MATHÉMATIQUES VISION APPRENTISSAGE

université PARIS-SACLAY

Automated Deep Learning

LIU Zhengying

Laboratoire en Recherche Informatique (LRI)

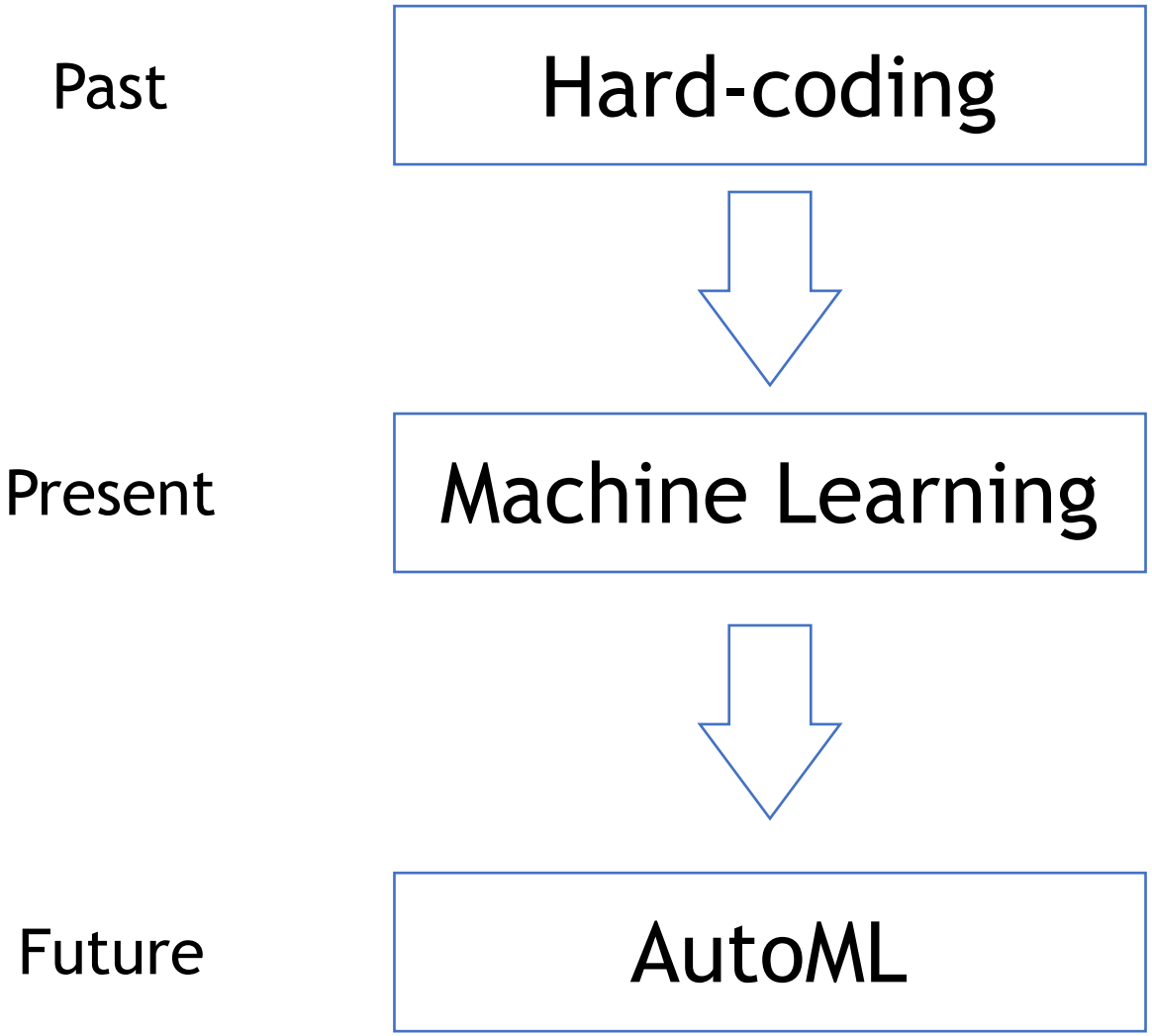
U. Paris-Sud / U. Paris Saclay / Inria Saclay

18 Mar 2019

Contents

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

AutoML: an intro


Past
Hard-coding
Present
Machine Learning
Future
AutoML

Hard-coding

Hard-coded algorithm:
"if..else" rules, HOG, SIFT, etc

$$X \xrightarrow{f_\theta} Y$$

$\mapsto$ "dog"

$\mapsto$ "cat"

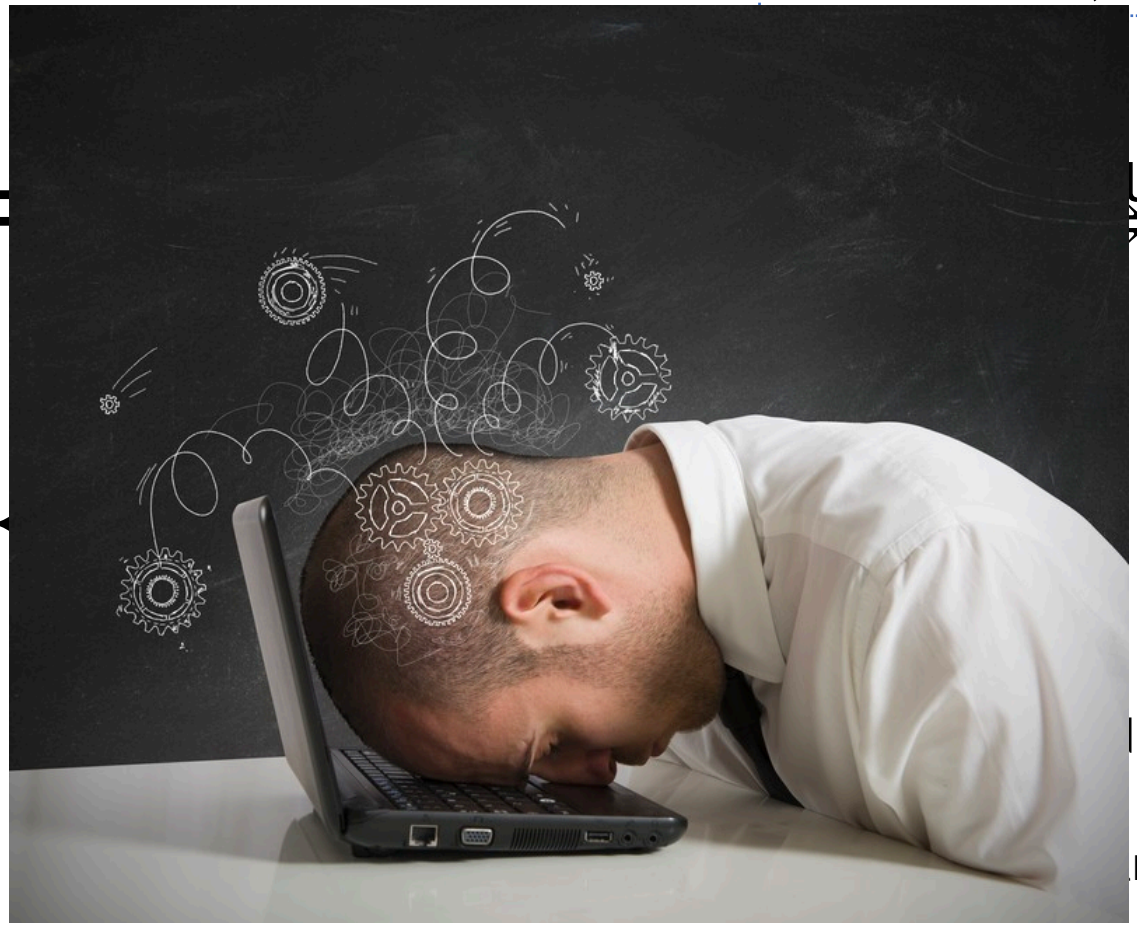
encoded by: parameters $\theta \in \Theta$

hand-crafted by engineers

Machine Learning

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

D =



lid

$P(f_\theta)$

performance
(e.g. accuracy)

Dogs vs Cats
dataset

=

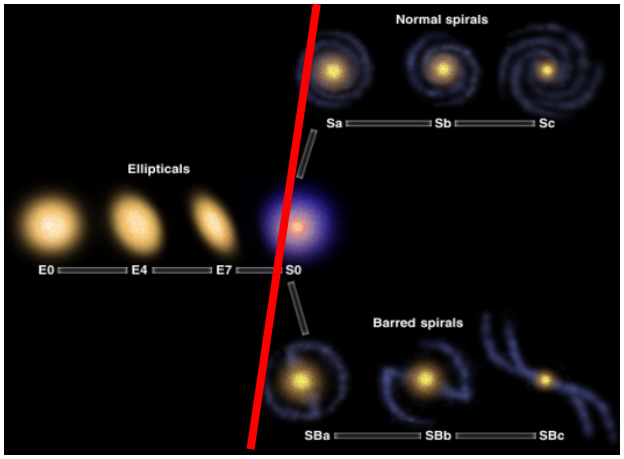
N (for another A)

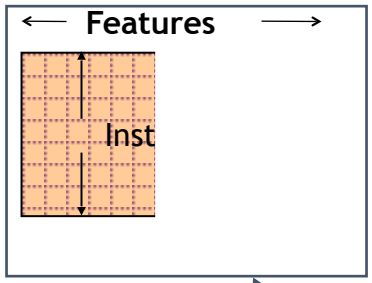
nother A)

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

hand-crafted by ML experts

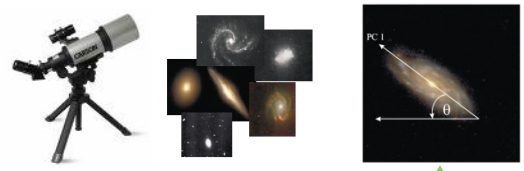
The typical process…





$$f(\mathbf{x};\theta_{f_1})$$
$$\dots$$
$$f(\mathbf{x};\theta_{f_z})$$

$$f(\mathbf{x};\theta_f)\quad g(\mathbf{x};\theta_g)\quad h(\mathbf{x};\theta_h)\quad l(\mathbf{x})$$



Feature selection

Classification method

Data

Data Preprocessing

Model evaluation

Training model

$f(\theta_f)$

ERROR | $f(\theta)$] ?

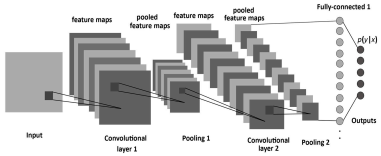
Machine Learning

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

$$D = \{X_i, Y_i\} \xrightarrow{A_\lambda} f_\theta \overset{D_{\text{valid}}}{\Longrightarrow} P(f_\theta)$$

(or $p_\theta(y \mid x)$)

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 $\lambda \in \Lambda$

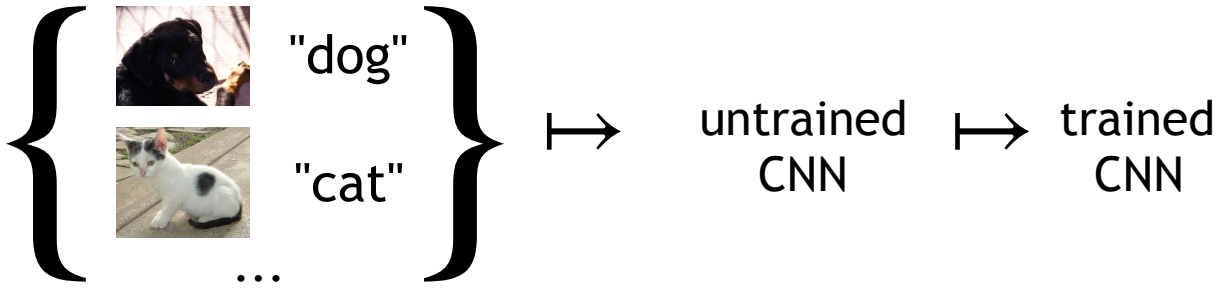
hand-crafted by ML experts

AutoML
(Hyperparameter Optimization)

HPO algorithm:
SMAC (Auto-sklearn), Neural Architecture Search, MCTS, etc

$$D = \{X_i, Y_i\} \longmapsto \mathfrak{A} \longrightarrow A_\lambda \overset{D_{\text{train}}}{\Rightarrow} f_\theta \overset{D_{\text{valid}}}{\Rightarrow} P$$

$$(D = D_{\text{train}} \amalg D_{\text{valid}})$$



zero hyperparameters

no hand-crafting at all!

can be learned too $\Longrightarrow$ meta-learning

AutoML
(Meta-learning)

Meta-learning algorithm: Auto-sklearn, SVM applied to meta-features, etc

$$\mathfrak{D} = \{D_j, A_{\lambda_j}, P_j\} \xrightarrow{\mathscr{A}} \mathfrak{A}$$

can be considered as reward/fitness

$$\left\{ \begin{array}{c} \text{Dogs vs Cats}, CNN_{\lambda_1}, 0.67 \\ \text{Iris}, DT_{\lambda_2}, 0.78 \\ \text{CIFAR-10}, SVM_{\lambda_3}, 0.55 \\ \text{MNIST}, SVM_{\lambda_4}, 0.88 \\ \text{CIFAR-10}, CNN_{\lambda_5}, 0.80 \\ \ldots \end{array} \right\} \mapsto \text{Learned HPO algorithm}$$

can be combined with Hyperparameter Optimization

we can also have $\mathfrak{D} = \{D_j, A_{\lambda_j}, f_{\theta_j}, P_j\}$

with trained $f_{\theta_j} \Longrightarrow$ transfer learning

Overview

	Schema	Problem
Hard-coding	$X \xrightarrow{f} Y$	$\max_f P(f; D_{test})$
Machine Learning	$D_{train} \xrightarrow{A} f$	$\max_A P(\hat{f}; D_{test})$ where $\hat{f} = A(D_{train})$
AutoML	$\mathfrak{D}_{train} \xrightarrow{\mathscr{A}} \mathfrak{A}$	$\max_{\mathscr{A}} \sum_{D_{tr}, D_{te} \in \mathfrak{D}_{test}} P(\hat{f}; D_{te})$ where $\hat{f} = \hat{A}(D_{tr})$ with $\hat{A} = \hat{\mathfrak{A}}(D_{tr}), \hat{\mathfrak{A}} = \mathscr{A}(\mathfrak{D}_{train})$

Test score: $P(f; D_{test}) = \frac{1}{|D_{test}|} \sum_{X,Y \in D_{test}} S(f(X), Y))$

unknown true test labels at training time

use estimation: cross-validation, hold-out validation, etc

The AutoML problem: definition

$$\max_{\mathcal{A}} \sum_{\substack{D_{tr}, D_{te} \\ \in \mathfrak{D}_{test}}} P(\hat{f}; D_{te}) \qquad \text{where } \hat{f} = \hat{A}(D_{tr}) \text{ with } \hat{A} = \hat{\mathfrak{A}}(D_{tr}), \hat{\mathfrak{A}} = \mathcal{A}(\mathfrak{D}_{train})$$

supervised learning

reinforcement learning

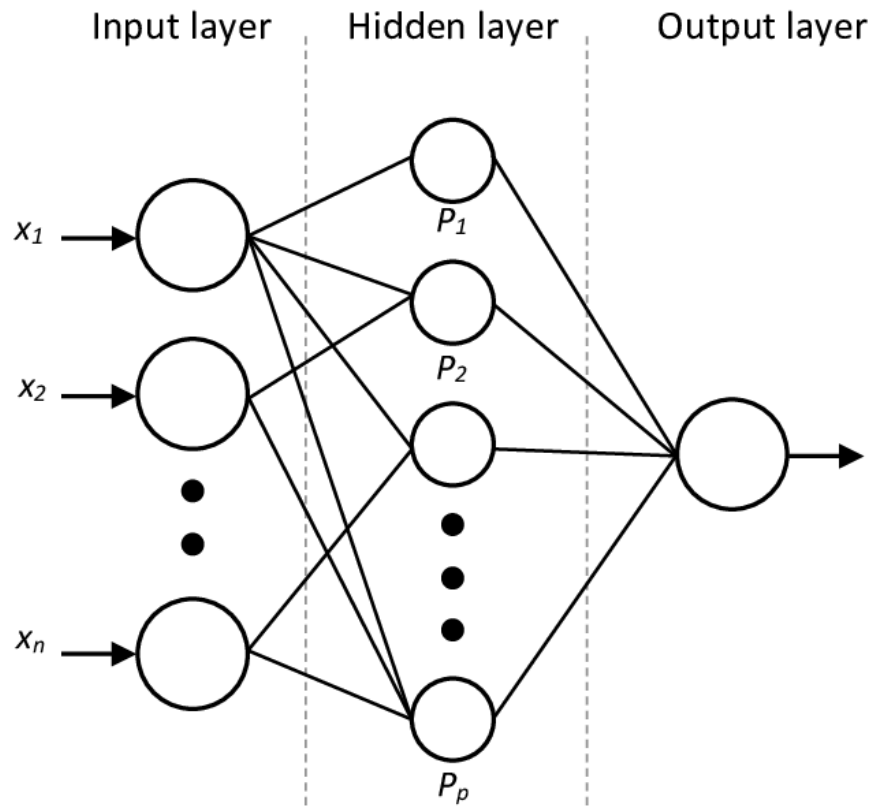
learning to learn ⇐ two layers of learning

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

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

$$(D_{tr}, A_{\lambda_1}, f_{\theta_1}, P_1), (D_{tr}, A_{\lambda_2}, f_{\theta_2}, P_2), (D_{tr}, A_{\lambda_3}, f_{\theta_3}, P_3), \cdots$$

Exercise: a toy example



X

$Y = f(X) \in \{0,1\}$

neural network with one hidden layer

θ? λ? hard-coding approach? ML? AutoML?

AutoML: what's exciting?

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

→ AI for everyone

AutoML: already a hot topic

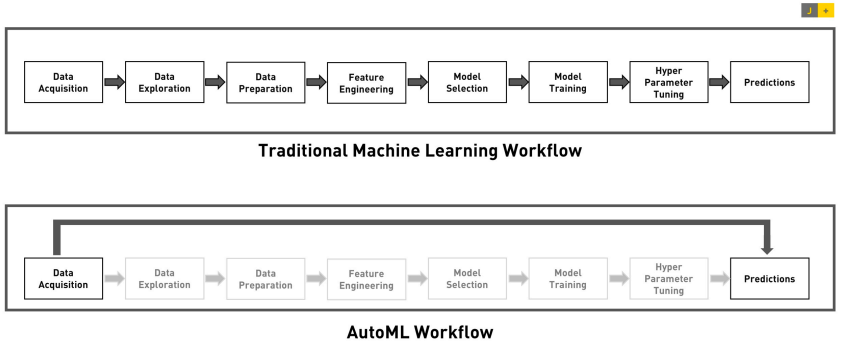
Google's AutoML

AutoML.org

Lifelong AutoML

4Paradigm

AutoML



AUTO KERAS

Auto-Sklearn

AutoML methods

with application to Deep Learning

We'll focus on the simplest case

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

⇒ Hyperparameter Optimization

⇒ single fixed training dataset: D_{tr}

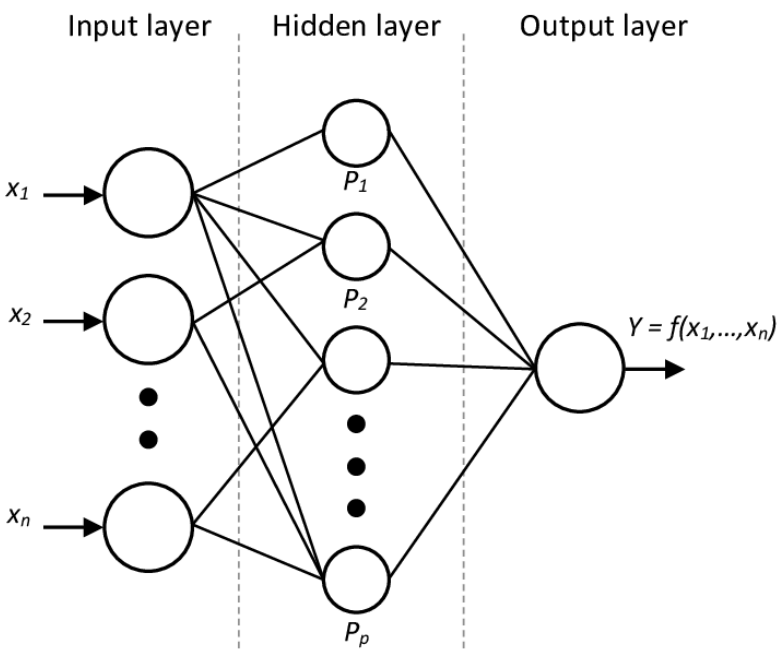
⇒ we only need to focus on $A_\lambda, \lambda \in \Lambda$

Reminder:

$$\max_{\mathscr{A}} \sum_{\substack{D_{tr}, D_{te} \\ \in \mathfrak{D}_{test}}} P(\hat{f}; D_{te}) \qquad \text{where } \hat{f} = \hat{A}(D_{tr}) \text{ with } \hat{A} = \mathfrak{A}(D_{tr}), \mathfrak{A} = \mathscr{A}(\mathfrak{D}_{train})$$

Search Space

How do we describe (encode) a learning algorithm?



in <u>natural language</u>:
"a feed-forward neural network with one hidden layer of p=10 neurons, using ReLU as activation and Adam as optimizer, with learning rate lr=0.001, ..."

<u>formally</u>:

$$A_\lambda, \lambda \in \Lambda \; ??$$

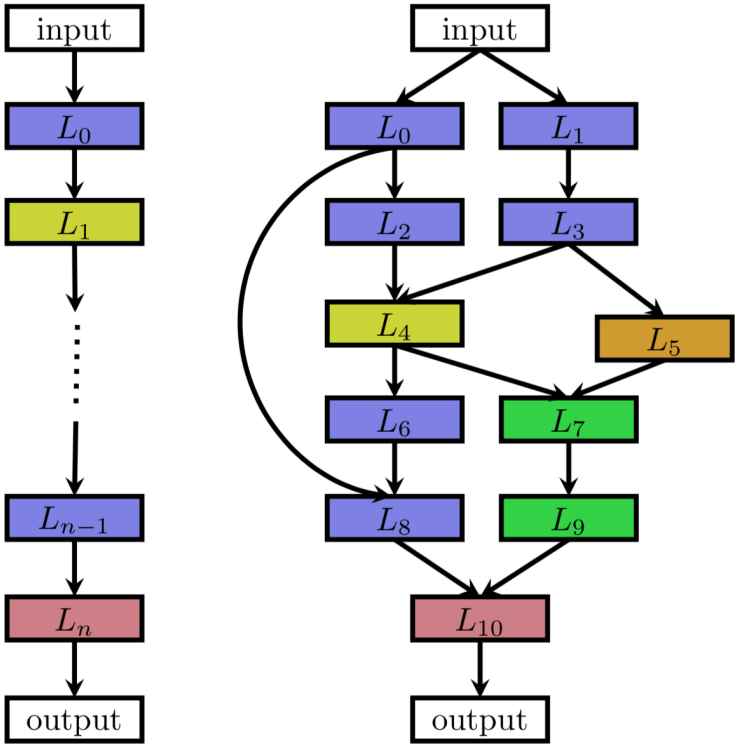
Search Space (for DL)

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

chain-structured (feed-forward)

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

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



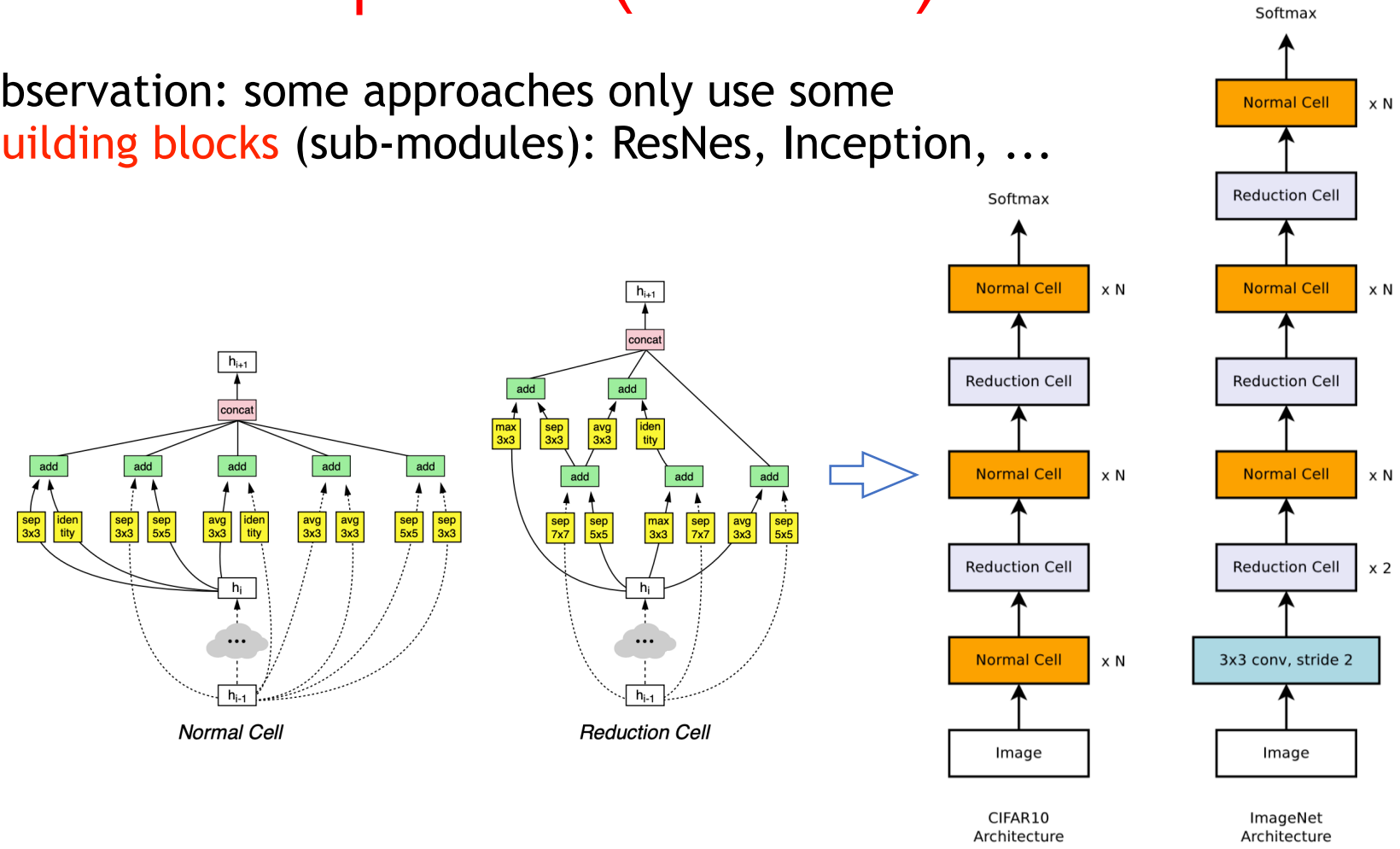
multi-branch

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

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*

Hyperparameter Optimization: a reformulation

an HPO algorithm aims to solve: $\max_{\lambda \in \Lambda} P(\hat{f}; D_{test})$ where $\hat{f} = A_\lambda(D_{train})$

unknown test score: $P(\hat{f}; D_{test})$ ⇨ 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

$$\hat{P} : \Lambda \to \mathbb{R}$$
$$\lambda \mapsto s = \hat{P}(A_\lambda) = \hat{P}(\hat{f})$$

is an estimation of the test score

Remark: some approaches optimize λ and θ at the same time ⇨ bi-level optimization (not discussed)

Search Strategy

- Heuristic search
  - Grid Search
  - Random Search
  - Evolutionary Algorithms
- Bayesian Optimization
- Reinforcement Learning methods

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

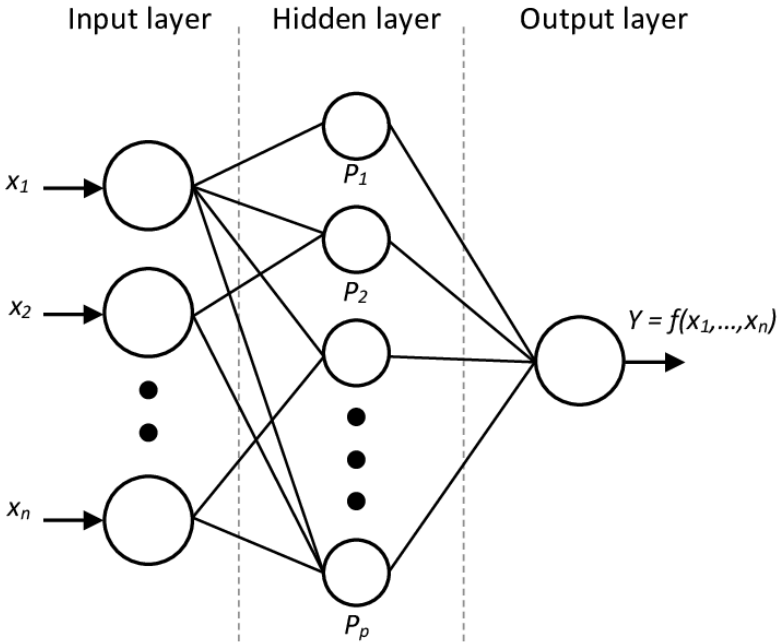
learning rate

try every possible combination in

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

evaluate it and return argmax in the end

curse of dimensionality!



Random Search

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

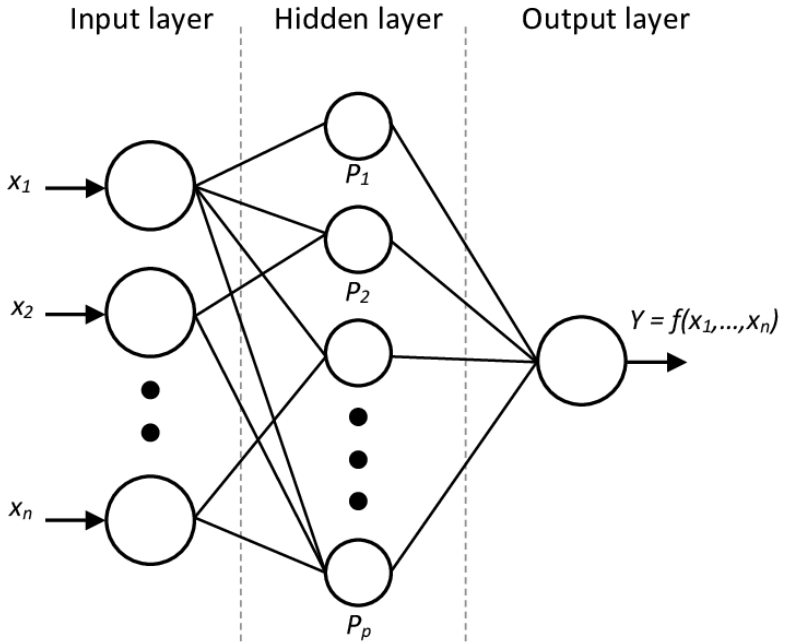
neurons in hidden layer

learning rate

Randomly sample certain number of combinations in

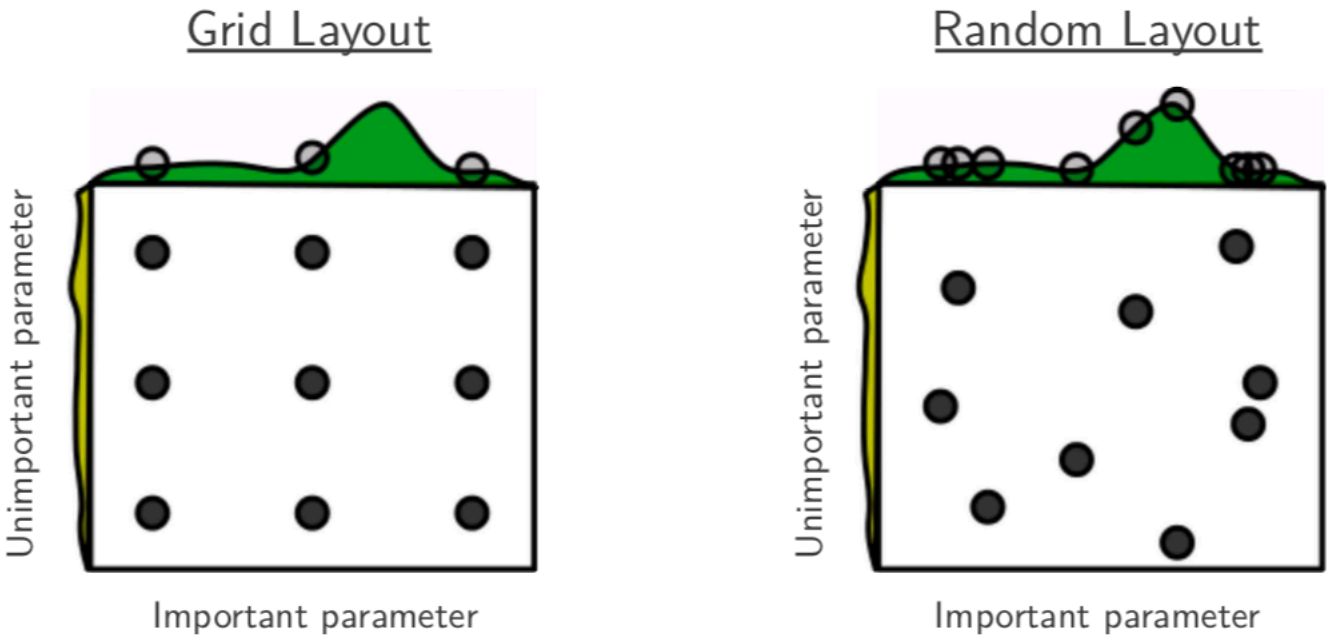
$$\Lambda = \Lambda_1 \times \Lambda_2$$

evaluate it and return argmax in the end



Grid Search and Random Search

two model-free black-box optimization methods



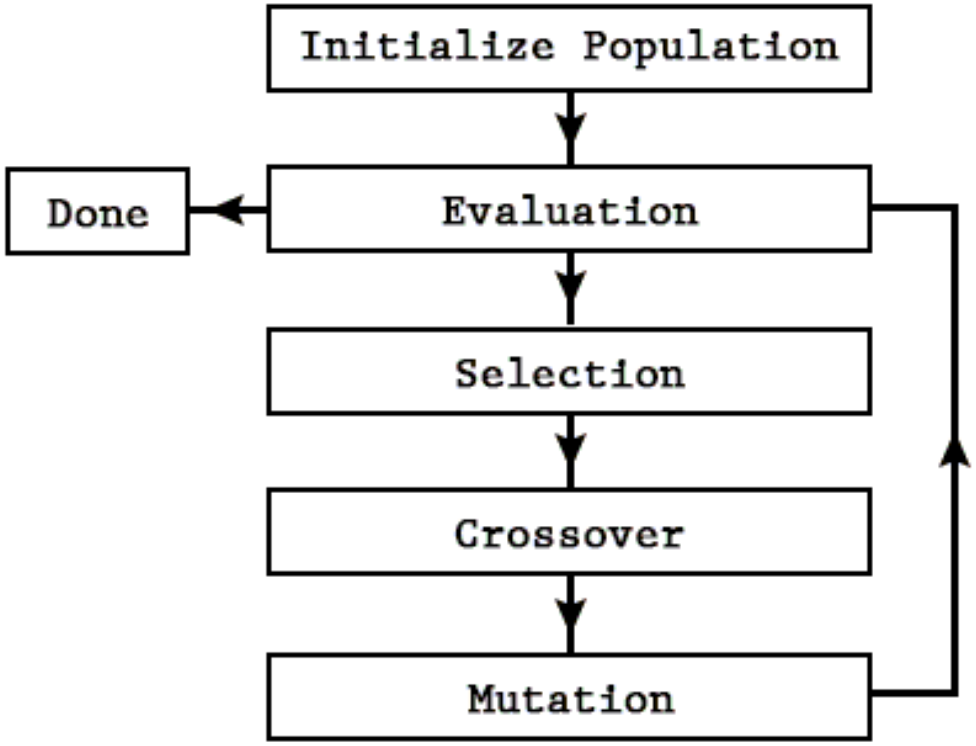
RS tends to perform better than GS when some HP are more important than others

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)

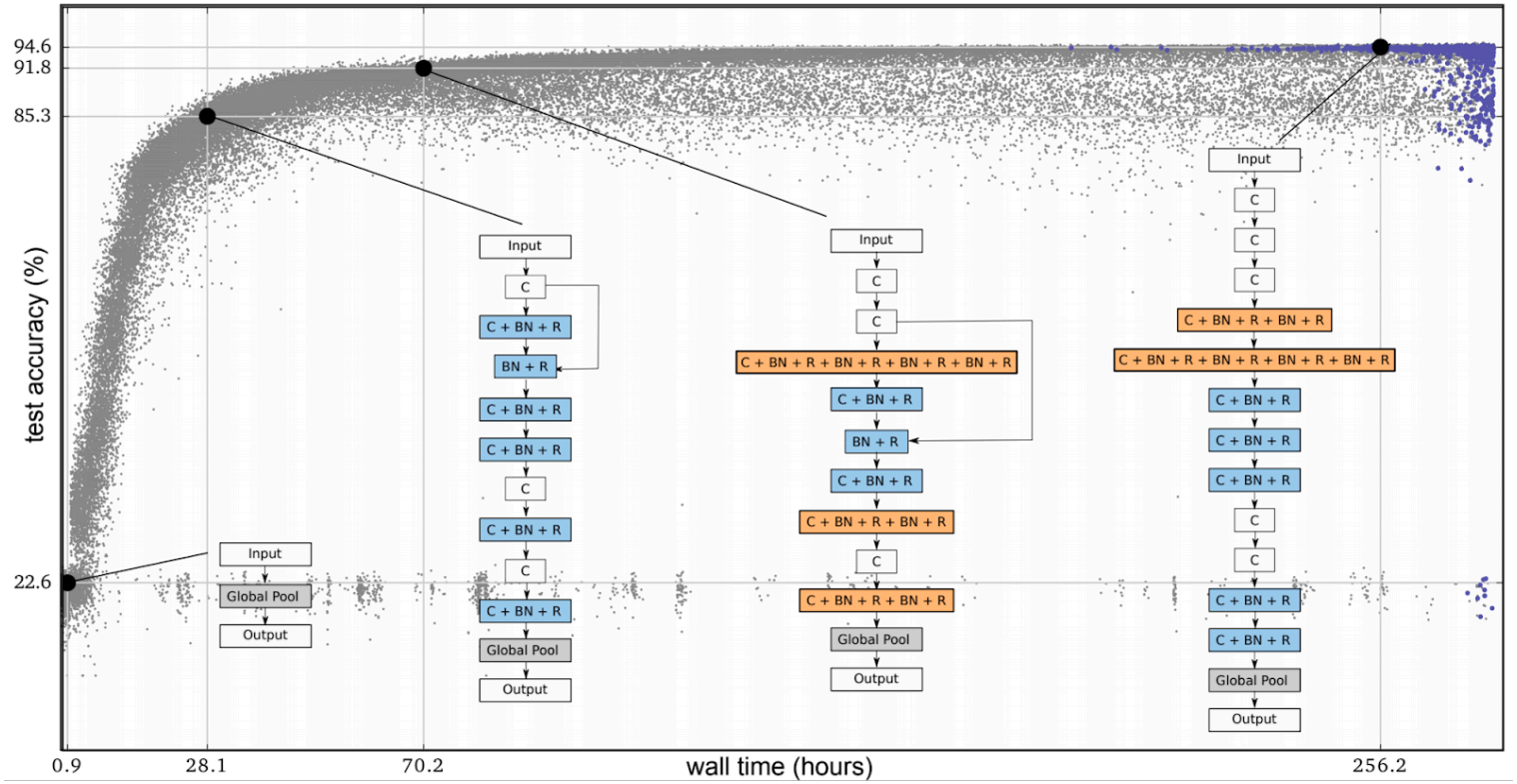
chromosome (DNA): tensor graph

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

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

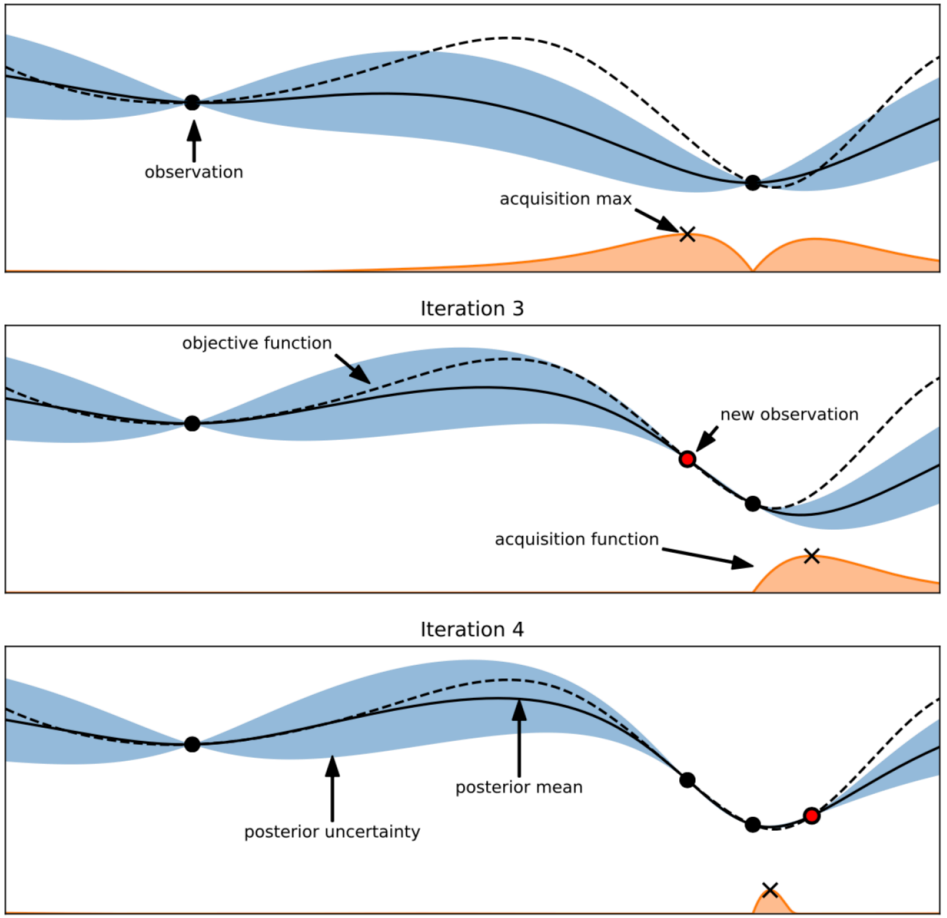
Original idea:

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

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

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

repeat until convergence



Bayesian Optimization (cont'd)

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

usual acquisition function:
Expected Improvement (EI)

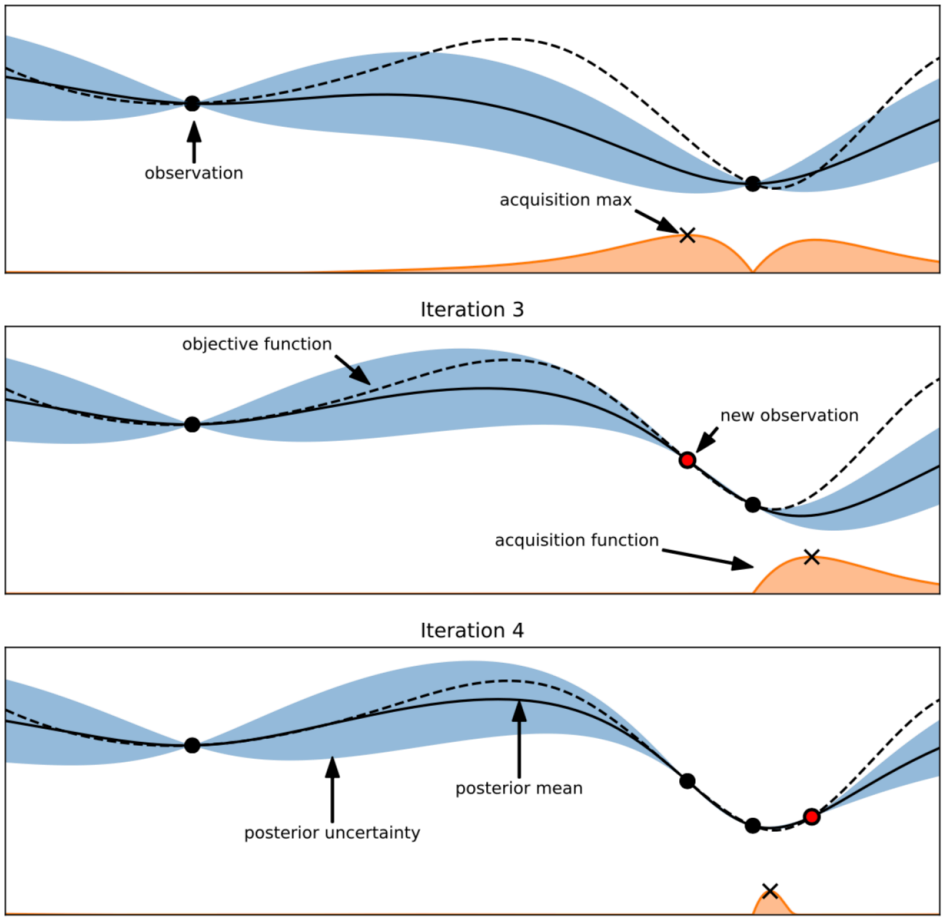
$$a_{EI}(\lambda \mid D_n) = \mathbb{E}[\max(\hat{P}(\lambda) - s_{\max}, 0)]$$

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



Bayesian Optimization: an example

Bergstra JS, Bardenet R, Bengio Y, Kégl B. **Algorithms for Hyper-Parameter Optimization**. *NIPS2011*

Tree Parzen Estimator (TPE) -> Hyperopt

model $p(\lambda \mid s < \alpha)$ and $p(\lambda \mid s > \alpha)$ instead of $p(s \mid \lambda)$

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

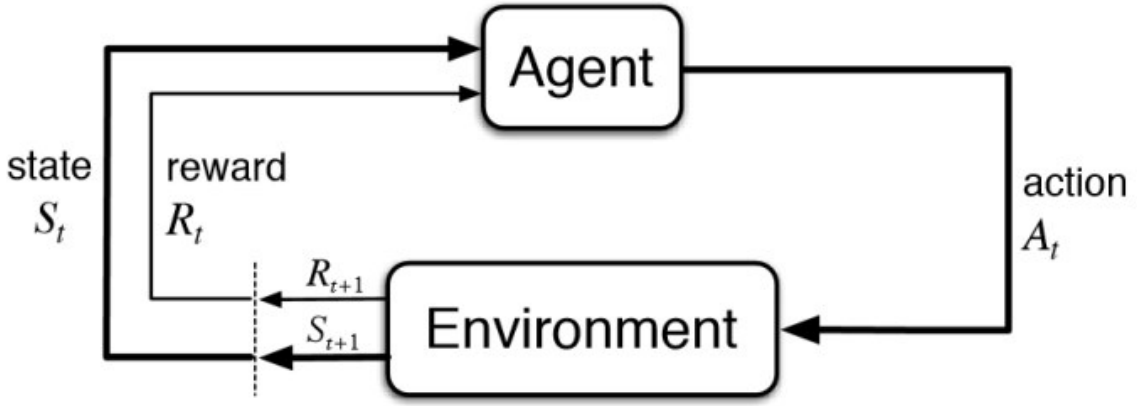
$$p(x|y) = \begin{cases} \ell(x) & \text{if } y < y^* \\ g(x) & \text{if } y \geq y^*, \end{cases}$$

$$\mathrm{EI}_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y) p(y|x) dy = \int_{-\infty}^{y^*} (y^* - y) \frac{p(x|y)p(y)}{p(x)} dy$$

$$EI_{y^*}(x) = \frac{\gamma y^* \ell(x) - \ell(x) \int_{-\infty}^{y^*} p(y) dy}{\gamma \ell(x) + (1-\gamma) g(x)} \propto \left(\gamma + \frac{g(x)}{\ell(x)}(1-\gamma)\right)^{-1}$$

Reinforcement Learning

A reminder:



State space: S

Action space: A

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

Reward: $\mathcal{R}^a_{ss'} : S \times A \times S \to \mathbb{R}$

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

that maximizes the (discounted) expected return

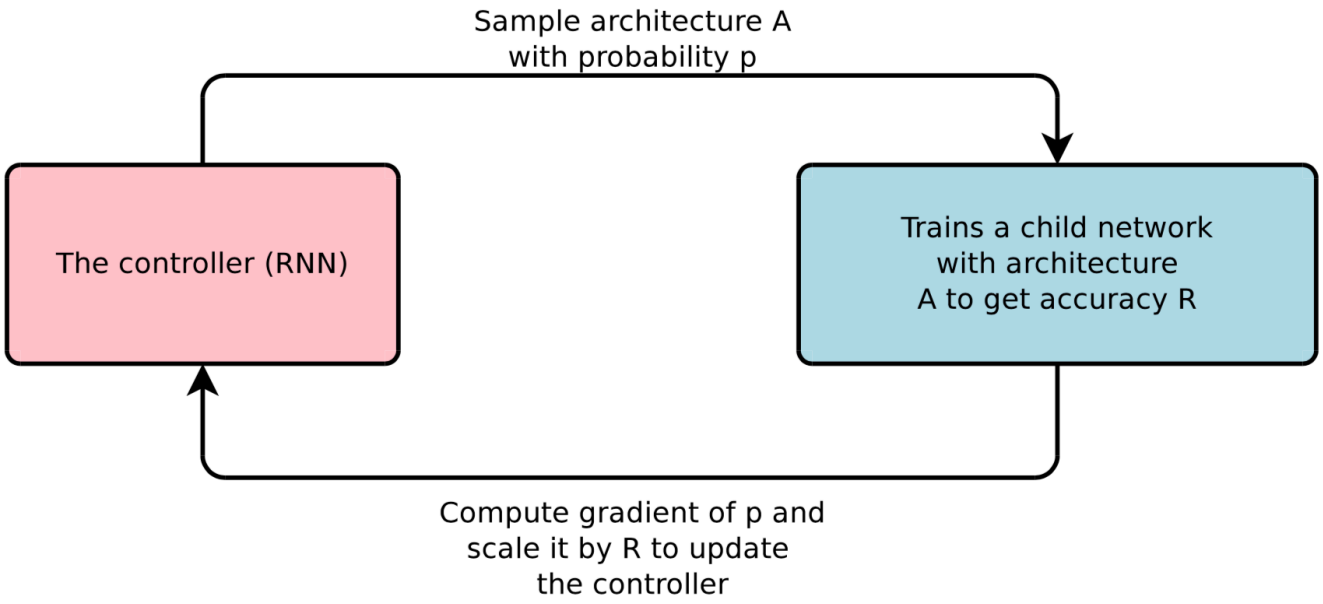
$$\mathbb{E}_\pi \left[\sum_{t=1}^{T} \gamma^t r_r \right]$$

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: $\nabla_{\theta_c} J(\theta_c) = \sum_{t=1}^{T} E_{P(a_{1:T};\theta_c)}\left[\nabla_{\theta_c} \log P(a_t|a_{(t-1):1};\theta_c) R\right]$

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

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

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!!!

Other AutoML methods than black-box optimization

Transfer Learning

Meta-learning

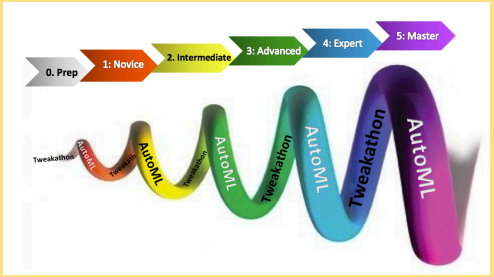
Ensemble methods
(competition winners)

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)

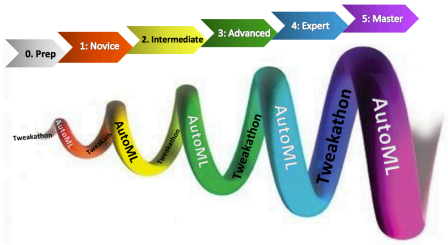
* Guyon I, Bennett K, Cawley G, et al. Design of the 2015 ChaLearn AutoML challenge. *IJCNN 2015*

AutoML challenges



Overview of AutoML challenges

	Year	Prize/teams	Winner
AutoML1	2015-2016 (1.5 year)	$30,000 600+ teams	AAD_Freiburg
AutoML2	2017-2018 (4 months)	$10,000 250+ teams	AAD_Freiburg
AutoML3	2018 (2.5 months)	$15,000 200+ teams	autodidact.ai
AutoDL	Coming soon!		



PAKDD 2018

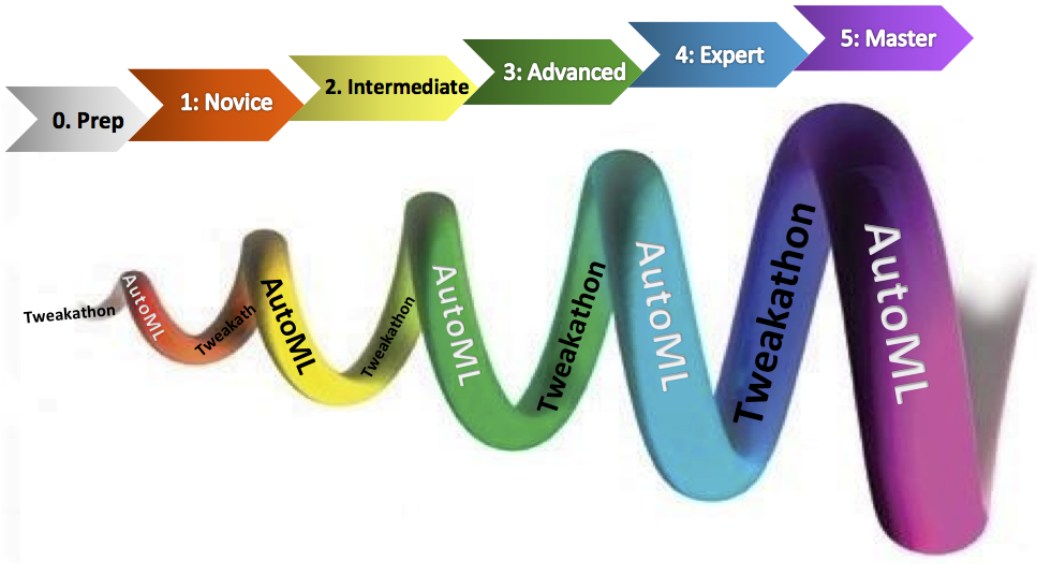
AUTOML Challenge 2018

4Paradigm CHALEARN

Lifelong AutoML

4Paradigm CHALEARN

AutoML1 - 5 rounds

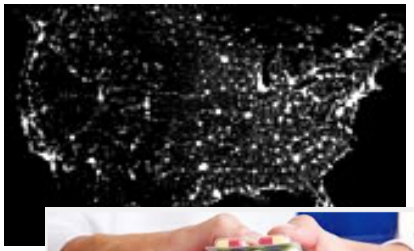


2 steps per round:

- AutoML
- Tweakathon

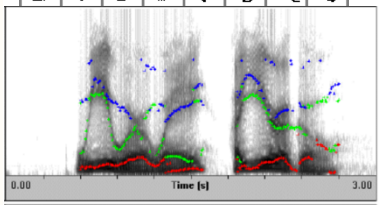
1. **NOVICE:** Binary classification.
2. **INTERMEDIATE:** Multiclass classification.
3. **ADVANCED:** Multiclass and multilabel.
4. **EXPERT:** Classification and regression.
5. **MASTER:** All of the above.

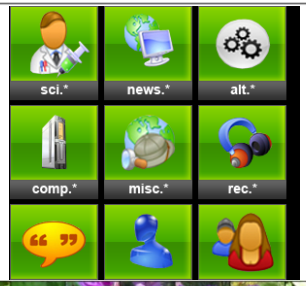
AutoML1 - Data: 30 large datasets

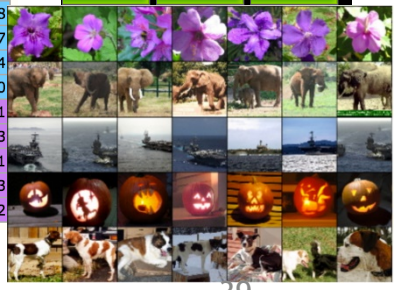


http://automl.chalearn.org/data

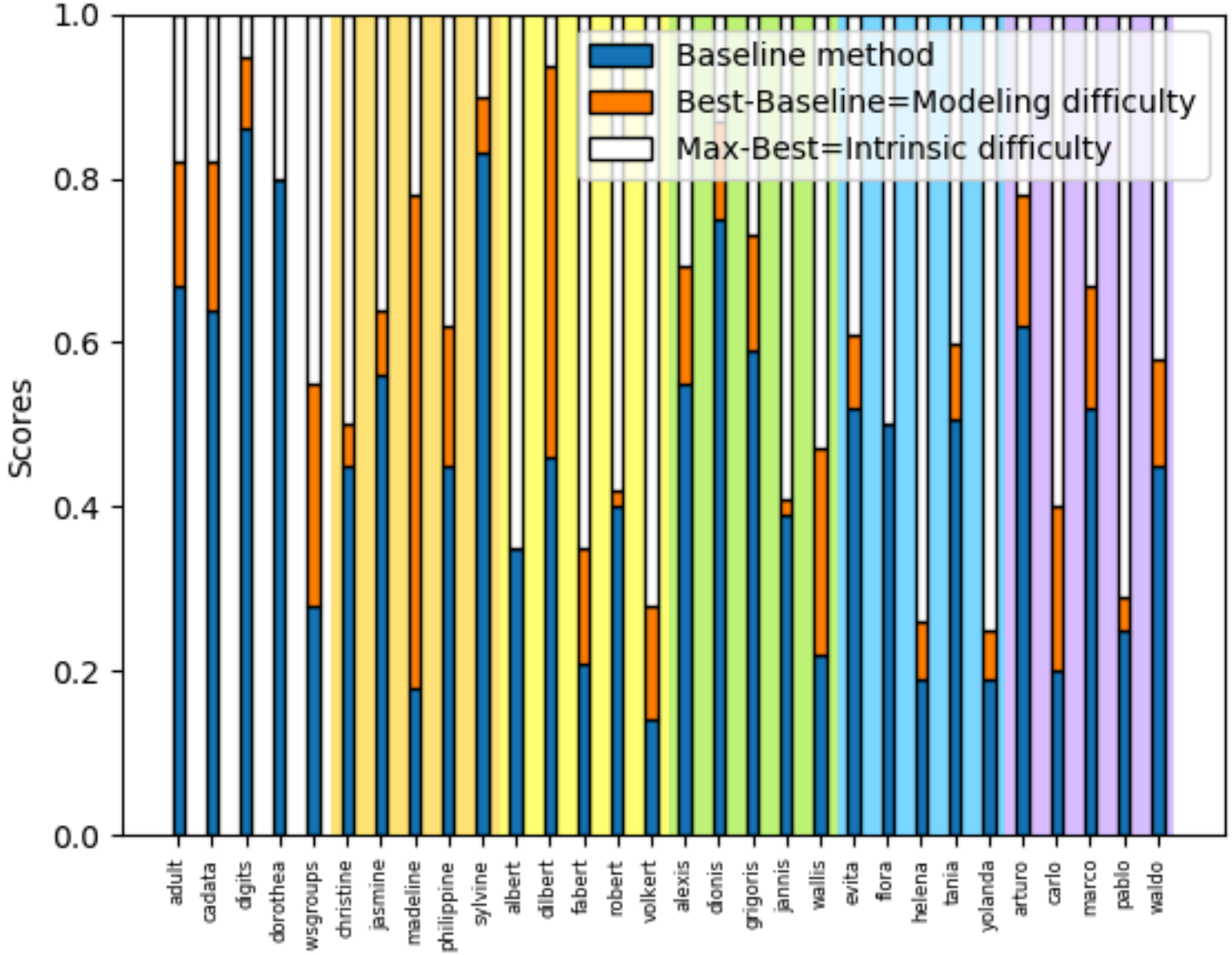
und	Num	Name	Task	Metric	Time	Cnum	Cbal	Sparse	Missng	Catvar	Irrvar	Pte	Pva	Ptr	N	Ptr/N
0	1	ADULT	multilabel	F1	300	3	1	0.16	0.011	1	0.5	9768	4884	34190	24	1,424.58
0	2	CADATA	regression	R2	200	0	NaN	0	0	0	0.5	10640	5000	5000	16	312.5
0	3	DIGITS	multiclass	BAC	300	10	1	0.42	0	0	0.5	35000	20000	15000	1568	9.57
0	4	DOROTHEA	binary	AUC	100	2	0.46	0.99	0	0	0.5	800	350	800	100000	0.01
0	5	NEWSGROUPS	multiclass	PAC	300	20	1	1	0	0	0	3755	1877	13142	61188	0.21
1	1	CHRISTINE	binary	BAC	1200	2	1	0.071	0	0	0.5	2084	834	5418	1636	3.31
1	2	JASMINE	binary	BAC	1200	2	1	0.78	0	0	0.5	1756	526	2984	144	20.72
1	3	MADELINE	binary	BAC	1200	2	1	1.2 E-06	0	0	0.92	3240	1080	3140	259	12.12
1	4	PHILIPPINE	binary	BAC	1200	2	1	0.0012	0	0	0.5	4664	1166	5832	308	18.94
1	5	SYLVINE	binary	BAC	1200	2	1	0.01	0	0	0.5	10244	5124	5124	20	256.2
2	1	ALBERT	binary	F1	1200	2	1	0.049	0.14	1	0.5	51048	25526	425240	78	5,451.79
2	2	DILBERT	multiclass	PAC	1200	5	1	0	0	0	0.16	9720	4860	10000	2000	5
2	3	FABERT	multiclass	PAC	1200	7	0.96	0.99	0	0	0.5	2354	1177	8237	800	10.3
2	4	ROBERT	multiclass	BAC	1200	10	1	0.01	0	0	0	5000	2000	10000	7200	1.39
2	5	VOLKERT	multiclass	PAC	1200	10	0.89	0.34	0	0	0	7000	3500	58310	180	323.94
3	1	ALEXIS	multilabel	AUC	1200	18	0.92	0.98	0	0	0	15569	7784	54491	5000	10.9
3	2	DIONIS	multiclass	BAC	1200	355	1	0.11	0	0	0	12000	6000	416188	60	6,936.47
3	3	GRIGORIS	multilabel	AUC	1200	91	0.87	1	0	0	0	9920	6486	45400	301561	0.15
3	4	JANNIS	multiclass	BAC	1200	4	0.8	7.3 E-05	0	0	0.5	9851	4926	83733	54	1,550.61
3	5	WALLIS	multiclass	AUC	1200	11	0.91	1	0	0	0	8196	4098	10000	193731	0.05
4	1	EVITA	binary	AUC	1200	2	0.21	0.91	0	0	0.46	14000	8000	20000	3000	6.67
4	2	FLORA	regression	ABS	1200	0	NaN	0.99	0	0	0.25	2000	2000	15000	200000	0.08
4	3	HELENA	multiclass	BAC	1200	100	0.9	6 E-05	0	0	0	18628	9314	65196	27	2,414.67
4	4	TANIA	multilabel	PAC	1200	95	0.79	1	0	0	0	44635	22514	157599	47236	3.34
4	5	YOLANDA	regression	R2	1200	0	NaN	1 E-07	0	0	0.1	30000	30000	400000	100	4000
5	1	ARTURO	multiclass	F1	1200	20	1	0.82	0	0	0.5	2733	1366	9565	400	23.91
5	2	CARLO	binary	PAC	1200	2	0.097	0.0027	0	0	0.5	10000	10000	50000	1070	46.73
5	3	MARCO	multilabel	AUC	1200	180	0.76	0.99	0	0	0	20482	20482	163860	15299	10.71
5	4	PABLO	regression	ABS	1200	0	NaN	0.11	0	0	0.5	23565	23565	188524	120	1,571.03
5	5	WALDO	multiclass	BAC	1200	4	1	0.029	0	1	0.5	2430	2430	19439	270	72







AutoML1 - Results



AutoML1 - Conclusion

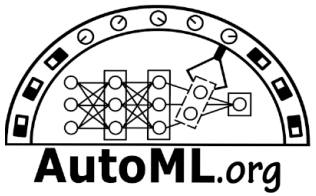
fixed time / fixed resource learning

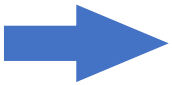
Deep Learning was far from winning

winner solution **Auto-Sklearn**:

meta-learning + BO + ensemble

everybody used ensemble methods

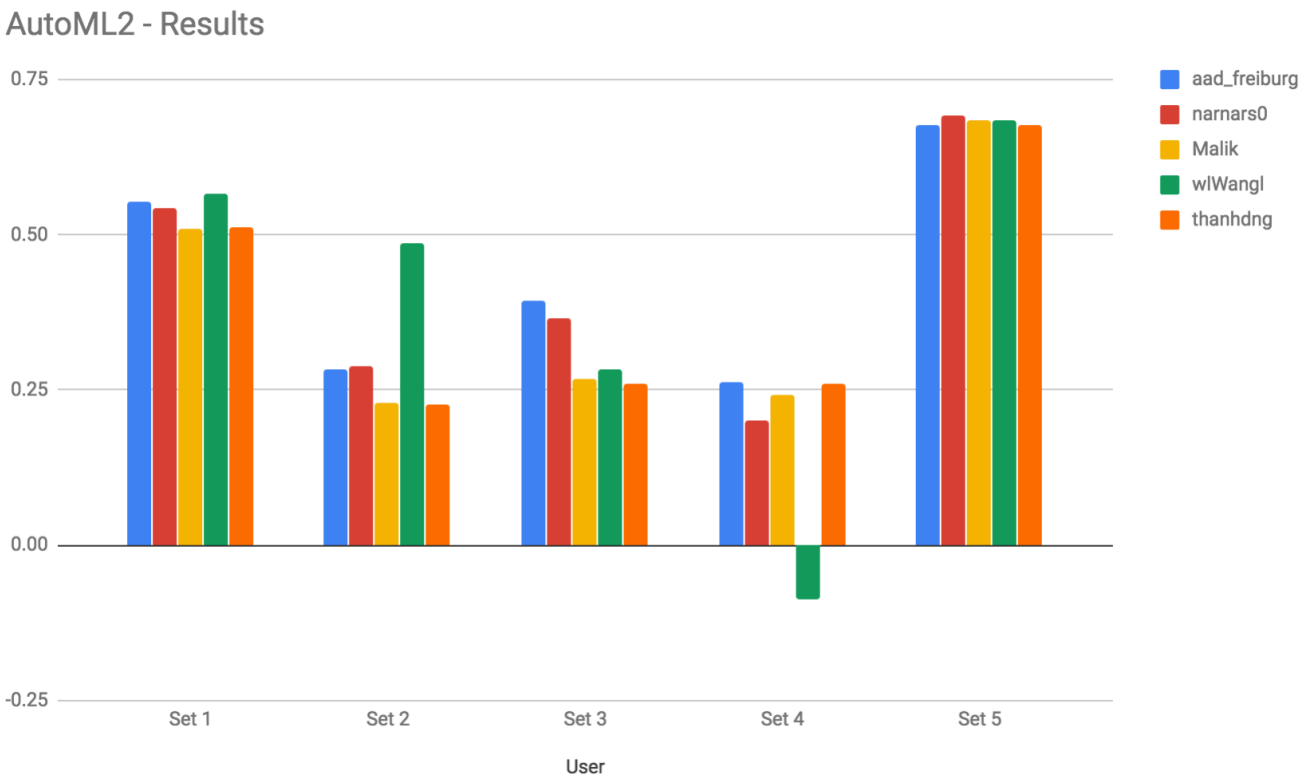




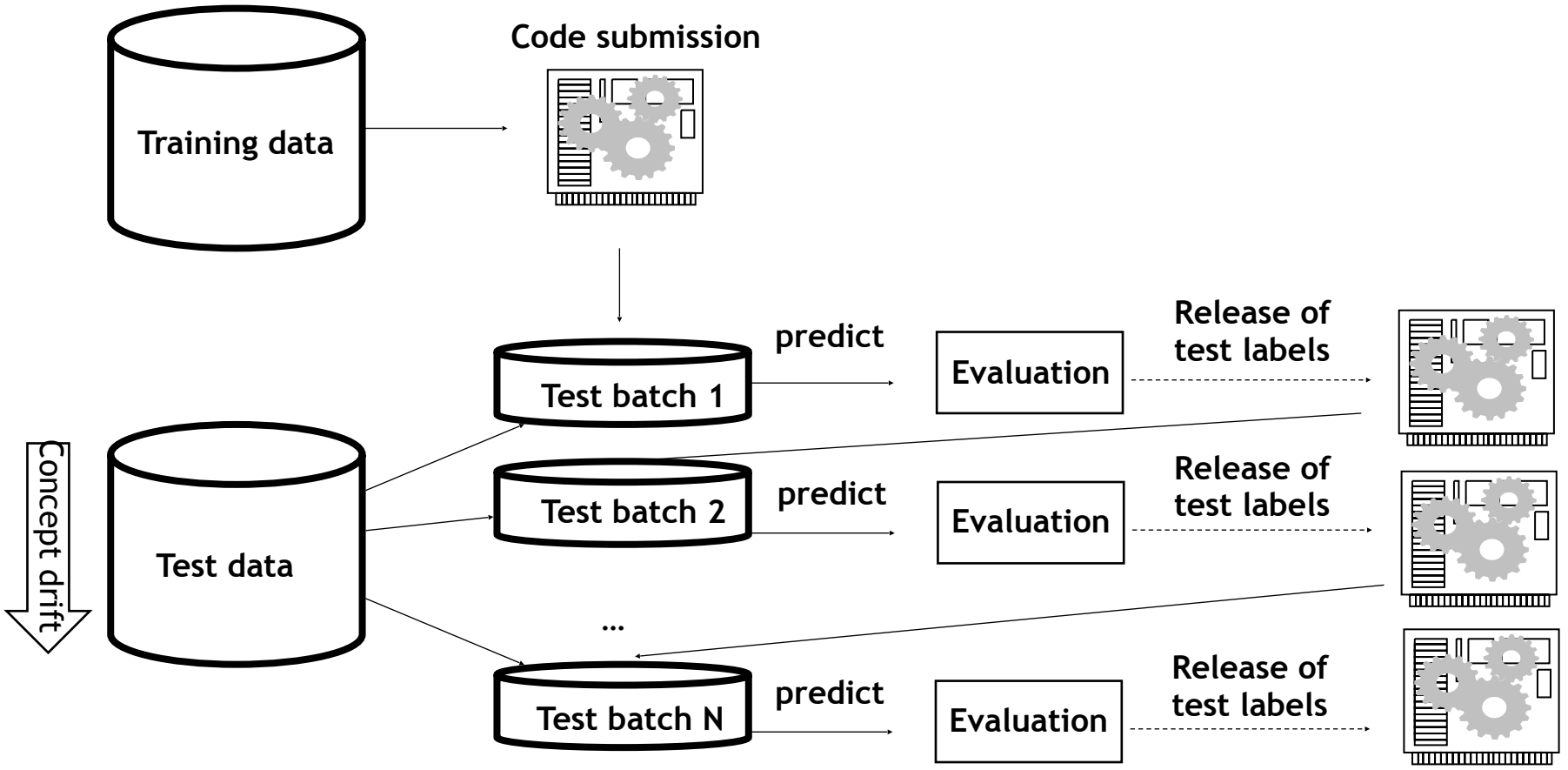
AutoML2, AutoML3, …

AutoML2

Similar settings to AutoML 1 but simplified



AutoML3 @ NIPS2018



AutoML2 / AutoML3 - Winners

In AutoML2, Auto-sklearn won again

all other methods are heavily influenced by auto-sklearn

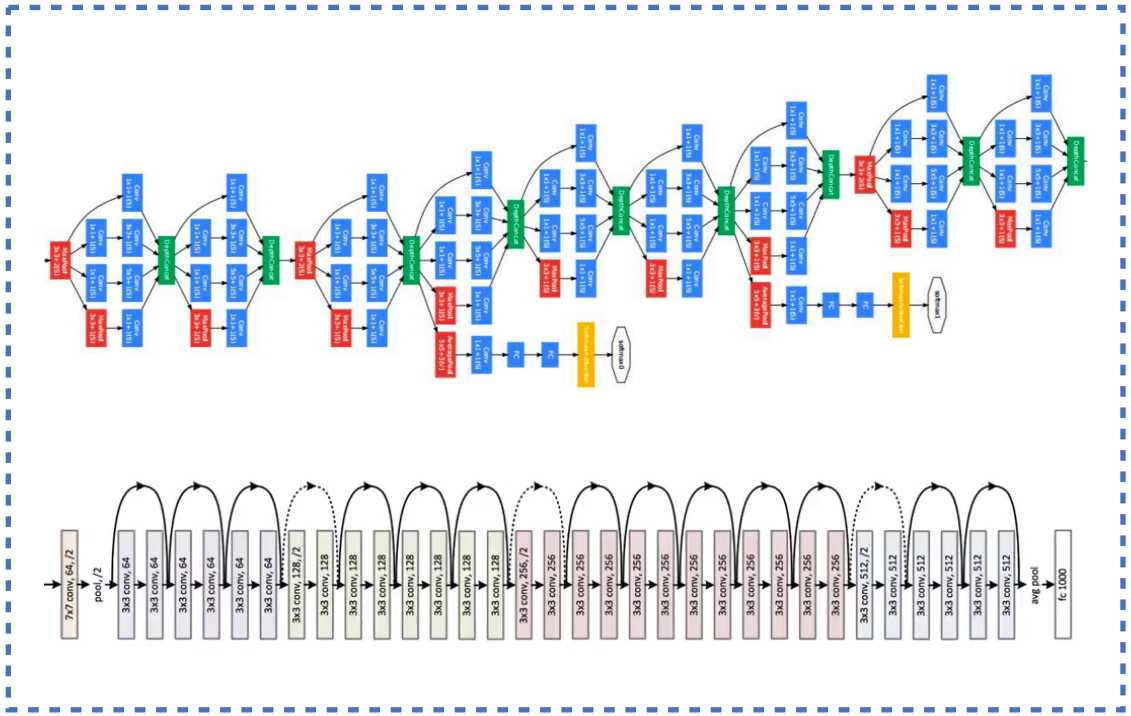
However, in AutoML 3, the winner was gradient tree boosting

Deep Learning methods never won in these challenge. This may be due to data type: feature-based

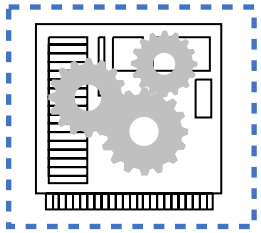
-> AutoDL challenge

AutoDL challenge

Towards fully automated deep learning…

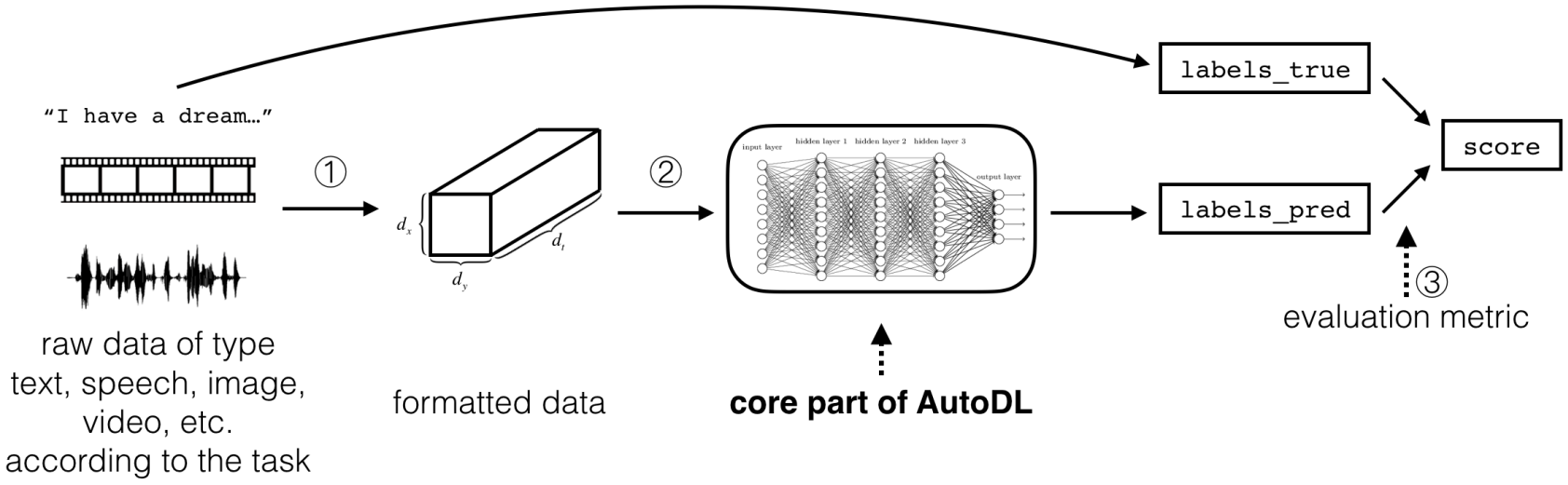


Human intuition

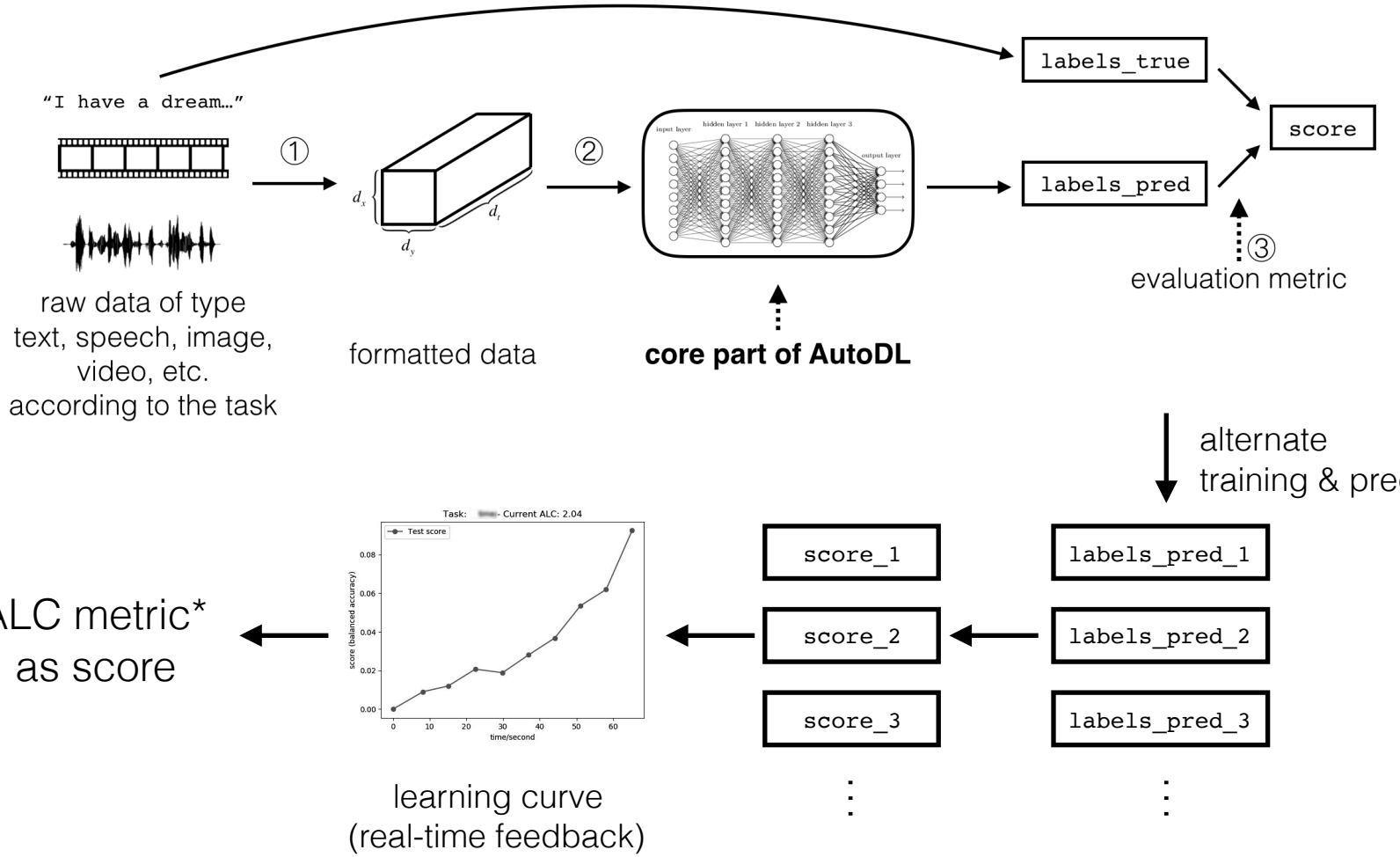


AutoDL

AutoDL process



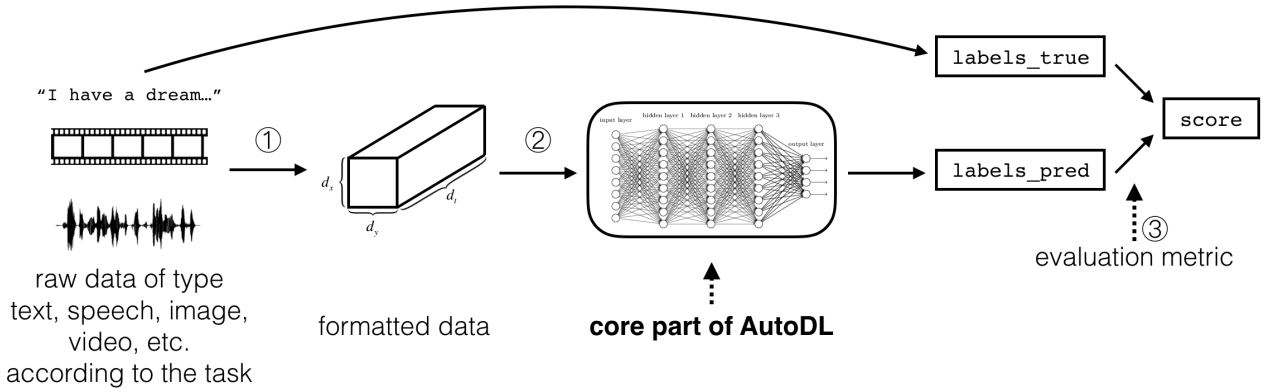
AutoDL process (continued)



*ALC: Area under Learning Curve

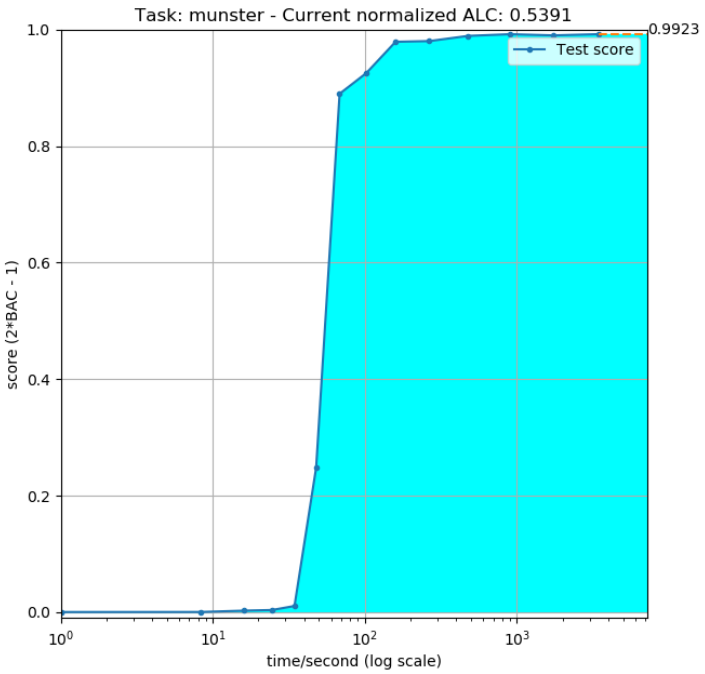
AutoDL challenge

- Similar settings but with:
  - bigger, raw datasets
  - 3D tensor representation
  - any-time setting
- support from Google
- multi-label tasks
- ALC metric



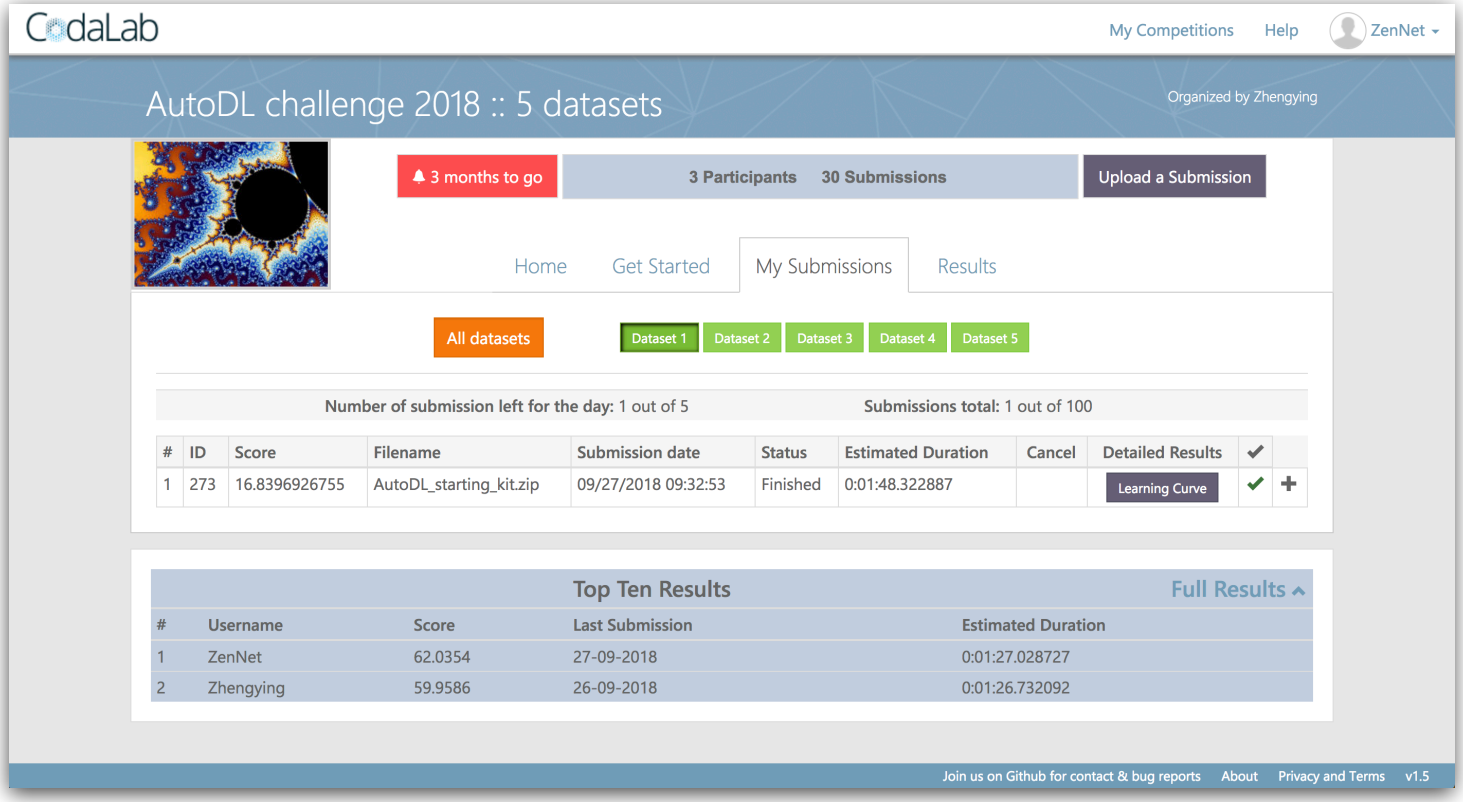
AutoDL challenge

- New features*:
  - Google Cloud
  - parallel tasks
  - real-time learning curve
  - task canceling
  - new UI
  - …



*on CodaLab platform

AutoDL challenge



Coming soon!!

Conclusion

Take-home messages

Exploration-exploitation trade-off: keep it in mind!

Ensemble methods usually win

Join the AutoDL challenge!

Open problems

Theoretical possibility of AutoML

Can we beat "No Free Lunch"? Why? How?

Computational considerations

Statistical vs Computational trade-off: what's the limit?

Theoretical guarantee of ensemble methods

Rigorous mathematical proof of the effectiveness of ensemble methods

Thank you!
Questions?

Join us: automl@chalearn.org

- as participant:
  - of **AutoDL**
- as organizer of **AutoDL**:
  - contribute data
  - prepare starting kit