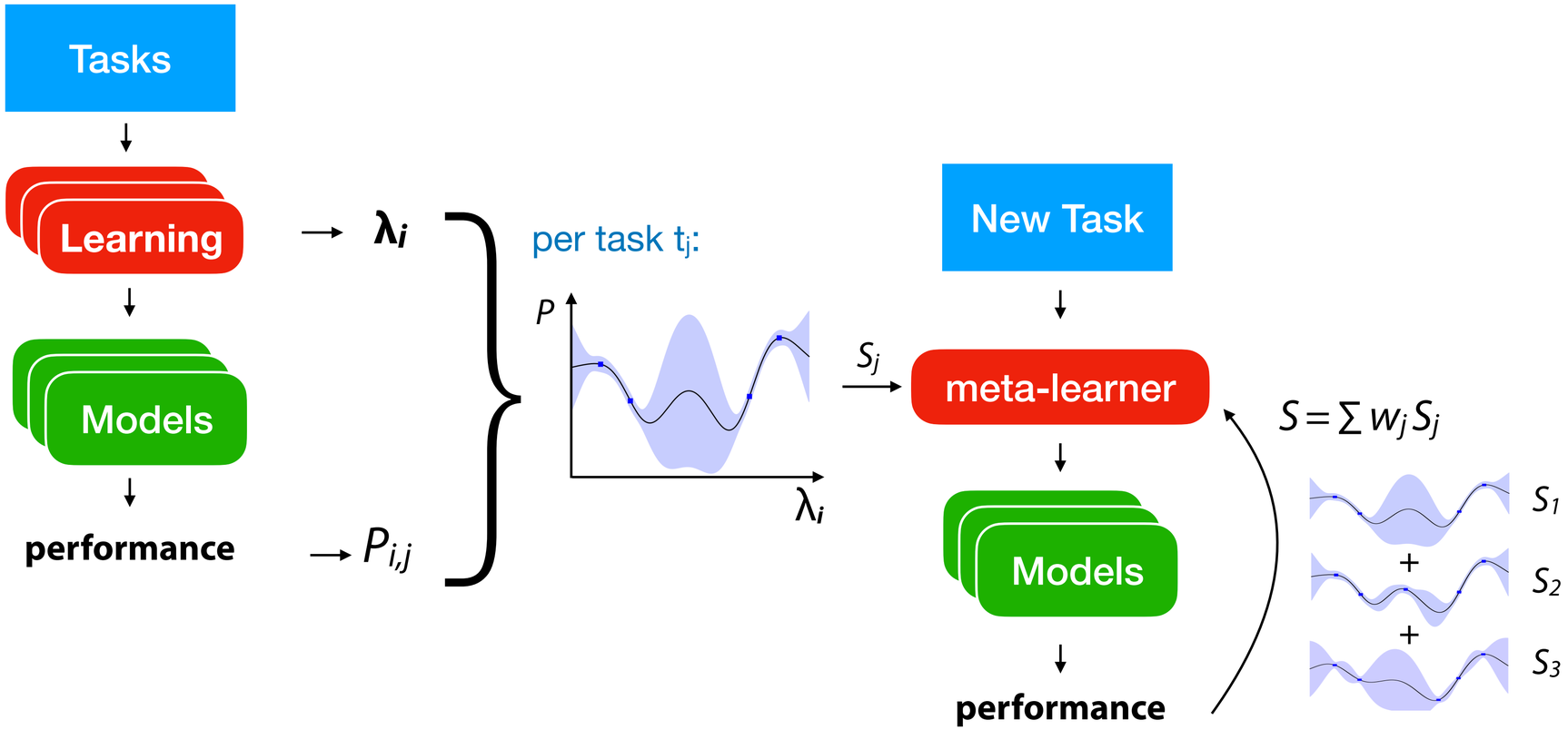


Surrogate model transfer

[1] Wistuba et al. 2018

[2] Feurer et al. 2018

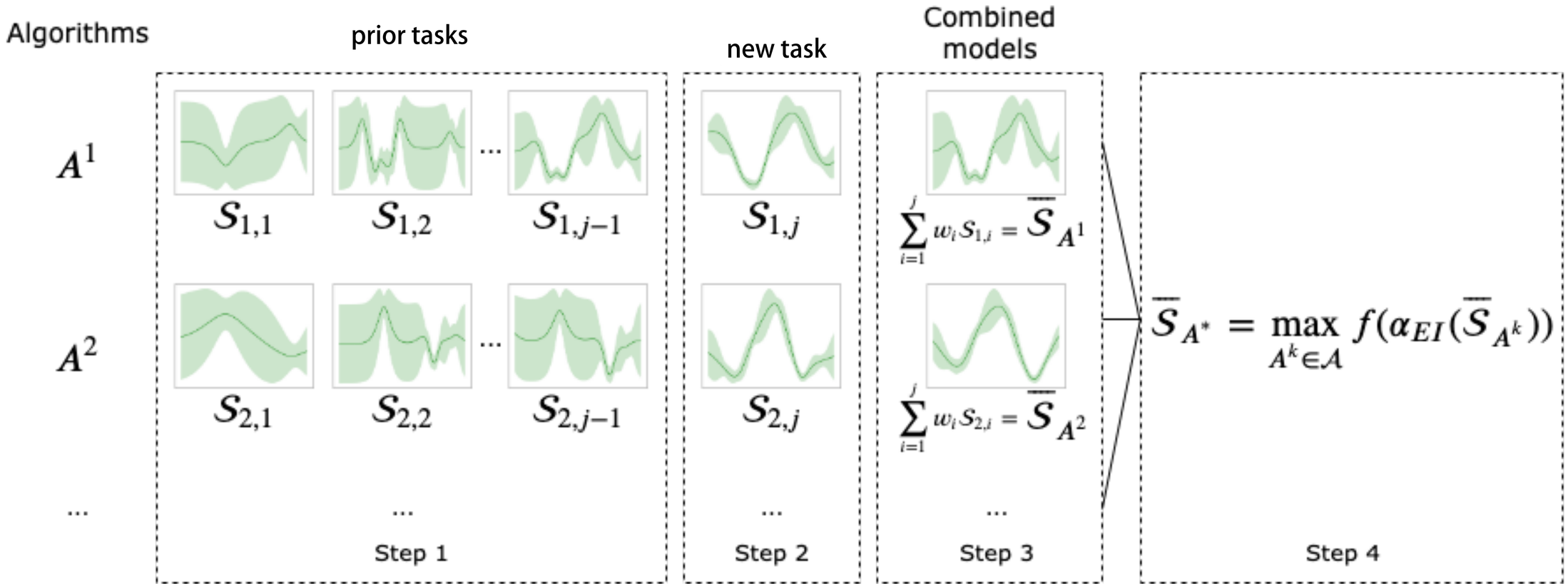
- If task j is *similar* to the new task, its surrogate model S_j will likely transfer well
- Sum up all S_j predictions, weighted by task similarity (as in active testing)[1]
- Build combined Gaussian process, *weighted by current performance* on new task[2]



Manolache & Vanschoren 2019

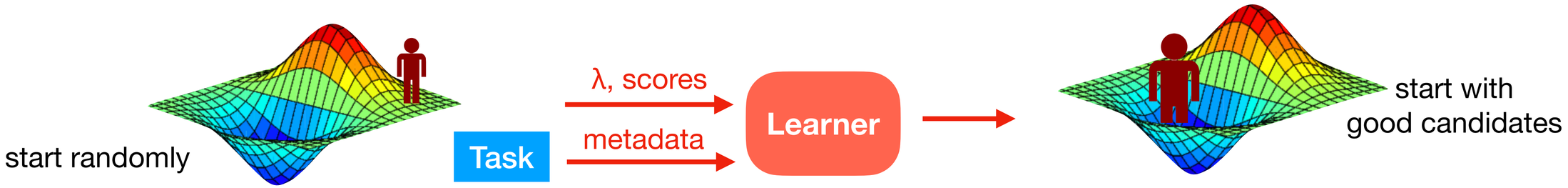
Surrogate model transfer

- Store surrogate model S_{ij} for every pair of task i and algorithm j
- Simpler surrogates, better transfer
- Learn weighted ensemble -> significant speed up in optimization



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 AutoML Challenges

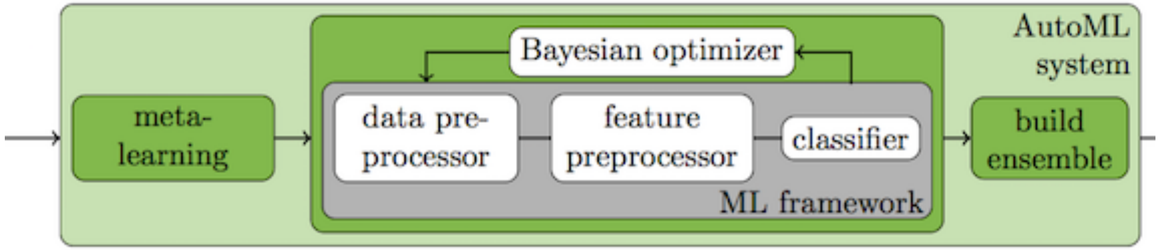
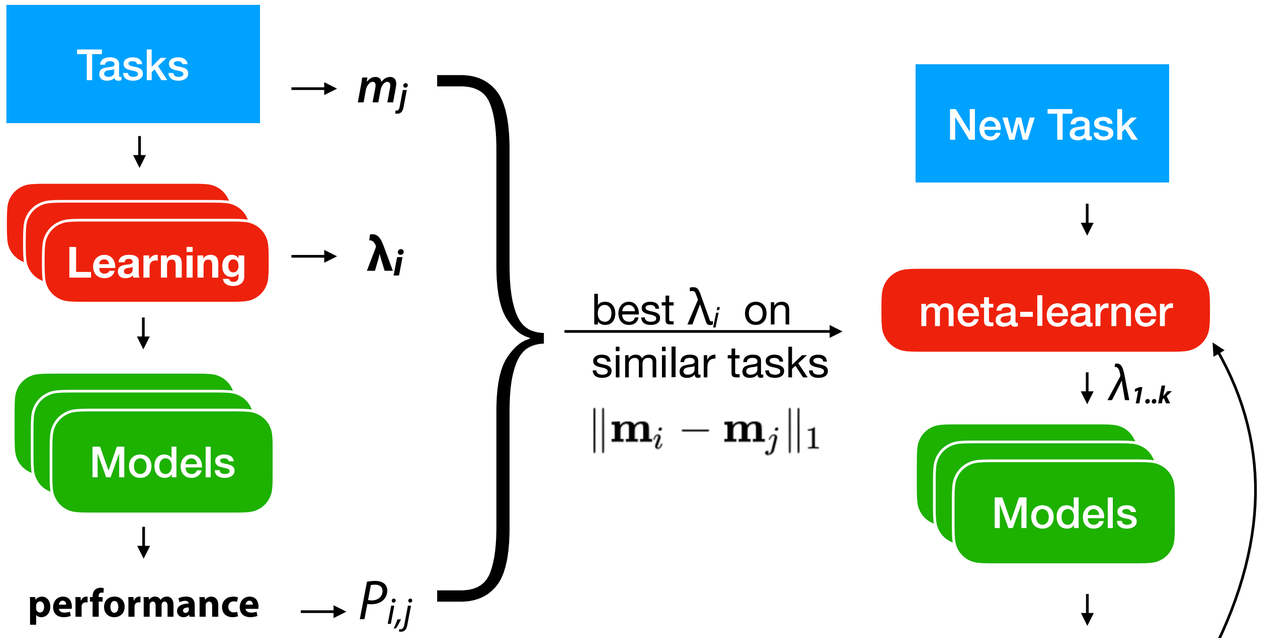
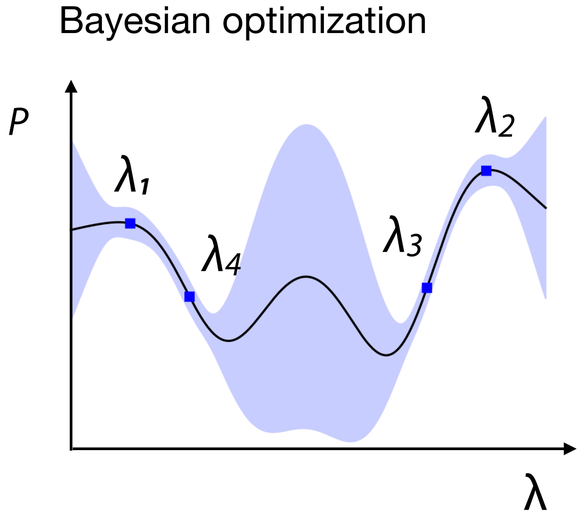


Figure source: Feurer et al., 2015



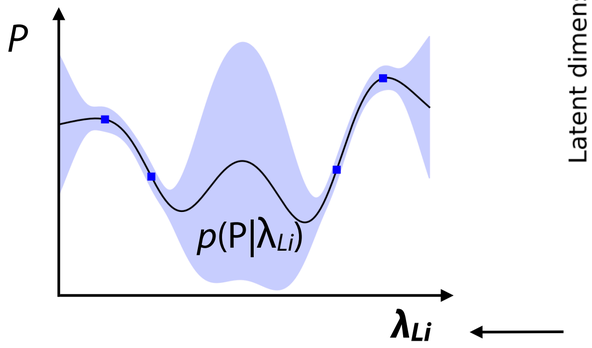


Fusi et al. 2017

Probabilistic Matrix Factorization

- 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



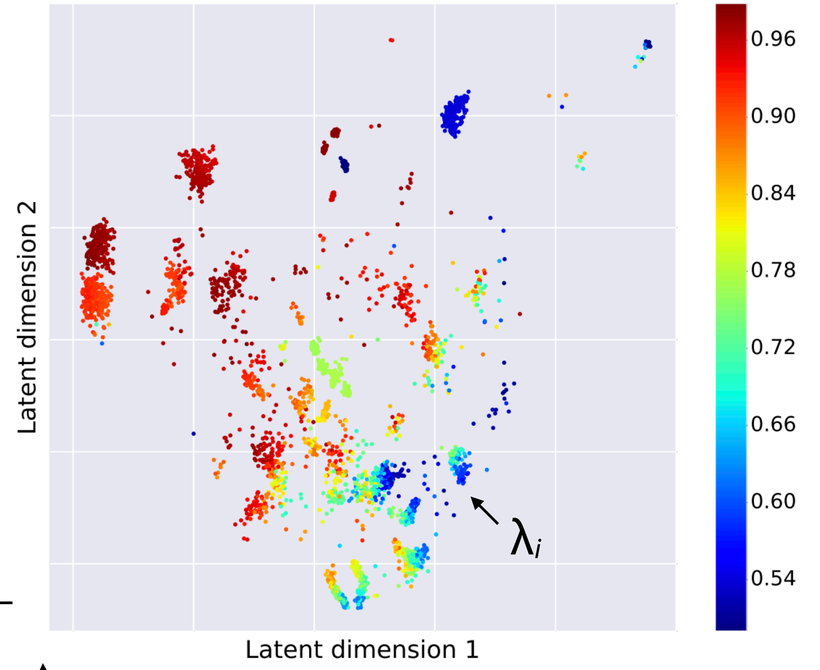
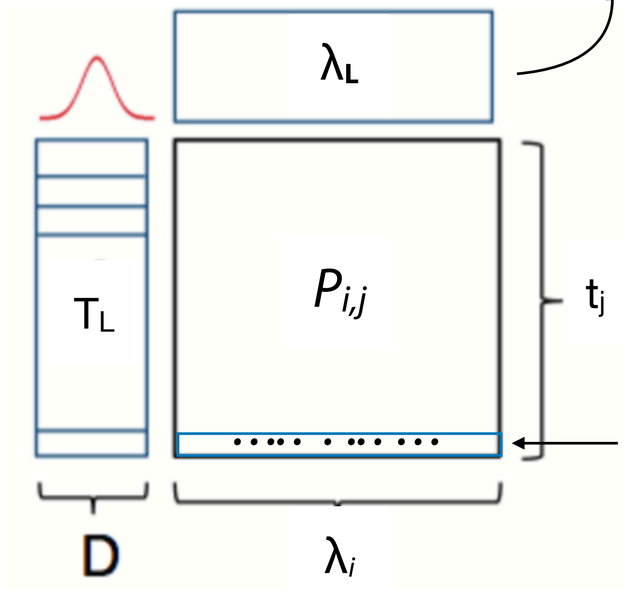


Figure source: Fusi et al., 2017



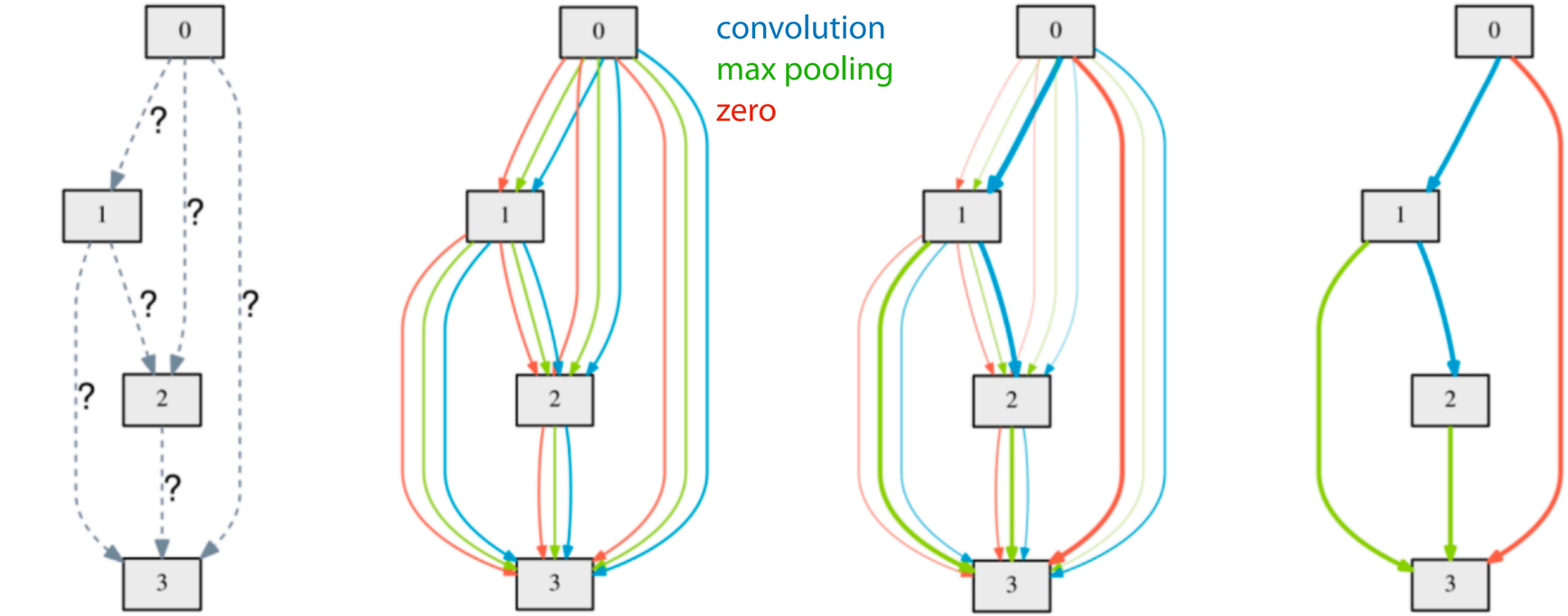
latent representation

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

Liu et al. 2018

DARTS: Differentiable NAS

- Fixed (one-shot) structure, learn which operators to use
- Give all operators a weight α_i
- Optimize α_i and model weights ω_j using bilevel optimization
  - approximate $\omega_j^*(\alpha_i)$ by adapting ω_j after every training step

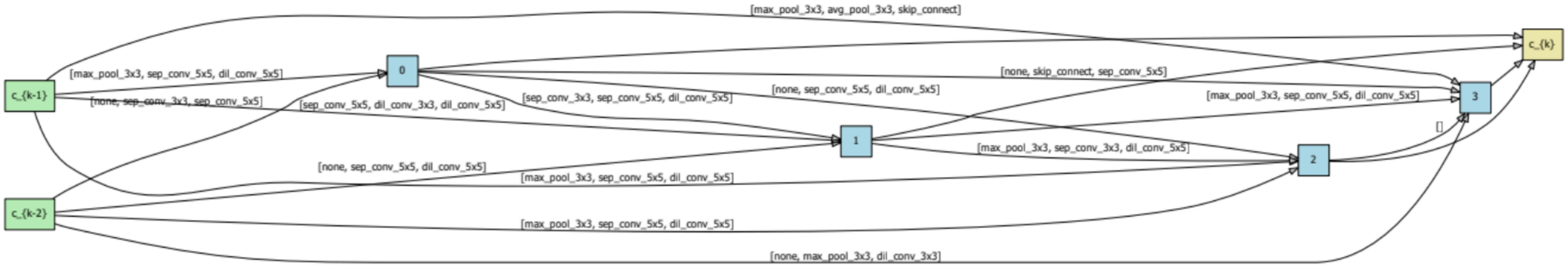


One-shot model | **operator weights α_i** | **interleaved optimization of α_i and ω_j with SGD** | **argmax α_i**

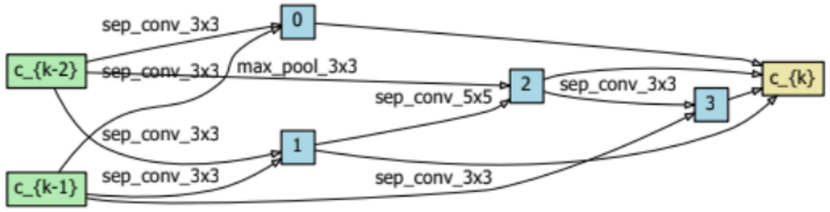
Figure source: Liu et al., 2018

Warm-started DARTS

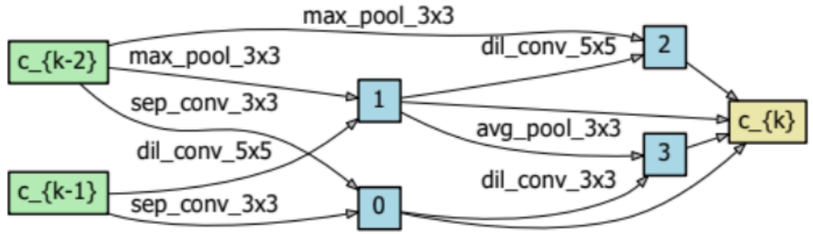
- Warm-start DARTS with architectures that worked well on similar problems
- Slightly better performance, but much faster (5x)



(a) FLOWER_V3 transfer architecture for normal cell



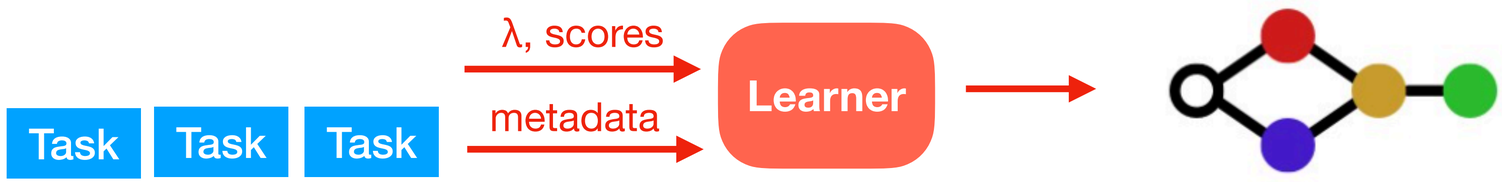
(b) Normal cell found on *aircraft* task



(c) Normal cell found on *birds* task

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]

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

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

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

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

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

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

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

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

[1] Ramachandran et al. 2017

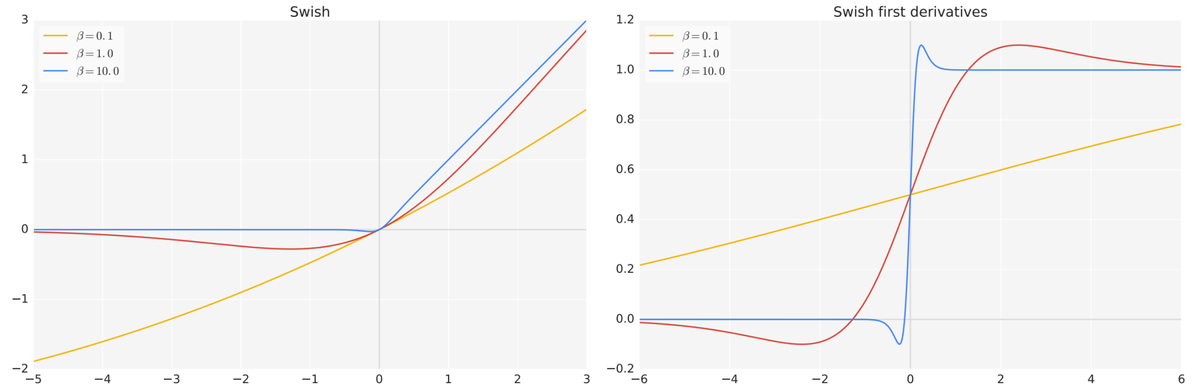
[2] Bello et al. 2017

[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

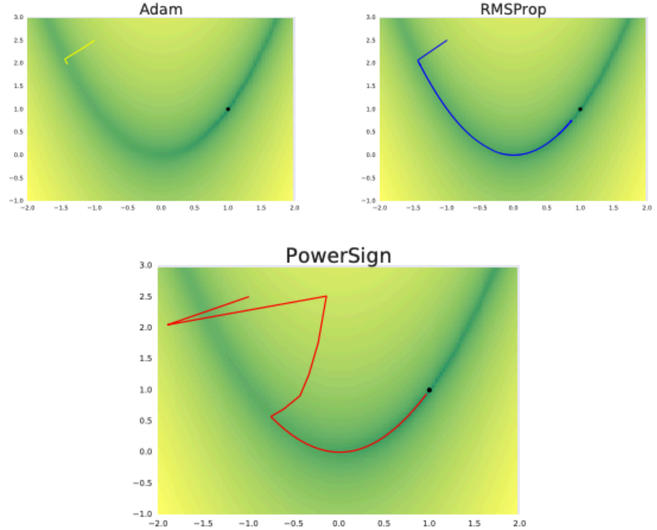
$$Swish : \frac{x}{1 + e^{-\beta x}}$$



- 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



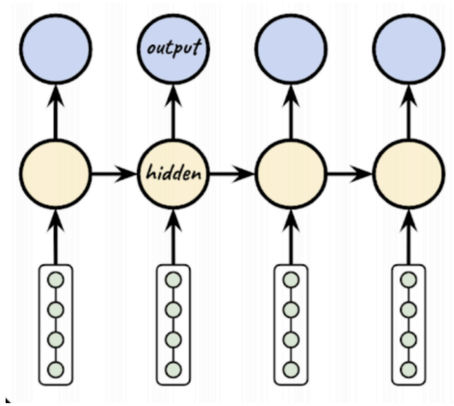
- Learn acquisition functions for Bayesian optimization [3]

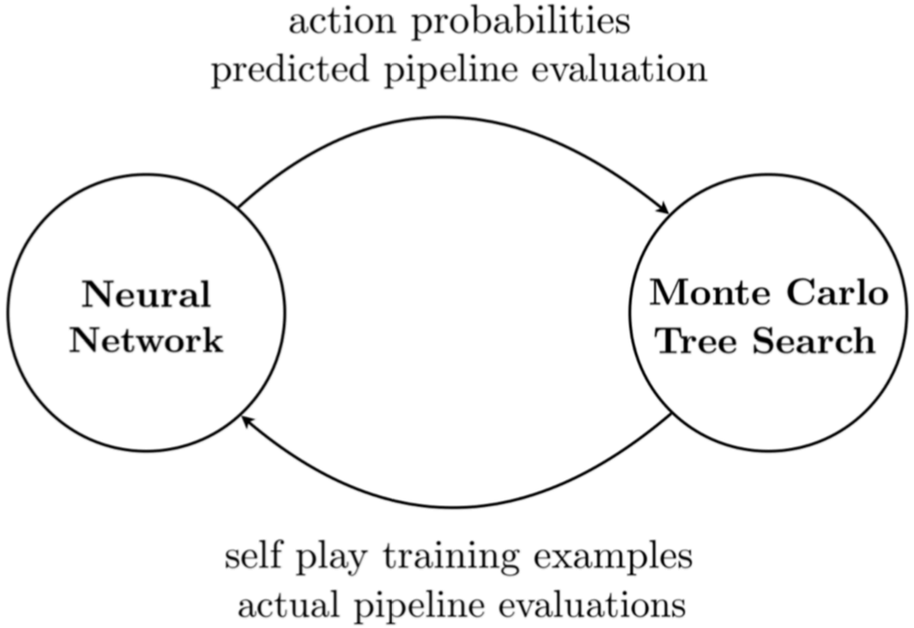
Figure source: Ramachandran et al., 2017 (top), Bello et al. 2017 (bottom)

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
    - With grammar to avoid invalid pipelines
  - Neural network (LSTM) Predicts pipeline performance (can be pre-trained on prior datasets)





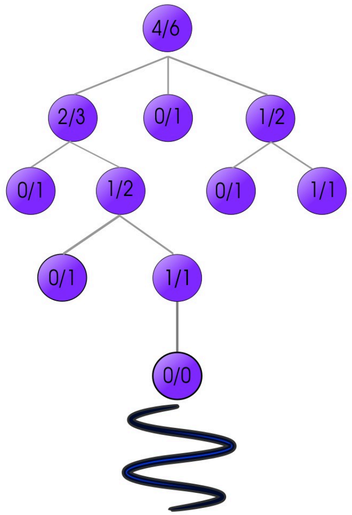


Figure source: Drori et al., 2019

Wong et al. 2018

Neural Architecture Transfer learning

- Warm-start a deep RL controller based on prior tasks
- Much faster than single-task equivalent

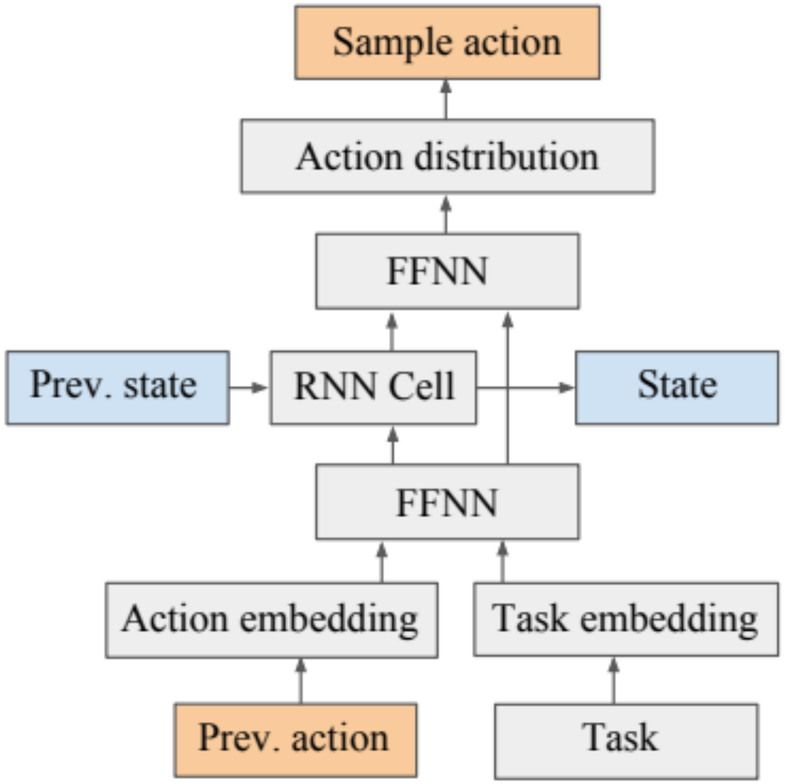


Figure source: Wong et al., 2018

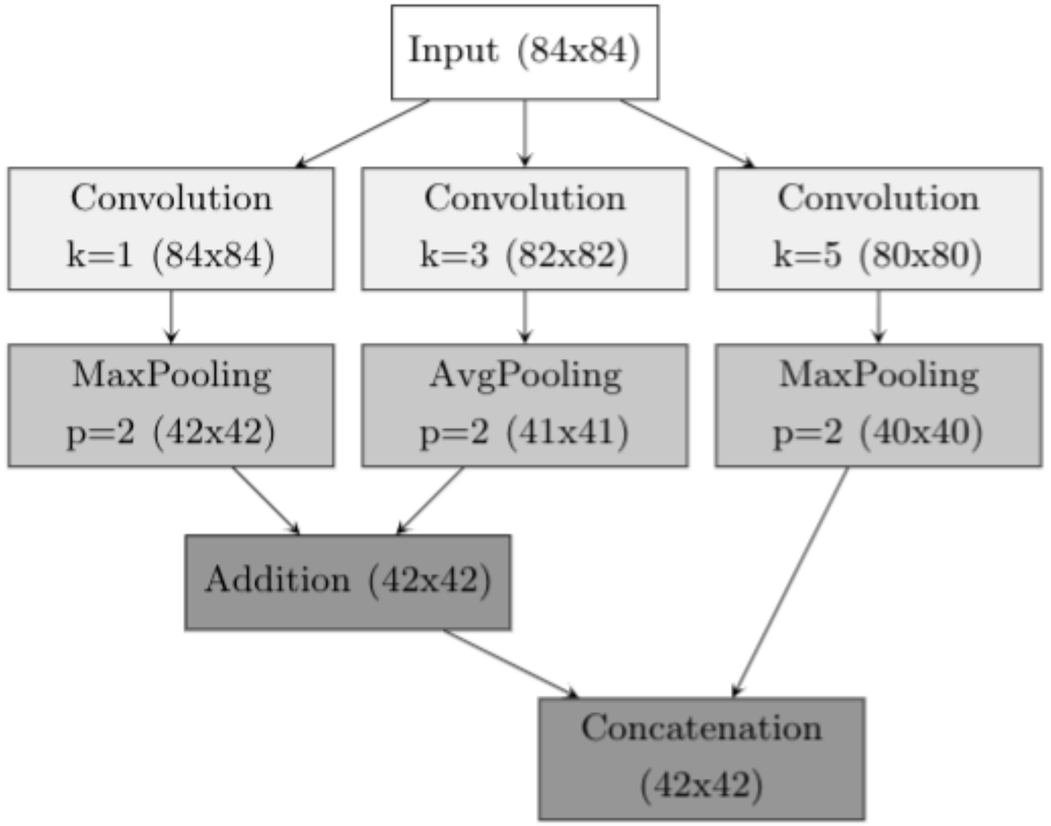
Meta-Reinforcement Learning for NAS

Gomez & Vanschoren, 2019

- Train an agent how to build a neural net, across tasks
- Should transfer but also adapt to new tasks

[0, 0, 0, 0, 0]
[0, 0, 0, 0, 0]
[1, 1, 1, 0, 0]
[2, 2, 2, 1, 0]
[3, 1, 3, 0, 0]
[4, 3, 2, 3, 0]
[5, 1, 5, 0, 0]
[6, 2, 2, 5, 0]
[7, 5, 0, 2, 4]
[8, 7, 0, 0, 0]

Actions: add/remove certain layers in certain locations

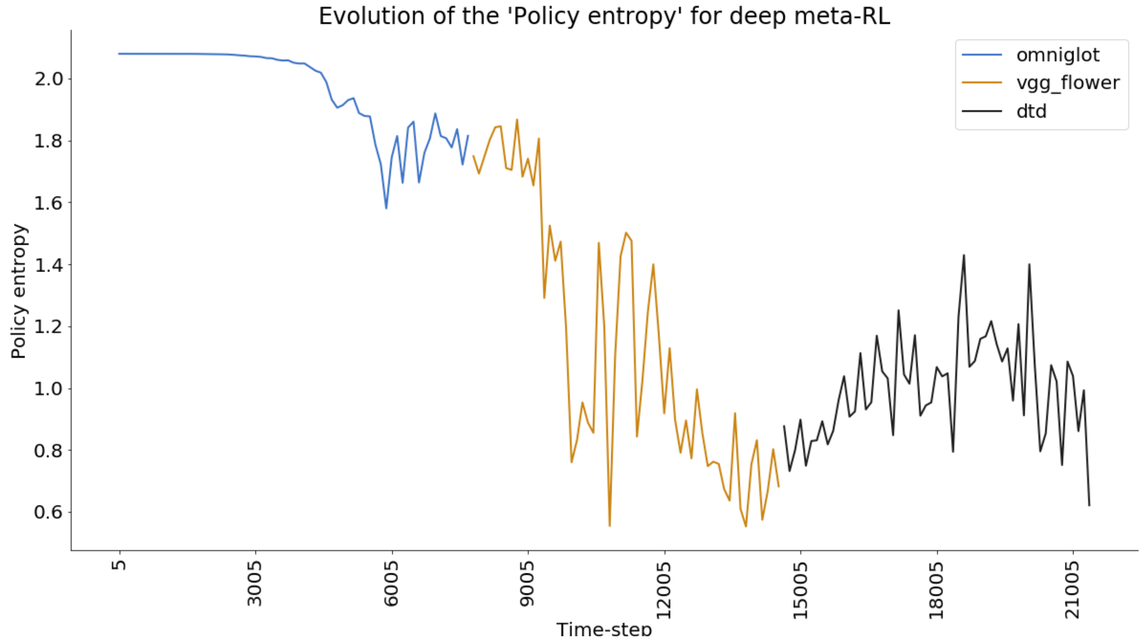


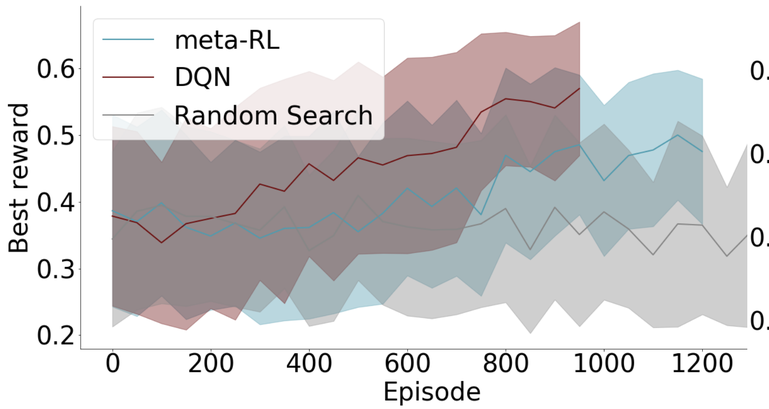
Gomez & Vanschoren, 2019

Meta-Reinforcement Learning for NAS

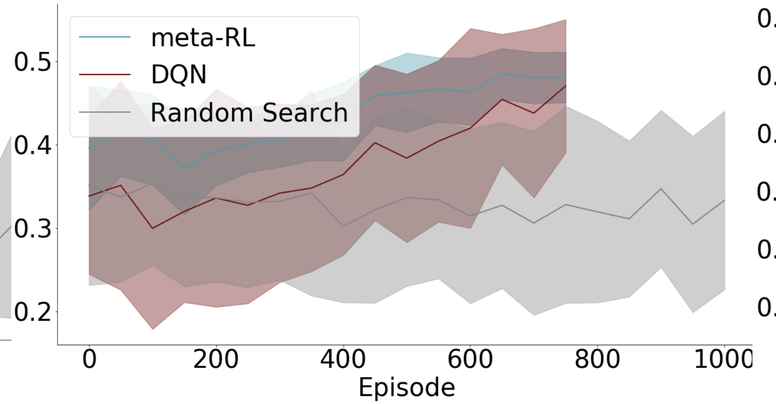
Results on increasingly difficult tasks:

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

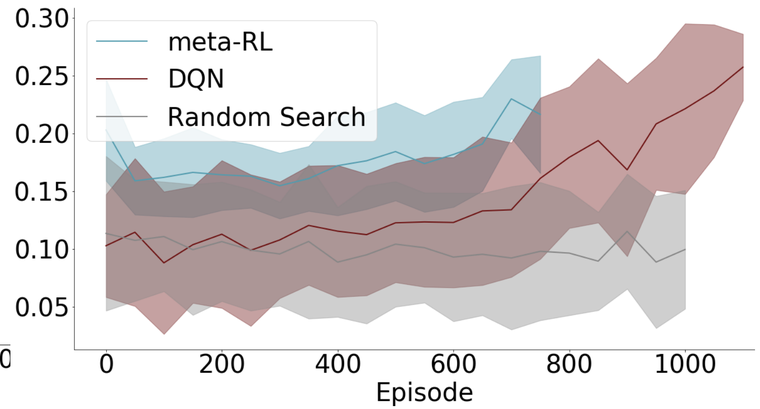




omniglot



vgg_flower



dtd

Elsken et al., 2020

MetaNAS: MAML + Neural Architecture Search

- Combines gradient based meta-learning (REPTILE) with NAS
- During meta-train, it optimizes the meta-architecture (DARTS weights) along with the meta-parameters (initial weights) Θ
- During meta-test, the architecture can be adapted to the novel task through gradient descent

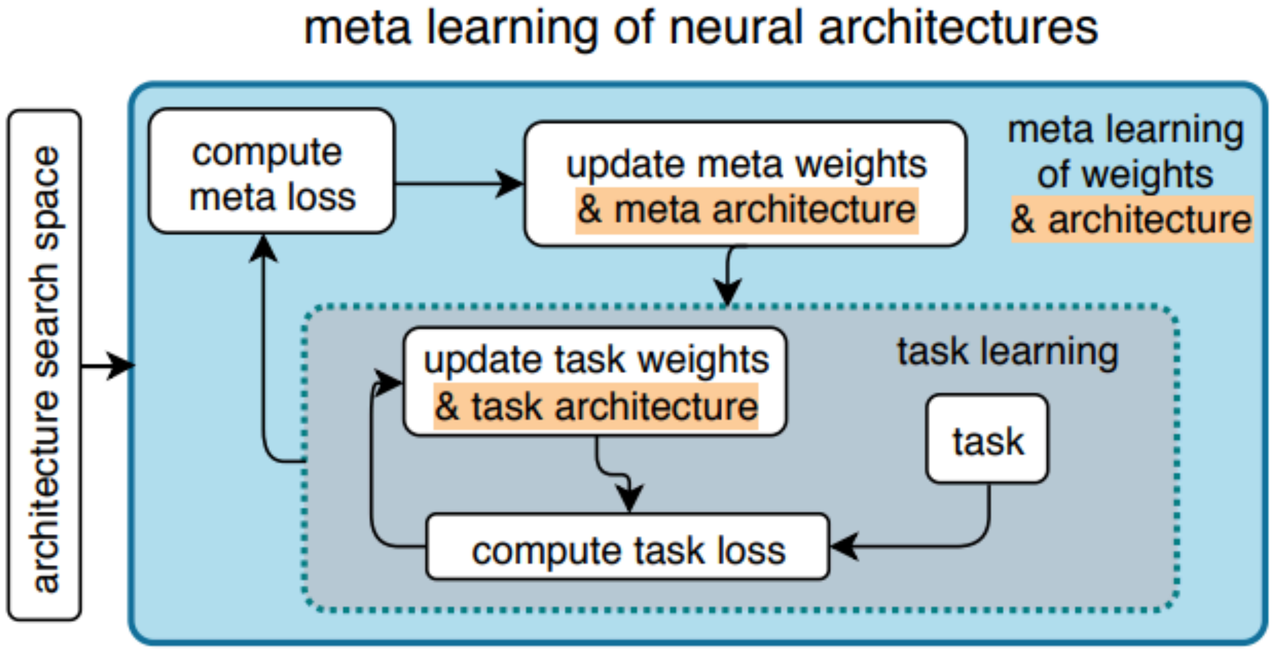


Figure source: Elsken et al., 2020

Thank you

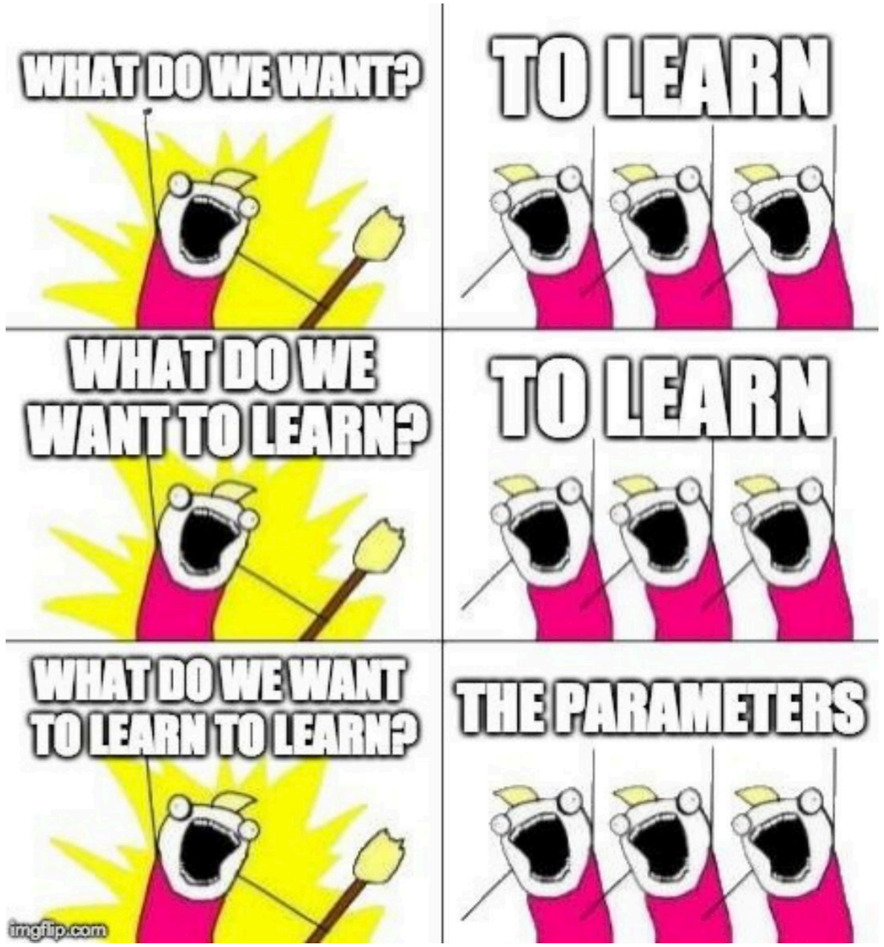


Image source: Pesah et al. 2018