université PARIS-SACLAY

Automated Machine Learning

Lisheng Sun-Hosoya
LIU Zhengying
Laboratoire en Recherche Informatique (LRI)
U. Paris-Sud / Inria / U. Paris Saclay

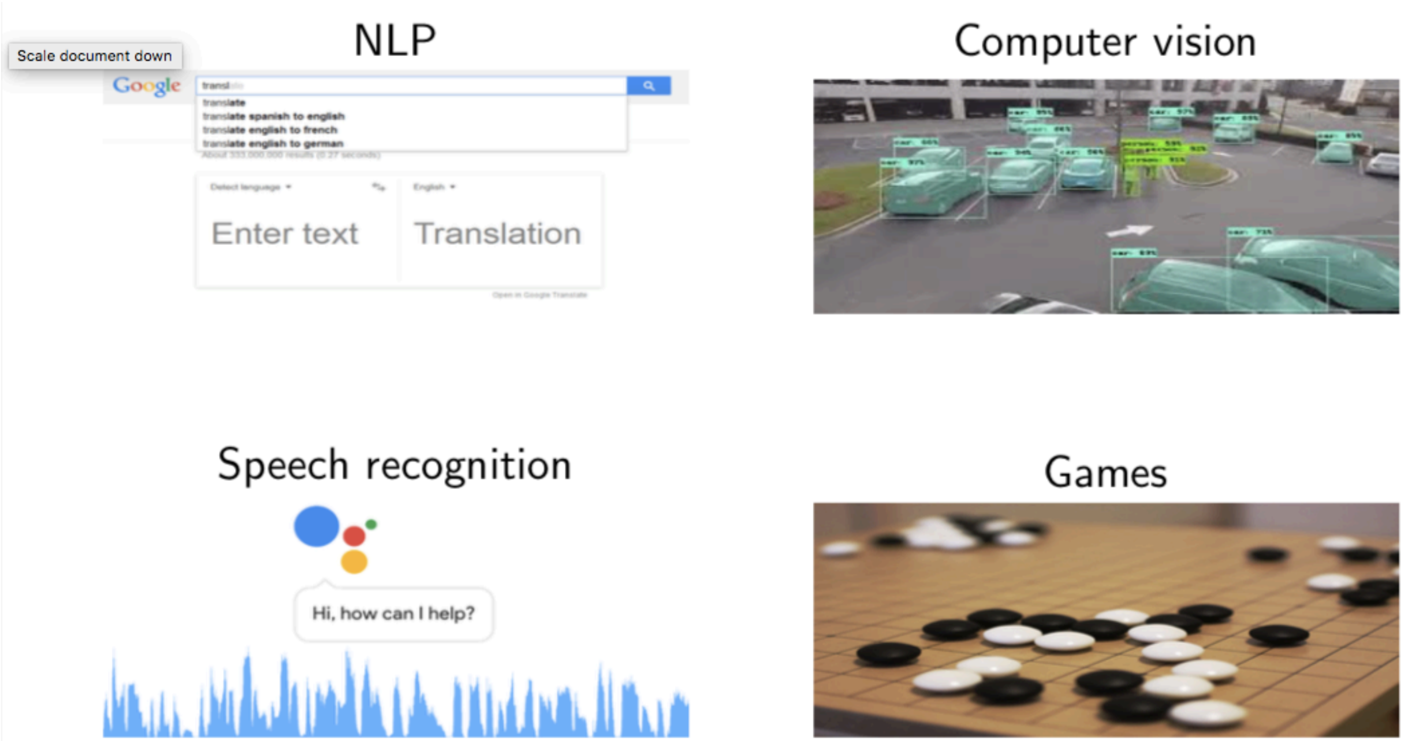
9 Mar 2021

Contents

- AutoML: an intro
- AutoML methods
  with application to Deep Learning
- AutoML challenges

AutoML: an intro

Successes of Machine learning



… relies on **extensive** and **manual** tuning of algorithms and their hyperparameters

Machine Learning

Machine Learning algorithm:
Decision Tree, CNN, SVM, etc

$$D = \{X_i, Y_i\} \xrightarrow{\beta_\lambda} \alpha_\theta \overset{D_{\text{va}}}{\Longrightarrow} P(\alpha_\theta)$$

(or $p_\theta(y\,|\,x)$)

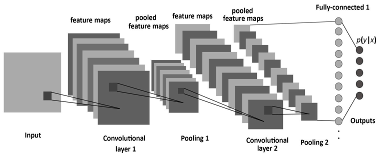
performance
(e.g. accuracy)

Dogs vs Cats dataset = {

"dog"

"cat"

... } $\mapsto$



trained CNN

CIFAR-10 dataset $\mapsto$ another trained CNN (for another A)

Iris dataset $\mapsto$ trained SVM (for another A)

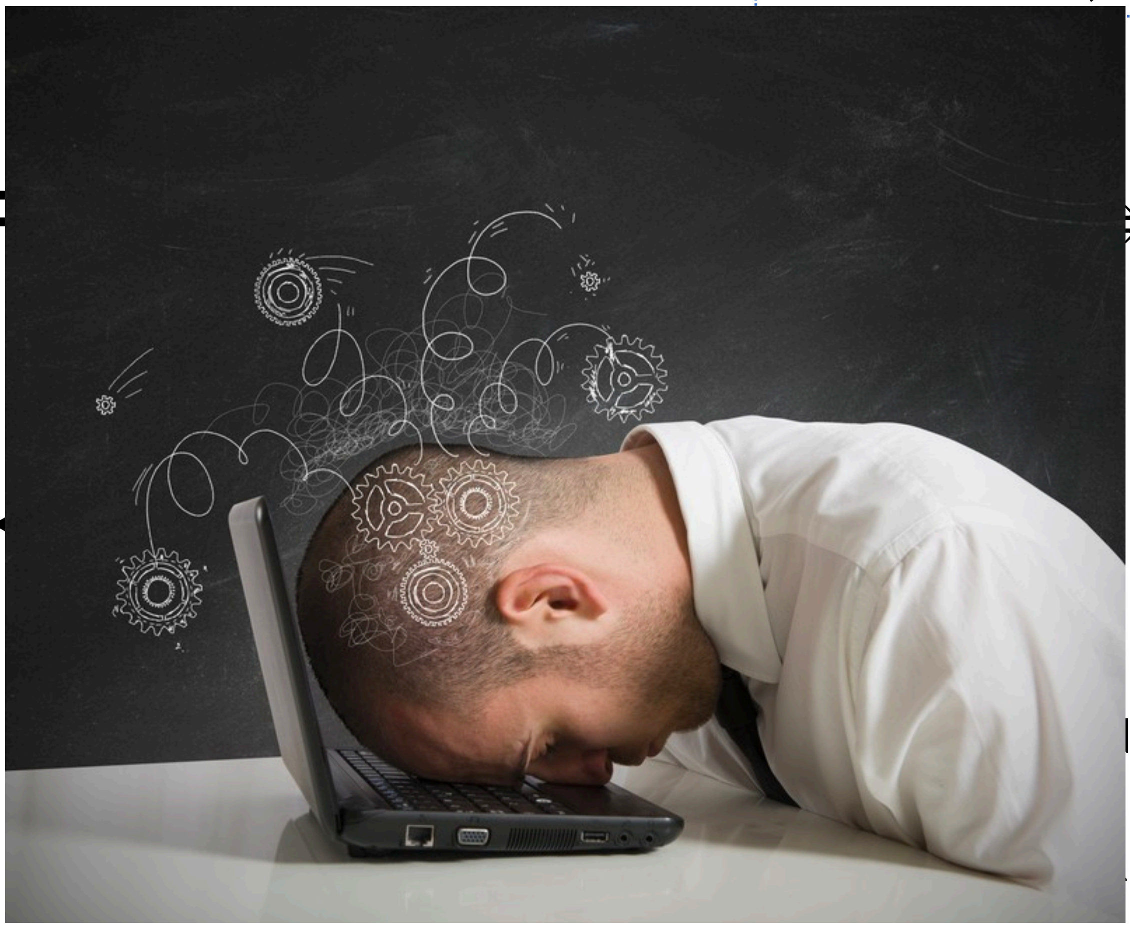
encoded by: hyperparameters $\lambda \in \Lambda$

hand-crafted by <u>ML experts</u>

Machine Learning

Machine Learning algorithm:
Decision Tree, CNN, SVM, etc

$D =$



$\Rightarrow \quad P(\alpha_\theta)$

performance
(e.g. accuracy)

Dogs vs Cats
dataset

=

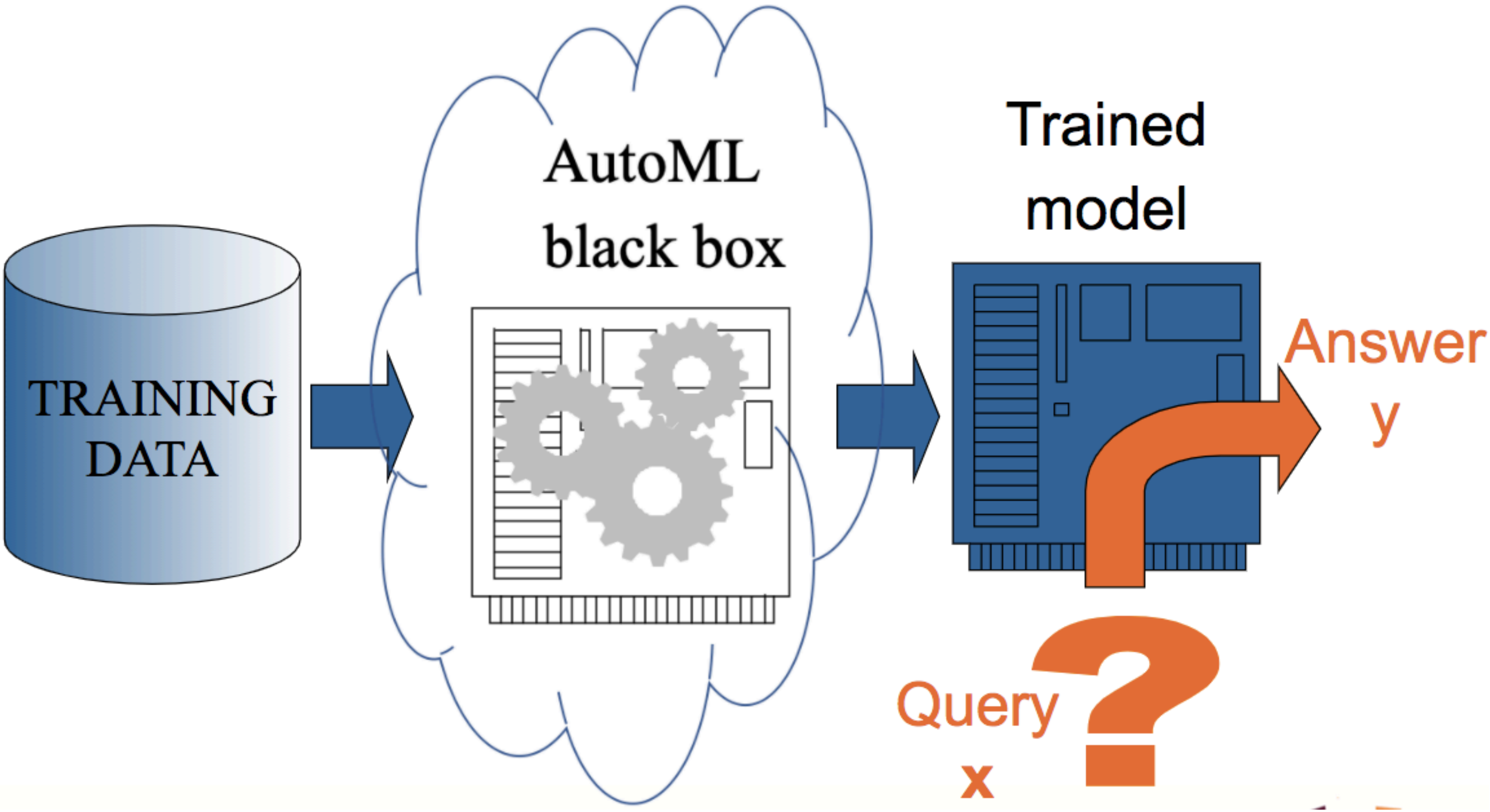
N (for another A)

nother A)

encoded by: hyperparameters $\lambda \in \Lambda$

hand-crafted by ML experts

Today's lecture



The AutoML problem: definition

$$\max_{\gamma} \sum_{D_{tr}, D_{te} \in \mathfrak{D}_{te}} P(\hat{\hat{\alpha}}; D_{te})$$ where $\hat{\hat{\alpha}} = \hat{\beta}(D_{tr})$ and $\hat{\beta} = \gamma(\mathfrak{D}_{tr})$

supervised learning

reinforcement learning

learning to learn ⇐ two layers of learning

$P(\hat{\hat{\alpha}}; D_{te})$ may involve **time** ⇒ computational efficiency: should be not only correct but also fast

initially we may have $\mathfrak{D}_{tr} = \varnothing$ ⇒ no prior experience BUT can be generated

$$(D_{tr}, \beta_1, \alpha_1, P_1)\ , (D_{tr}, \beta_2, \alpha_2, P_2)\ , (D_{tr}, \beta_3, \alpha_3, P_3)\ , \ldots$$

Table 7.1 **Supervised learning illustration of the three-level formulation.** An algorithm's level is entirely determined by its type of *input* and *output*. For a given task, finding a good α-level algorithm is the ultimate goal. γ-level algorithms exploit data from *all past experience*, in the form of a "meta-dataset", to allow us to select a better β-level algorithm, which in turn exploits the dataset of a given task to produce an α-level algorithm by training.

Level	Input	Output	Examples	Encoded by
α-level	sample or example (e.g. an image)	prediction of label (e.g. 'dog' or 'cat')	heuristically hard-coded classifier or already trained classifier	parameters, hyper-parameters (if any) and meta-parameters (if any)
β-level	task/dataset (e.g. MNIST, CIFAR-10)	α-level algorithm	learning algorithms (e.g. SVM, CNN); HPO algorithms (e.g. grid search cross-validation, SMAC [56], NAS [124])	hyper-parameters and meta-parameters (if any)
γ-level	meta-dataset (e.g. OpenML [115])	β-level algorithm	meta-learning algorithms (e.g. meta-learning part in Auto-sklearn [36]); **algorithms from this thesis.**	meta-parameters

Image from L. Sun. Meta-Learning as a Markov Decision Process. Machine Learning, 2019

AutoML: what's exciting?

- 100% autonomous
- Beat "no free lunch"
- Any time
- Any resource

➡ AI for everyone

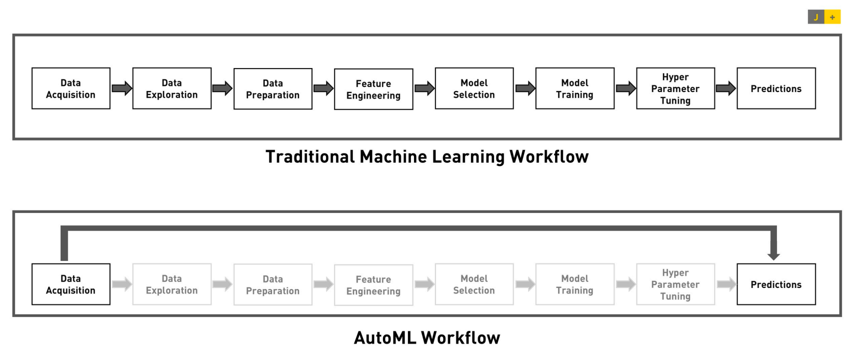
AutoML: a trending topic

Google's AutoML

AutoDL

AutoML.org

AutoML



AUTO KERAS

Auto-Sklearn

AutoML methods

with application to Deep Learning

We'll focus on the simplest case

$\mathfrak{D}_{tr} = \varnothing$ (initially) and $\mathfrak{D}_{te} = \{(D_{tr}, D_{te})\}$ (single dataset)

$\Rightarrow$ Hyperparameter Optimization

$\Rightarrow$ single fixed training dataset: D_{tr}

$\Rightarrow$ we only need to focus on $\beta_\lambda, \lambda \in \Lambda$

Reminder:

$$\max_{\gamma} \sum_{\substack{D_{tr}, D_{te} \\ \in \mathfrak{D}_{te}}} P(\hat{\hat{\alpha}}; D_{te})$$

where $\hat{\hat{\alpha}} = \hat{\beta}(D_{tr})$ and $\hat{\beta} = \gamma(\mathfrak{D}_{tr})$

Hyperparameter Optimization: a reformulation

an HPO algorithm aims to solve: $\max_{\lambda \in \Lambda} P(\hat{\alpha}; D_{te})$ where $\hat{\alpha} = \beta_\lambda(D_{tr})$

unknown test score: $P(\hat{\alpha}; D_{te})$ ⇨ use an estimation (e.g. CV): $\hat{P}(\lambda)$

so usually the problem becomes

$$\boxed{\max_{\lambda \in \Lambda} \hat{P}(\lambda)}$$

black-box optimization

expensive to compute

⇨ surrogate model (not discussed)

where

$$\hat{P} : \Lambda \to \mathbb{R}$$
$$\lambda \mapsto s = \hat{P}(\lambda) \approx P(\beta_\lambda(D_{tr}), D_{va})$$

is an estimation of the test score

Remark: some approaches optimize λ and θ at the same time ⇨ bi-level optimization (ex. DARTS H. Liu et al., 2018)

$\beta_\lambda, \lambda \in \Lambda$ encodes an architecture A

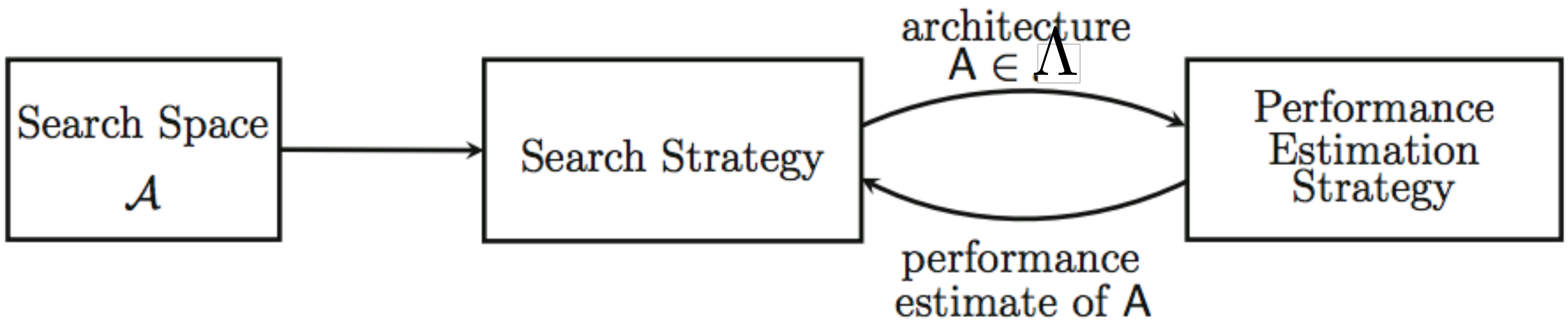


Image adapted from: Automated Machine Learning - Methods, Systems, Challenges, Frank Hutter et. al, (2018) Springer.

3 ingredients in HPO (NAS):

- Search space
- Search strategy
- Performance estimation strategy

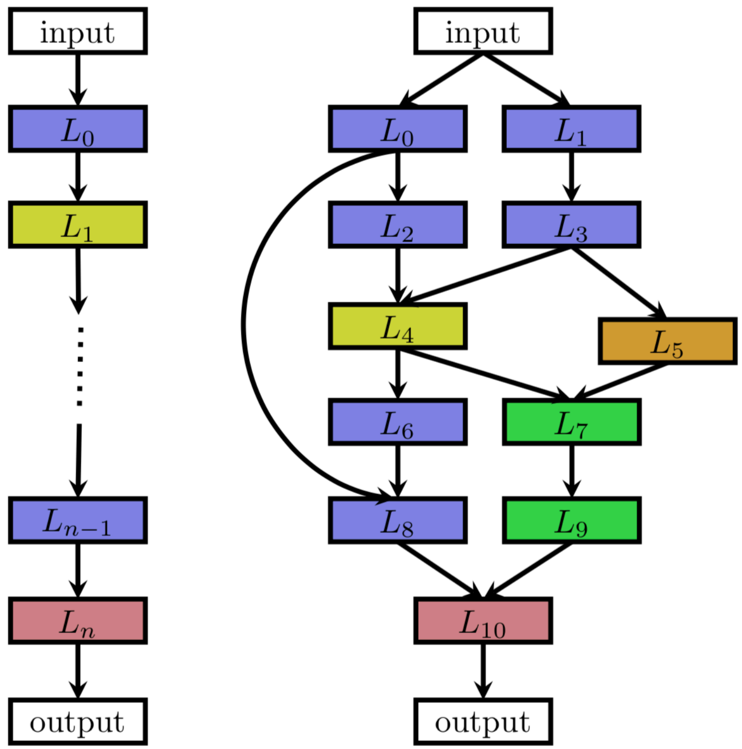
Search Space (for DL)

$\beta_\lambda, \lambda \in \Lambda$: architecture, optimizer, regularization, etc

chain-structured (feed-forward)

$\text{A} = L_n \circ L_{n-1} \circ \ldots \circ L_0$

$L_i^{in} = L_{i-1}^{out}$



multi-branch

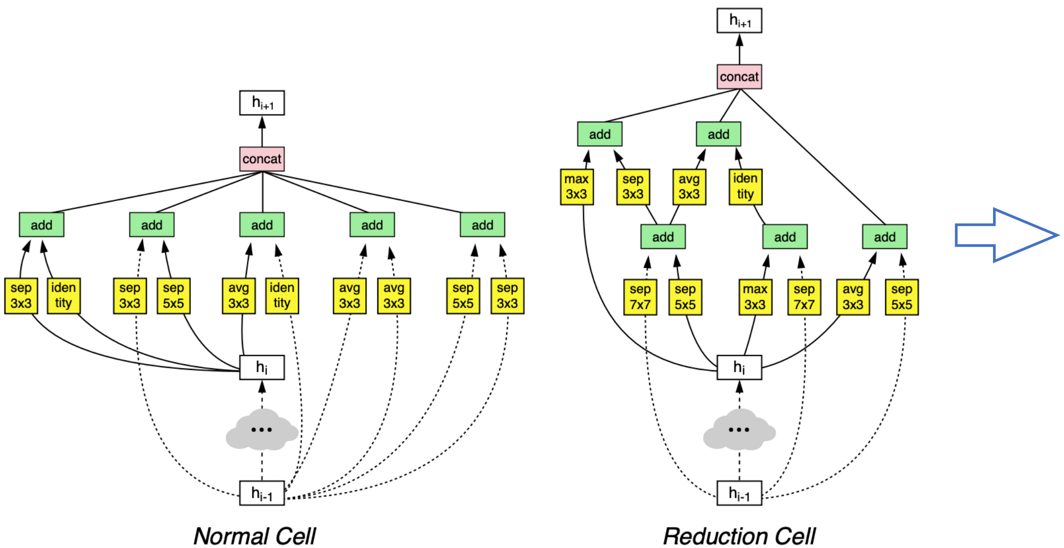
$L_i^{in} = g_i(L_{i-1}^{out}, \ldots, L_0^{out})$

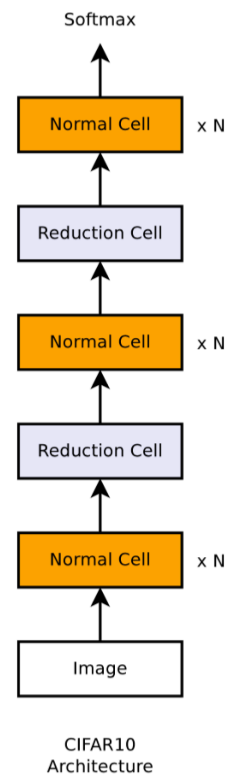
Different layer types are visualized by different colors.

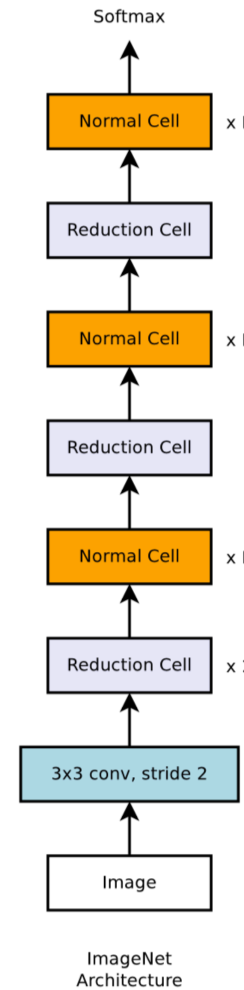
Automated Machine Learning - Methods, Systems, Challenges, Frank Hutter et. al, (2018) Springer.

Search Space (for DL)

observation: some approaches only use some building blocks (sub-modules): ResNes, Inception, ...



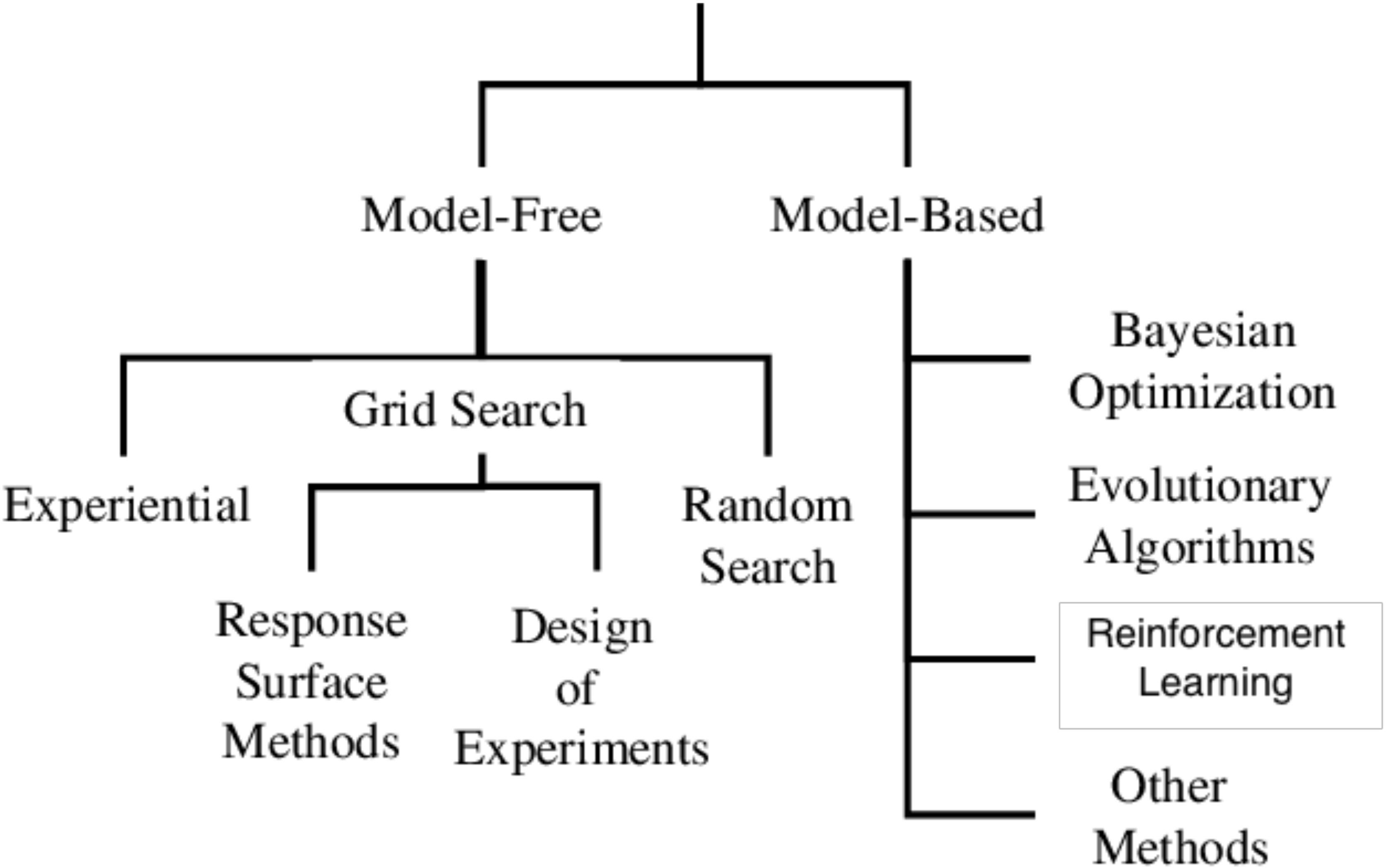




"NASNet search space" only uses two building blocks

Zoph B, Vasudevan V, Shlens J, Le QV. Learning Transferable Architectures for Scalable Image Recognition. *CVPR2018*

Search Strategy



Grid Search (exhaustive search)

$\Lambda = \Lambda_1 \times \Lambda_2$ with $\Lambda_1 = \{1,2,3,4\}$ and $\Lambda_2 = \{0.001,0.001,0.1,1\}$

neurons in hidden layer

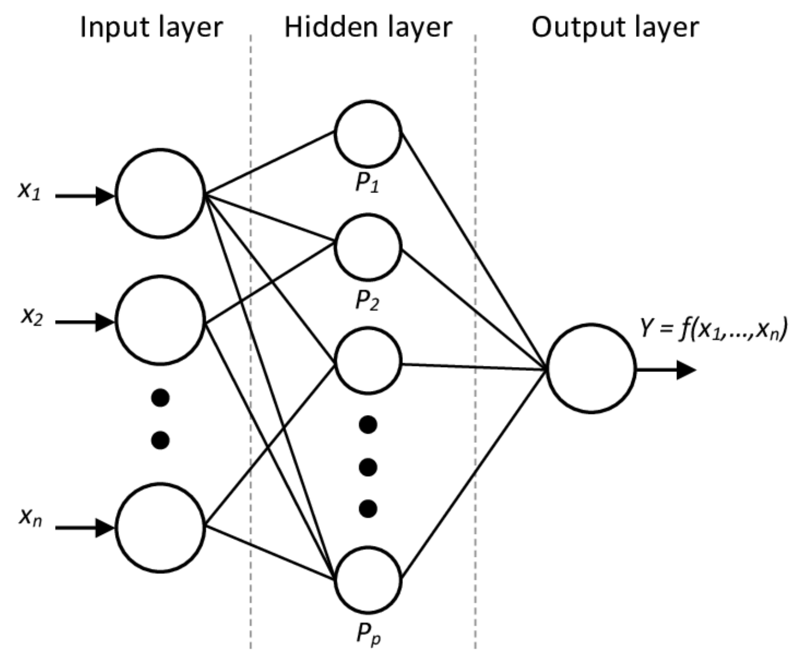
learning rate

try every possible combination in

$$\Lambda = \Lambda_1 \times \Lambda_2$$

evaluate it and return argmax in the end

curse of dimensionality!



Random Search

$$\Lambda = \Lambda_1 \times \Lambda_2 \text{ with } \Lambda_1 = \{1,2,3,4\} \text{ and } \Lambda_2 = \{0.001,0.001,0.1,1\}$$

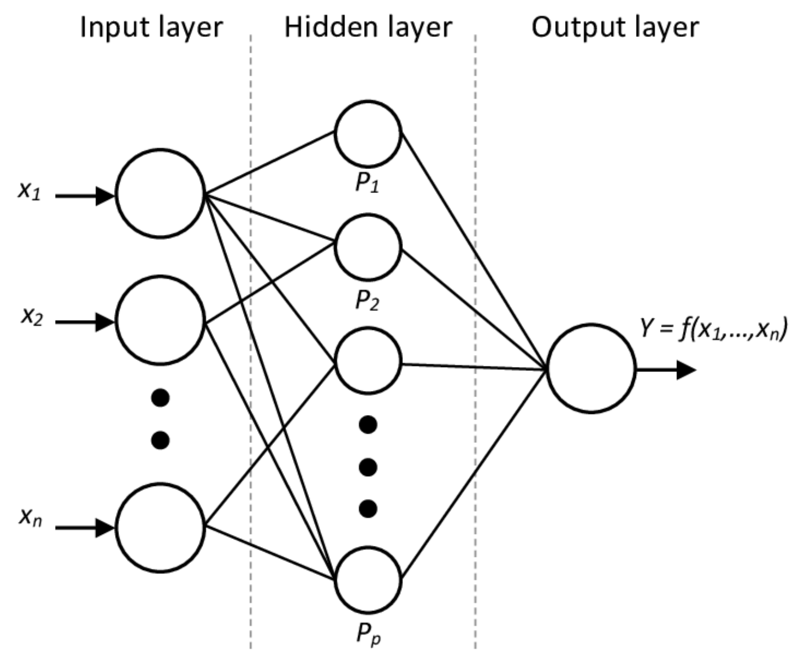
\# neurons in hidden layer

learning rate

Randomly sample certain number of combinations in

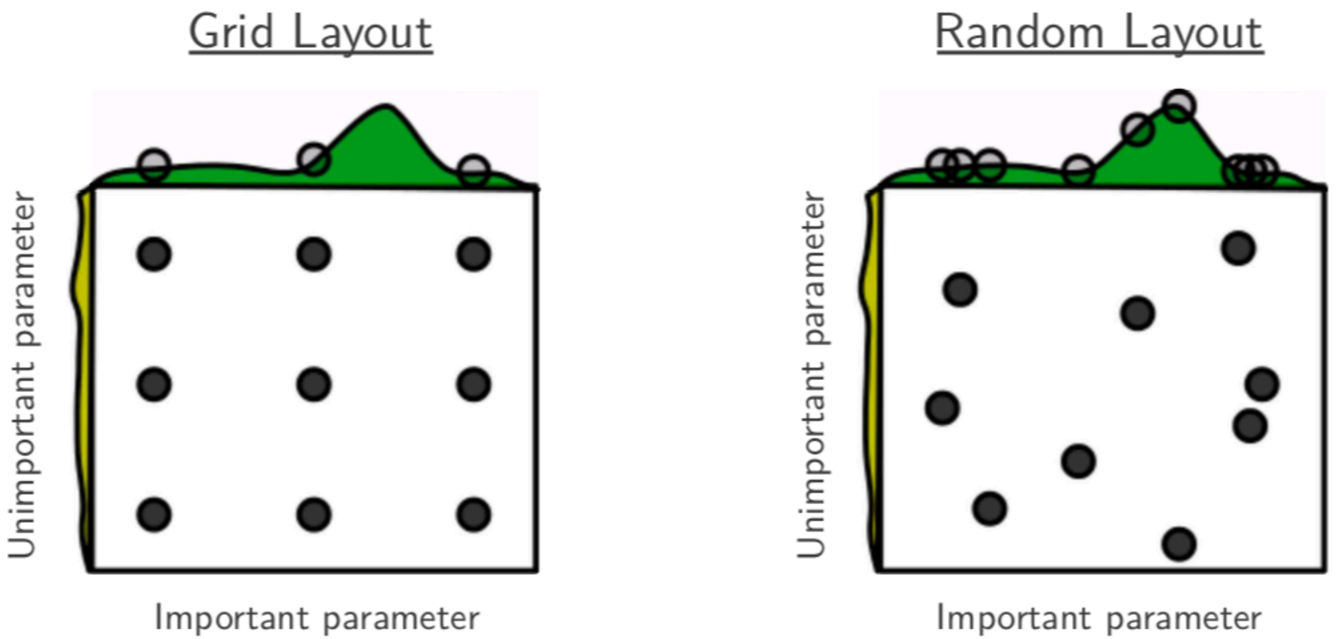
$$\Lambda = \Lambda_1 \times \Lambda_2$$

evaluate it and return argmax in the end



Grid Search and Random Search

two model-free black-box optimization methods



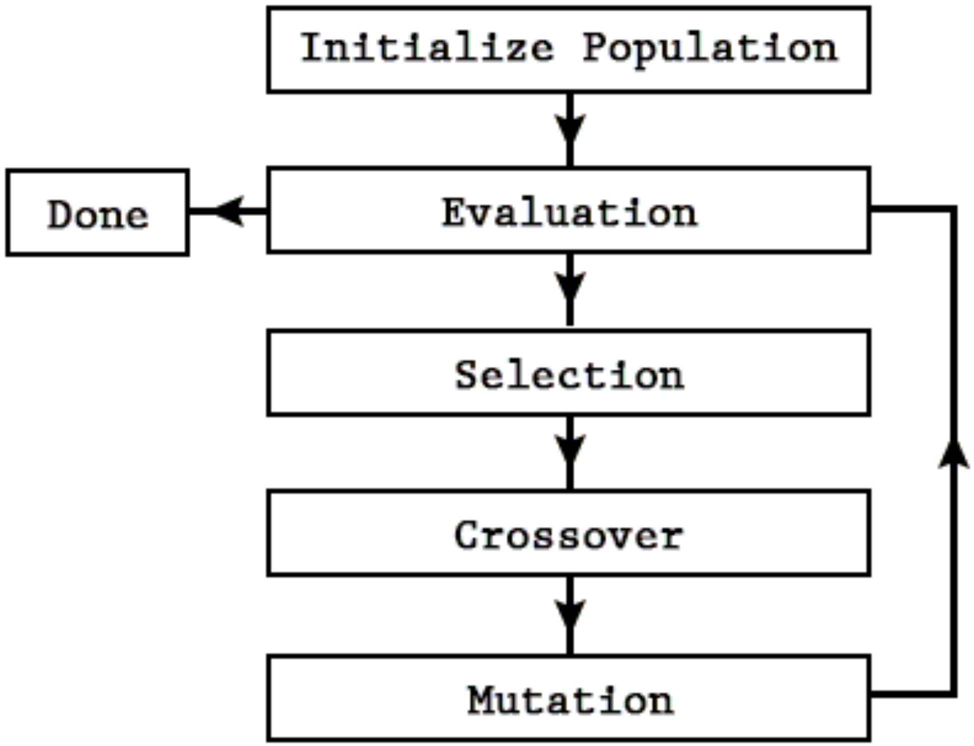
RS tends to perform better than GS when some HP are more important than others

Random Search provides already a strong HPO baseline (surprisingly...?)

Bergstra J, Bengio Y. **Random Search for Hyper-Parameter Optimization**. *JMLR*. 2012

Evolutionary Algorithms

Population-based derivative-free optimization methods



Optimize w.r.t a population (a set of points) or a distribution instead of one single point

Often encode an individual by "chromosome"

Explore new points by mutation or crossover

Select individuals by fitness

Just some vocabulary...but the idea is simple

Easy to parallelize

similar to: genetic algorithms, evolutionary strategies, particle swarm optimization

Evolutionary Algorithm: an example

Real E, Moore S, Selle A, et al. **Large-Scale Evolution of Image Classifiers**. *ICML2017*

1000 individuals

fitness: accuracy on validation dataset

pair-wise competition
(select two individuals and kill the weaker one)

the winner gets to reproduce and mutate

massively-parallel
(due to huge computation cost)

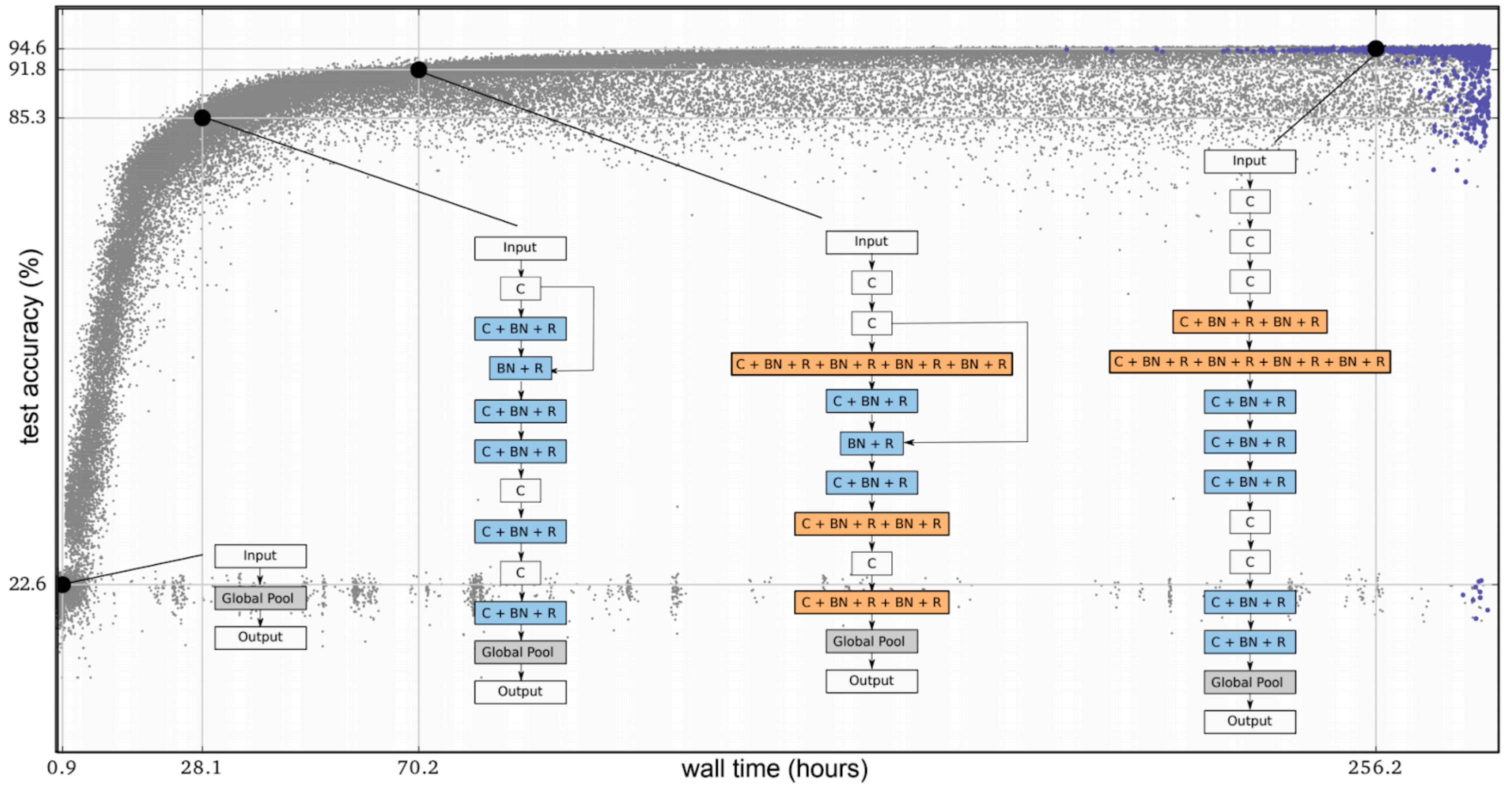
chromosome (DNA): tensor graph

begins from single layer individuals

possible mutations:

- ALTER-LEARNING-RATE
- IDENTITY
- RESET-WEIGHTS
- INSERT-CONVOLUTION
- REMOVE-CONVOLUTION.
- ALTER-STRIDE
- ALTER-NUMBER-OF-CHANNELS
- FILTER-SIZE
- INSERT-ONE-TO-ONE
- ADD-SKIP
- REMOVE-SKIP

Evolutionary Algorithm: an example



Real E, Moore S, Selle A, et al. Large-Scale Evolution of Image Classifiers. *ICML2017*

Bayesian Optimization

$$\max_{\lambda \in \Lambda} \hat{P}(\lambda) \qquad \text{with } \hat{P} : \Lambda \to \mathbb{R}, \quad \lambda \mapsto s$$

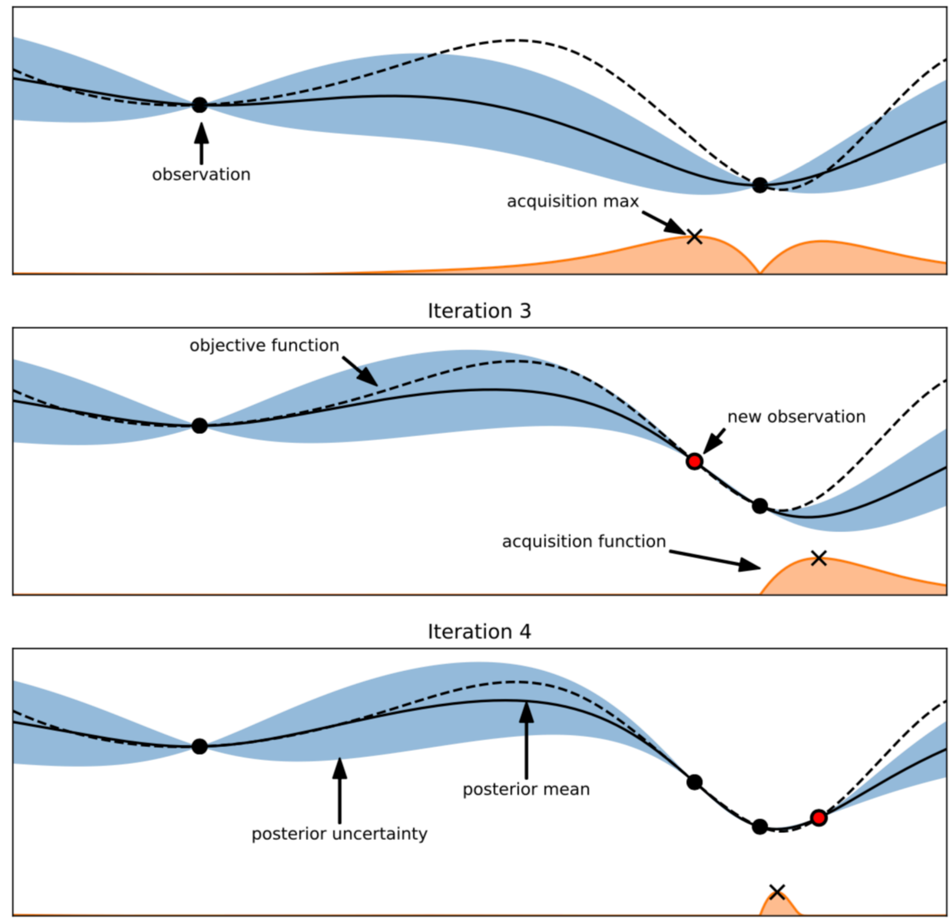
Original idea:

λ and $s = \hat{P}(\lambda)$ follow prior distributions $p(\lambda), p(s \,|\, \lambda)$

we choose next point to evaluate by maximizing an acquisition function (active learning-like)

we gain more information and update $p(\lambda)$ and $p(s \,|\, \lambda)$ (or $p(s, \lambda)$)

repeat until convergence



Automated Machine Learning - Methods, Systems, Challenges, Frank Hutter et. al, (2018) Springer.

Bayesian Optimization (cont'd)

$$\max_{\lambda \in \Lambda} \hat{P}(\lambda) \quad \text{with } \hat{P} : \Lambda \to \mathbb{R}, \quad \lambda \mapsto s$$

usual acquisition function:
Expected Improvement (EI)

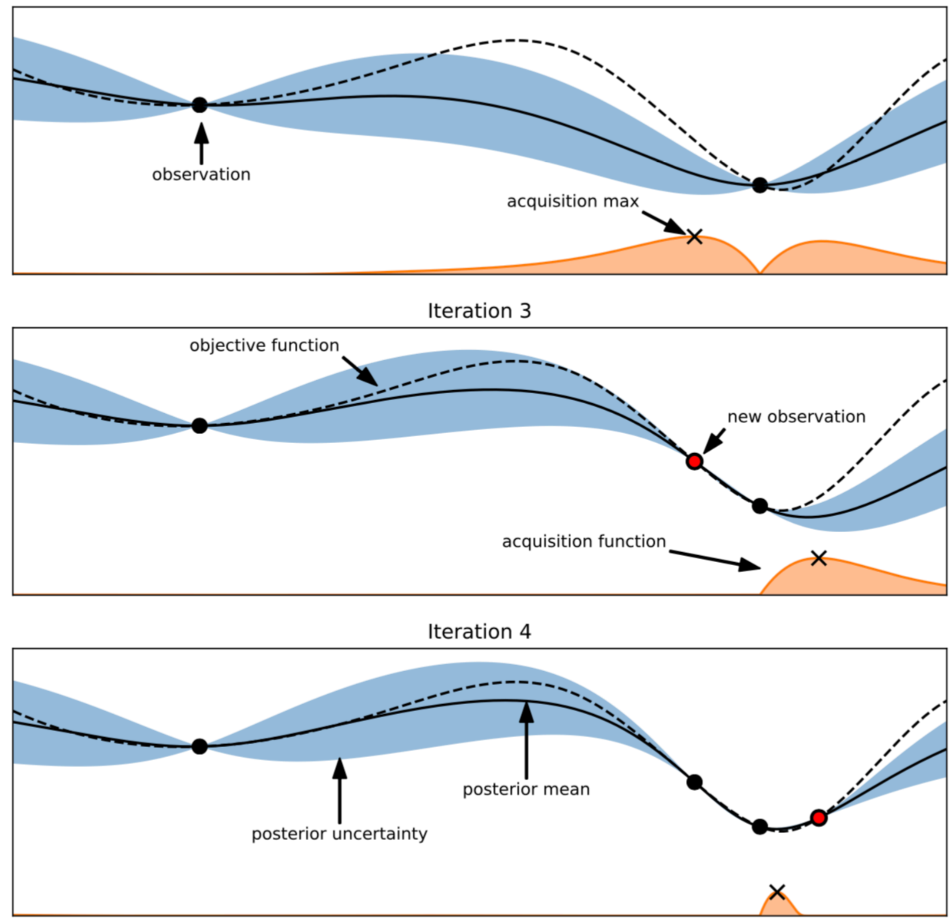
$$a_{EI}(\lambda \mid D_n) = \mathbb{E}[\max(\hat{P}(\lambda) - s_{\max}, 0)]$$

usual prior model:
Gaussian Process (GP)

but state-of-the-art tends to use tree-based classifier such as **Random Forest** to model

$$\hat{P}(\lambda) \text{ (or } p(s \mid \lambda) \text{)}$$

(thus not so Bayesian anymore...), see Auto-sklearn



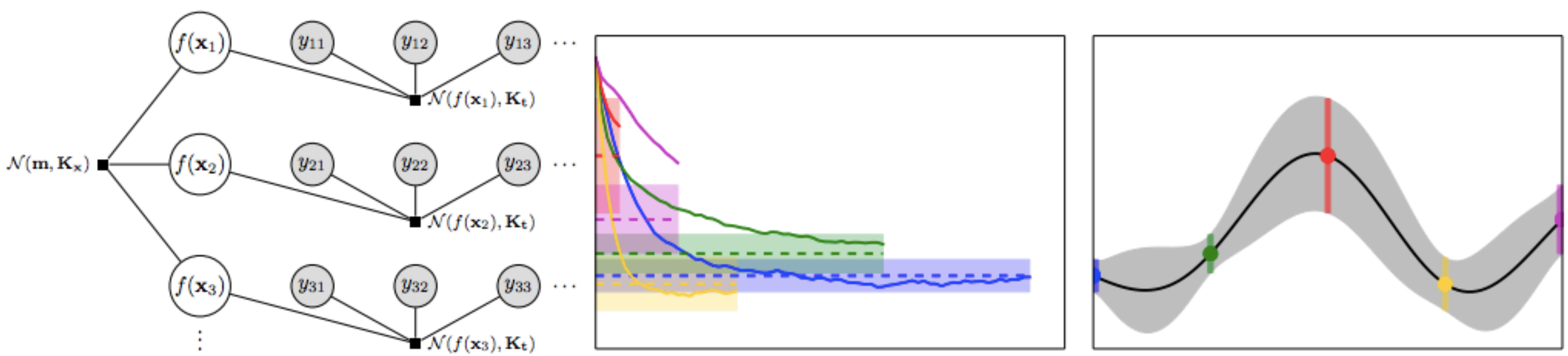
Automated Machine Learning - Methods, Systems, Challenges, Frank Hutter et. al, (2018) Springer.

Bayesian Optimization: an example

Swersky K, Snoek J, Ada;s RP. **Freeze-Thaw Bayesian Optimization**. 2014

Intuition:
Maintains a set of "frozen" (partially completed but not being actively trained) models and uses an information-theoretic criterion to determine which ones to "thaw" and continue training

Use Bayesian Optimization for:
- learning curve prediction —> offers quick evaluations
- HP space modeling



(a) Graphical Model

(b) Training curve predictions

$$p(f_{t+1}|\mathcal{D}_{1:t})$$

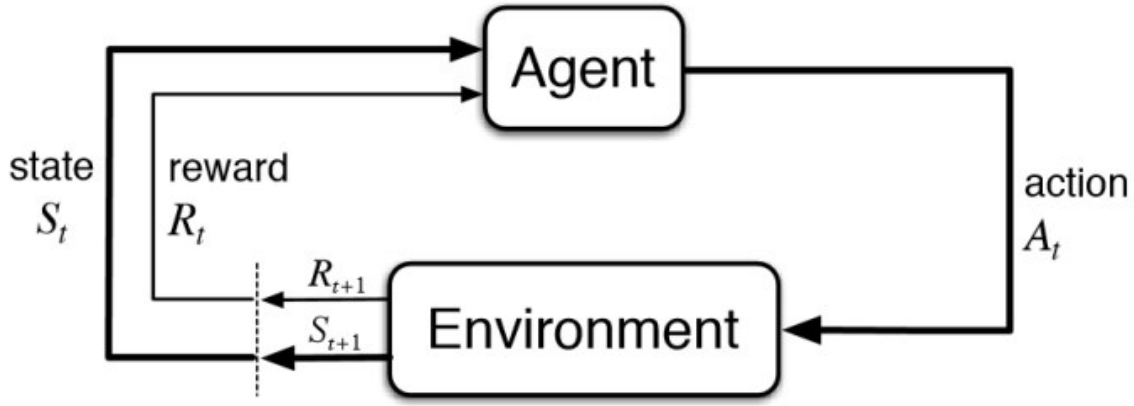
(c) Asymptotic GP

$$p(f_{new}|\mathcal{D}_{1:t}, x_{new})$$

use notation $f : x \mapsto y$ instead of $\hat{P} : \lambda \mapsto s$

Reinforcement Learning

A reminder:



State space: S

Action space: A

Transition model: $\mathscr{P}^a_{ss'} = p(s'|s,a) : S \times A \times S \rightarrow [0,1]$

Reward: $\mathscr{R}^a_{ss'} : S \times A \times S \rightarrow \mathbb{R}$

Goal: Learn a policy: $\pi(s,a) = p(a|s) : S \times A \rightarrow [0,1]$

that maximizes the (discounted) expected return

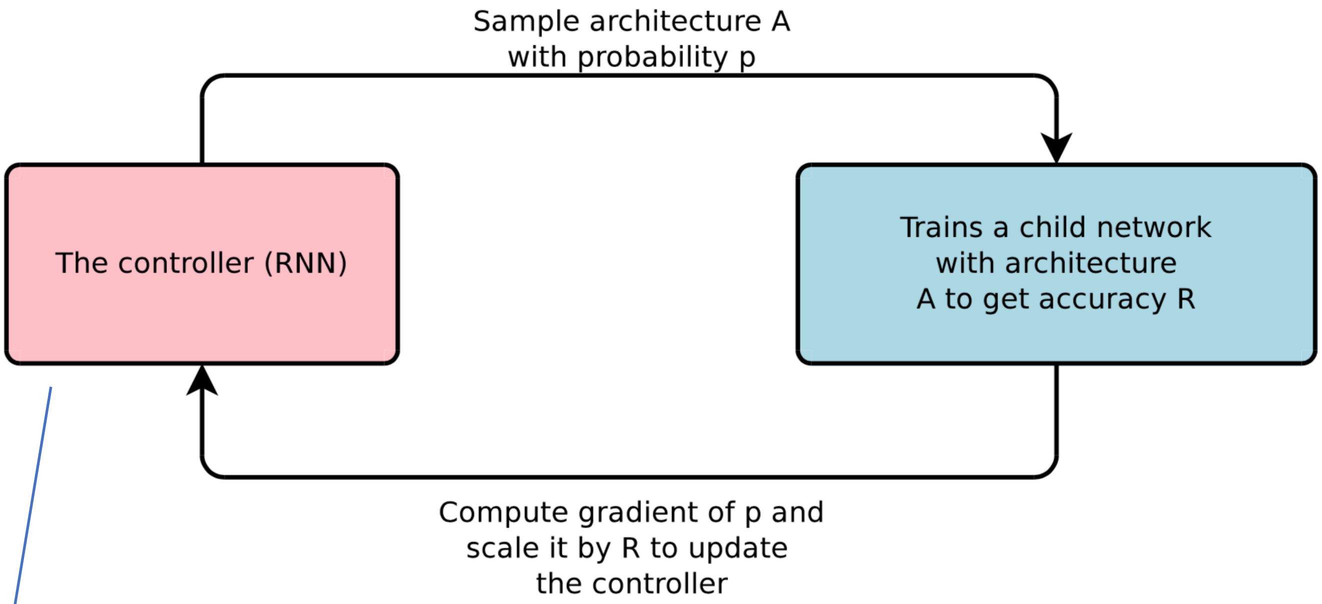
$$\mathbb{E}_\pi \left[\sum_{t=1}^{T} \gamma^t r_t \right]$$

with $T \in [0,+\infty], \gamma \in [0,1]$ and $s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, \ldots$ the agent's trajectory

Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Reinforcement Learning: an example

Zoph B, Le QV. **Neural Architecture Search with Reinforcement Learning**. ICLR 2017



Objective: $$J(\theta_c) = E_{P(a_{1:T};\theta_c)}[R]$$

REINCFORCE rule: $$\bigtriangledown_{\theta_c} J(\theta_c) = \sum_{t=1}^{T} E_{P(a_{1:T};\theta_c)}\big[\bigtriangledown_{\theta_c} \log P(a_t|a_{(t-1):1};\theta_c)R\big]$$

an estimation: $$\frac{1}{m}\sum_{k=1}^{m}\sum_{t=1}^{T} \bigtriangledown_{\theta_c} \log P(a_t|a_{(t-1):1};\theta_c)R_k$$

Summary

Method	Type	How to take next action	Update/Learn
Grid Search	model-free	loop over all choices (Cartesian product)	take max
Random Search	model-free	totally random	take max
Bayesian Optimization	sequential-based	maximizes acquisition function	update surrogate model
Evolutionary Algorithms	population-based	each individual randomly mutates	eliminate the weakest (with least fitness)
Reinforcement Learning	mixed/can be very general	according to learned policy	policy gradient method
Differentiable Methods	gradient-based	follow (negative) gradient	gradient descent

There is learning in EVERY method

Is there exploration-exploitation trade-off in each method?

How do we do benchmarking and fairly evaluate these methods?

⇨ AutoDL challenge!!!

Some other AutoML methods

Transfer Learning

Meta-learning

Ensemble methods
(competition winners)

embedded methods*: bi-level optimization methods
(related to transfer learning)

filter methods*: narrowing down the model space, without training the learning machine
(related to meta-learning)

* Guyon I, Bennett K, Cawley G, et al. Design of the 2015 ChaLearn AutoML challenge. *IJCNN 2015*

From one to multiple datasets: meta-learning

Given:

- Algorithms $j = 1, \ldots, m$
- PAST datasets $i = 1, \ldots, n-1$
- a NEW dataset n

Meta-dataset: $\mathbf{S}$ where $\mathbf{S}(i, j)$ = score of algo. j applied on dataset i.

Find

$$\operatorname{argmax}_{j=1,\ldots,m} \mathbf{S}(n, j)$$

I.e. We want to learn some transferable knowledge across datasets (a meta-learning model γ), to solve a new dataset better and faster.

* Sun-Hosoya. Meta-learning as a Markov decision process. *2019*

Meta-Learning: 1st trial with Auto-sklearn

Feurer M, Klein A, Eggensperger K, Springenberg JT, Blum M, Hutter F. **Efficient and Robust Automated Machine Learning**. 2015

Intuition:
Warm start the BO with meta-learning techniques, ensemble the top models.

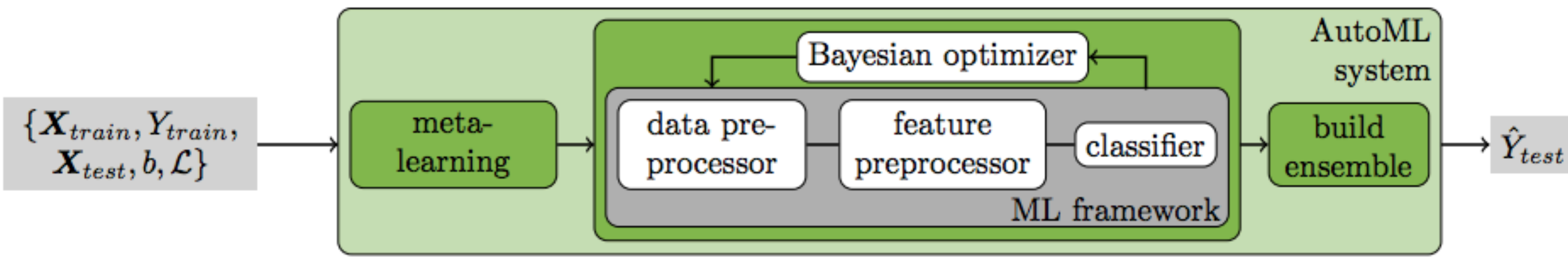


Figure 1: Our improved approach to AutoML. We add two components to Bayesian hyperparameter optimization of an ML framework: meta-learning for initializing the Bayesian optimizer and automated ensemble construction from configurations evaluated during optimization.

Meta-learning [Brazdil et al., 2009]:
- characterize the dataset using meta-features,
- Initialize BP with config. That performed well on old similar dataset

BO subroutine: SMAC [Hutter et al. 2011]:
- Random Forest prior
- Expected improvement acquisition
- 1 fold quick evaluation

Meta-Learning: example 2

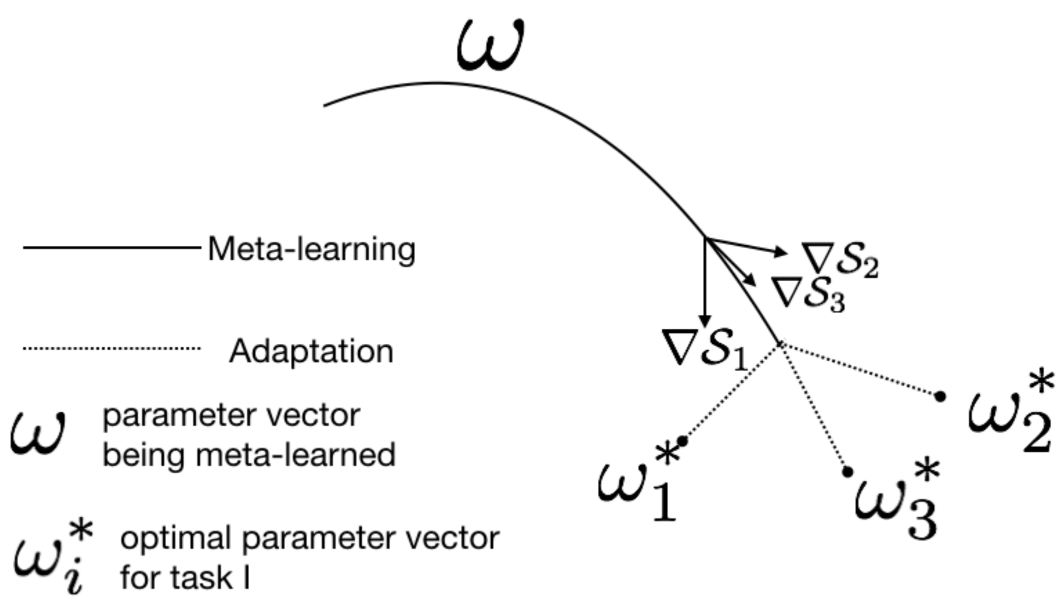
Model-Agnostic Meta-Learning [Finn et al. 2017]

- Assumption: a single learning algorithm (NN)
- Setting: Given a distribution of datasets noted $\mathbf{D}$; with $\boldsymbol{\omega}_i$ the optimal model for D_i

MAML finds a generally good solution:

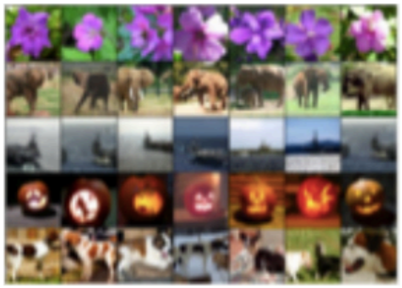
$$\boldsymbol{\omega} = \operatorname{argmax} \sum_{D_i} s_{D_i}(\boldsymbol{\omega} - \alpha \nabla_{\omega} s_{D_i})$$

This solution is used as starting point for the new pb.



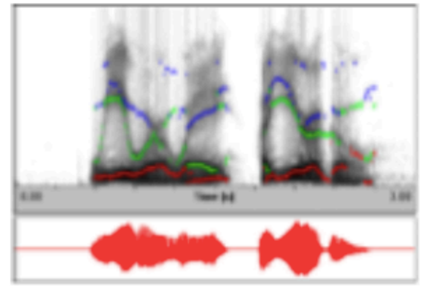
AutoML challenges

The AutoML challenge (Guyon et al., 2015-2016)



image

medical



speech

marketing

Task variabilities:

- classification / regression
- various scoring functions
- various time budget
- etc.

Goal: Find a process to identify the best $\boxed{\beta_\lambda}$ for each task

[1]: Design of the 2015 ChaLearn AutoML challenge, Guyon et al., 2015
[2]: Lessons learned from the AutoML challenge, Sun-Hosoya, Guyon and Sebag, 2018

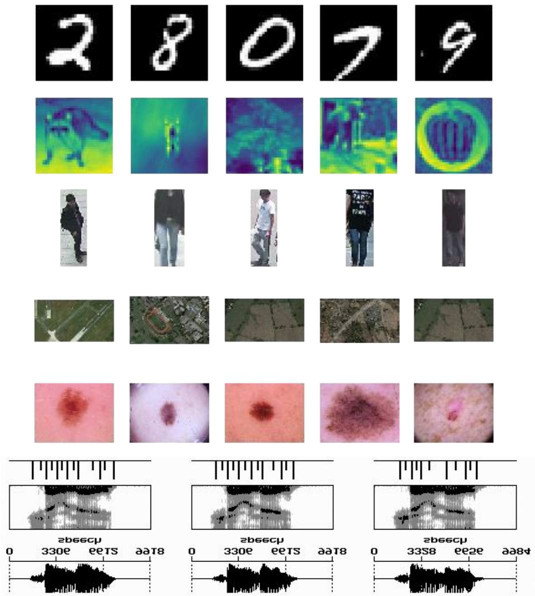
After the AutoML challenge series

http://automl.chalearn.org/

AutoDL

https://autodl.chalearn.org/

AutoDL challenge 2019-2020



#	Dataset	Challenge(s)	Phase	Domain	Type	Class num.	Sample number train	Sample number test	Tensor dimension time	Tensor dimension row	Tensor dimension col	Tensor dimension chnl
1	Munster	AutoCV1	public	HWR	image	10	60000	10000	1	28	28	1
2	Chucky	AutoCV1	public	objects	image	100	48061	11939	1	32	32	3
3	Pedro	AutoCV1	public	people	image	26	80095	19905	1	var	var	3
4	Decal	AutoCV1	public	aerial	image	11	634	166	1	var	var	3
5	Hammer	AutoCV1	public	medical	image	7	8050	1965	1	600	450	3
6	Ukulele	AutoCV1	feedback	HWR	image	3	6979	1719	1	var	var	3
7	Caucase	AutoCV1	feedback	objects	image	257	24518	6089	1	var	var	3
8	Beatriz	AutoCV1	feedback	people	image	15	4406	1094	1	350	350	3
9	Saturn	AutoCV1	feedback	aerial	image	3	324000	81000	1	28	28	4
10	Hippocrate	AutoCV1	feedback	medical	image	2	175917	44108	1	96	96	3
11	Loukoum	AutoCV1 AutoCV2	final final	HWR	image	3	27938	6939	1	var	var	3
12	Tim	AutoCV1	final	objects	image	200	80000	20000	1	32	32	3
13	Apollon	AutoCV1 AutoCV2 **AutoDL**	final final feedback	people	image	100	6077	1514	1	var	var	3
14	Ideal	AutoCV1 AutoCV2	final feedback	aerial	image	45	25231	6269	1	256	256	3
15	Ray	AutoCV1 **AutoDL**	final final	medical	image	7	4492	1114	1	976	976	3
16	Kraut	AutoCV2	public	action	video	4	1528	863	var	120	160	1
17	Katze	AutoCV2	public	action	video	6	1528	863	var	120	160	1
18	Kreatur	AutoCV2	public	action	video	4	1528	863	var	60	80	1
19	Freddy	AutoCV2	feedback	HWR	image	2	546055	136371	1	var	var	3
20	Homer	AutoCV2	feedback	action	video	12	1354	353	var	var	var	3
21	Isaac2	AutoCV2	feedback	action	video	249	38372	9561	var	102	78	1
22	Formula	AutoCV2	feedback	misc.	video	4	32994	8203	var	80	80	3
23	Fiona	AutoCV2 **AutoDL**	final final	action	video	6	8038	1962	var	var	var	3
24	Monica1	AutoCV2 **AutoDL**	final feedback	action	video	20	10380	2565	var	168	168	3
25	Kitsune	AutoCV2	final	action	video	25	18602	4963	var	46	82	3
26	data01	AutoSpeech	public	speech	time	100	3000	3000	var	1	1	1
27	data02	AutoSpeech	public	speech	time	7	428	107	var	1	1	1
28	data03	AutoSpeech	public	speech	time	3	796	200	var	1	1	1
29	data04	AutoSpeech	public	speech	time	20	939	474	var	1	1	1
30	data05	AutoSpeech	public	speech	time	10	199	597	var	1	1	1
31	data11	AutoSpeech	feedback	speech	time	55	1300	2000	var	1	1	1
32	data12	AutoSpeech	feedback	speech	time	5	3120	346	var	1	1	1
33	data13	AutoSpeech	feedback	speech	time	3	5000	1330	var	1	1	1
34	data14	AutoSpeech	feedback	speech	time	8	767	191	var	1	1	1
35	data15	AutoSpeech	feedback	speech	time	76	2286	571	var	1	1	1
36	data21	AutoSpeech	final	speech	time	50	800	1200	var	1	1	1
37	data22	AutoSpeech	final	speech	time	4	2649	294	var	1	1	1
38	data23 Oreal	AutoSpeech **AutoDL**	final final	speech	time	3	2000	264	var	1	1	1
39	data24	AutoSpeech	final	speech	time	16	384	386	var	1	1	1
40	data25 Sahak	AutoSpeech **AutoDL**	final feedback	speech	time	100	3008	752	var	1	1	1
41	O1	AutoNLP	public	english	text	2	7792	1821	var	1	1	1
42	O2	AutoNLP	public	english	text	20	11314	7532	var	1	1	1
43	O3	AutoNLP	public	english	text	2	60000	40000	var	1	1	1
44	O4	AutoNLP	public	chinese	text	10	55000	10000	var	1	1	1
45	O5	AutoNLP	public	chinese	text	18	156000	72000	var	1	1	1
46	PU1	AutoNLP	feedback	english	text	9	2822	499	var	1	1	1
47	PU2	AutoNLP	feedback	english	text	5	132651	23409	var	1	1	1
48	PU3	AutoNLP	feedback	chinese	text	2	1110203	195919	var	1	1	1
49	PU4	AutoNLP	feedback	chinese	text	11	100000	50000	var	1	1	1
50	PU5	AutoNLP	feedback	chinese	text	31	600000	400000	var	1	1	1
51	PR1	AutoNLP	final	english	text	20	33807	5967	var	1	1	1
52	PR2 Tanak	AutoNLP **AutoDL**	final feedback	english	text	2	42500	7501	var	1	1	1
53	PR3	AutoNLP	final	english	text	4	90000	30000	var	1	1	1
54	PR4	AutoNLP	final	chinese	text	11	100000	50000	var	1	1	1
55	PR5 Tal	AutoNLP **AutoDL**	final final	chinese	text	15	250000	132688	var	1	1	1
56	Adult	**AutoDL**	public	categorical	tabular	5	39074	9768	1	1	24	1
57	Dilbert	**AutoDL**	public	objects	tabular	5	14860	9720	1	1	2000	1
58	Digits	**AutoDL**	public	HWR	tabular	10	35000	35000	1	1	1568	1
59	Madeline	**AutoDL**	public	artificial	tabular	2	4220	3240	1	1	259	1
60	Barak	**AutoDL**	feedback	CE pair	tabular	4	21869	2430	1	1	270	1
61	Bilal	**AutoDL**	final	audio	tabular	20	10931	2733	1	1	400	1

- IMAGE
- VIDEO
- SPEECH
- TEXT
- TABULAR
- Multi-label tasks

Liu Z, Xu Z, Rajaa S, Madadi M. Towards Automated Deep Learning: Analysis of the AutoDL challenge series 2019. To appear in *NeurIPSCD2019* in Proceedings of Machine Learning Research (PMLR) 2019:10.

(1) Raw data from **5 modalities**: Image, Video, Speech, Text, Tabular.

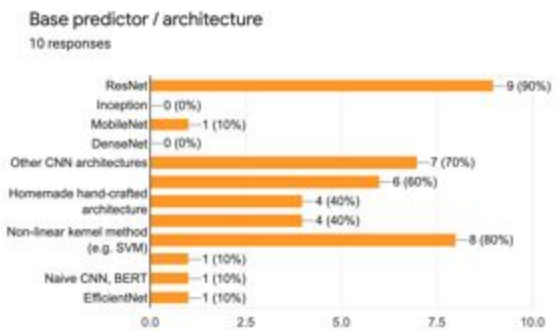
(2) Fixed time budget. Any-time learning (ALC metric). **Blind testing**.

(3) Starting kit, sample “public” data and **baselines** provided.

(4) Fixed computational resources.

(5) Using **Deep Learning** was **NOT imposed.**

Neural architectures used in the winning approaches



Architecture name	# Parameters	Domains	Teams
ResNet-18, ResNet-9 (He et al 2015)	11.4M, 5.7M	image, video	Kakaobrain, DeepWisdom, automl_freiburg
MC3 (Du Tran et al CVPR 2018)	**32.8M**	video	DeepWisdom
EfficientNet-(b0, b1, b2) (M. Tan and Q. Le. 2019)	5.3M, 7.8M, 9.2M	image, video	DeepWisdom, automl_freiburg
MobileNetV2 (M. Sandler et al 2019)	3.4M	image, video	team_zhaw, DeepBlueAI
TextCNN	variable	text	Upwind_flys, DeepWisdom
Fast RCNN (Ross Girshick)		text	DeepWisdom
LSTM, BiLSTM (Hochreiter, Schmidhuber 1997)	0.2M-1M	text, speech	frozenmad, PASA_NJU
GRU, BiGRU, (Kyunghyun Cho et al 2014) GRU with Attention	0.1M-1M	text, speech	DeepBlueAI, DeepWisdom
BERT-like (Tiny-BERT(X.Jiao et al))	<110M	text	frozenmad, upwind_flys
DNN	<1M	tabular	DeepWisdom

AutoML techniques vs domains

Approach	image	video	speech	text	tabular
Meta-learning	Offline meta-training transferred with AutoFolio [25] based on meta-features (*automl freiburg*) Offline meta-training generating solution agents, searching for optimal sub-operators in predefined sub-spaces, based on dataset meta-data. (*DeepWisdom*) MAML-like method [17] (*team zhaw*)				
Preprocessing	image cropping and data augmentation (*PASANJU*), fast autoaugment (*DeepBlueAI*)	Sub-sampling keeping 1/6 frames and adaptive image size (*DeepBlueAI*) Adaptive image size	MFCC, Mel Spectrogram, STFT	root features extractions with stemmer, meaningless words filtering (*DeepBlueAI*)	Numerical and Categorical data detection and encoding
Hyperparameter Optimization	Offline with BOHB [26] (Bayesian Optimization and Multi-armed Bandit) (*automl freiburg*) Sequential Model-Based Optimization for General Algorithm Configuration (SMAC) (*automl freiburg*)				Baysien Optimization (*PASANJU*) HyperOpt [27] (*Inspur AutoDL*)
Transfer learning	Pre-trained on ImageNet [28] (all teams except *Kon*)	Pre-trained on ImageNet [28] (all top-8 teams except *Kon*) MC3 model pretrained on Kinetics (*DeepWisdom*)	ThinResnet34 pre-trained on VoxCeleb2 (*DeepWisdom*)	FastText pre-trained on Common Crawl (*frozenmad*)	
Ensemble learning	Adaptive Ensemble Learning (ensemble latest 2 to 5 predictions) (*DeepBlueAI*)	Ensemble Selection [29] (top 5 validation predictions are fused) (*DeepBlueAI*); Ensemble models sampling 3, 10, 12 frames (*DeepBlueA*)	last best predictions ensemble strategy (*DeepWisdom*) averaging 5 best overall and best of each model: LR, CNN, CNN+GRU (*DeepBlueA*)	Weighted Ensemble over 20 best models [29] (*DeepWisdom*)	LightGBM ensemble with bagging method [30] (*DeepBlueAI*), Stacking and blending (*DeepWisdom*)

Teams vs domains

Team	image	video	speech	text	tabular
DeepWisdom	[**ResNet-18** and ResNet-9 models] [pretrained on ImageNet]	[MC3 model] [pretrained on Kinetics]	[fewshot learning] [LR, ThinRestnet34 models] [pretrained on VoxCeleb2]	[fewshot learning] [task difficulty and similarity evaluation for model selection] [SVM, **TextCNN**,[fewshot learning] RCNN, GRU, GRU with Attention]	[**LightGBM**, Xgboost, Catboost, DNN models] [no pretrained]
DeepBlueAI	[data augmentation with Fast AutoAugment] [**ResNet-18** model]	[subsampling keeping 1/6 frames] [Fusion of 2 best models]	[iterative data loader (7, 28, 66, 90%)] [MFCC and Mel Spectrogram preprocessing] [LR, CNN, CNN+GRU models]	[Samples truncation and meaningless words filtering] [Fasttext, **TextCNN**, BiGRU models] [Ensemble with restrictive linear model]	[3 **LightGBM** models] [Ensemble with Bagging]
PASA NJU	**ResNet-18** and SeResnext50; preprocessing: shape standardization and image flip (data augmentation)	**ResNet-18** and SeResnext50; preprocessing: shape standardization and image flip (data augmentation)	[data truncation(2.5s to 22.5s)][LSTM, **VggVox** ResNet with pretrained weights of DeepWis- dom(AutoSpeech2019) ThinRestnet34?]	[data truncation(300 to 1600 words)][TF-IDF and word embedding]	[iterative data loading] [Non Neural Nets models] [models complexity increasing over time] [Baysien Optimization of hyperparameters]
frozenmad	[images resized under 128x128] [progressive data loading increasing over time and epochs] [**ResNet-18** model] [pretrained on ImageNet]	[Successive frames difference as input of the model] [pretrained **ResNet-18** with RNN models]	[progressive data loading in 3 steps 0.01, 0.4, 0.7] [time length adjustment with repeating and clipping] [STFT and Mel Spectrogram preprocessing] [LR, LightGBM, **VggVox** models]	[TF-IDF and BERT tokenizers] [SVM, RandomForest , CNN, tinyBERT]	[progressive data loading] [no preprocessing] [Vanilla Decision Tree, RandomForest, Gradient Boosting models applied sequentially over time]

Lessons learned from the AutoDL challenge

(1) The winning methods are capable of generalizing on new unseen datasets => Potential universal AutoML solutions

(2) Domain-dependent approaches are dominant
=> No universal workflows, mostly hand-tuned meta-learning

(3) We cannot afford to run expensive NAS for every new task
=> Need transferability of learned architectures

(4) Beating Baseline 3 by using “true” meta-learning is hard
=> Need more meta-train datasets (public datasets)

MetaDL challenge

	Input	Output	Comp. Ex.
Alpha level: predict() in sklearn, a **classifier**	x example/sample (e.g. an image)	y labels	**Code Jam LeetCode**
Beta level: fit() in sklearn, a **learning algo.**	T ML task (dataset)	α alpha-level algo	**(Auto)ML challenges & AutoDL**
Gamma level: meta_fit() on a **meta-dataset**	$\mathfrak{D}$ Meta-dataset	β beta-level algo	**MetaDL**

Check and stay tuned https://metalearning.chalearn.org/

Conclusion

Take-home messages

AutoML problem can be formulated in 3 levels:
$\alpha \leftarrow \beta \leftarrow \gamma$

Domain specific AutoML solution generalizes

Hand-crafted gamma-level learning
=> Cross-domain meta-learning yet to be studied

Any-time learning aspect to be studied further

Stay tuned! autodl.chalearn.org