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

\[ D = \{X_i, Y_i\} \xrightarrow{\beta_{\lambda}} \alpha_{\theta} \xrightarrow{D_{va}} P(\alpha_{\theta}) \]

Dogs vs Cats dataset

- "dog"
- "cat"
- ...

CIFAR-10 dataset

Iris dataset

encoded by: hyperparameters \( \lambda \in \Lambda \)

hand-crafted by ML experts

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

Hand-crafted Models:
- another trained CNN
- trained SVM (for another \( A \))
Machine Learning

\[ D = \{ X_i, Y_i \} \]

\[ P(\alpha_\theta) \]

performance (e.g. accuracy)

\[ \text{Dogs vs Cats dataset} \]

 encoded by: hyperparameters \( \lambda \in \Lambda \)

hand-crafted by ML experts
Today’s lecture

TRAINING DATA → AutoML black box → Trained model

Query x → Answer y
The AutoML problem: definition

\[
\max_{\gamma} \sum_{\mathcal{D}_{tr}, \mathcal{D}_{te} \in \mathcal{D}_{te}} P(\hat{\alpha}; D_{te}) \quad \text{where} \quad \hat{\alpha} = \hat{\beta}(D_{tr}) \quad \text{and} \quad \hat{\beta} = \gamma(\mathcal{D}_{tr})
\]

learning to learn \quad \leftrightarrow \quad \text{two layers of learning}

\( P(\hat{\alpha}; D_{te}) \) may involve time

initially we may have \( \mathcal{D}_{tr} = \emptyset \)

computational efficiency: should be not only correct but also fast

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 $\alpha$-level algorithm is the ultimate goal. $\gamma$-level algorithms exploit data from *all past experience*, in the form of a “meta-dataset”, to allow us to select a better $\beta$-level algorithm, which in turn exploits the dataset of a given task to produce an $\alpha$-level algorithm by training.

<table>
<thead>
<tr>
<th>Level</th>
<th>Input</th>
<th>Output</th>
<th>Examples</th>
<th>Encoded by</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$-level</td>
<td>sample or example (e.g. an image)</td>
<td>prediction of label (e.g. ‘dog’ or ‘cat’)</td>
<td>heuristically hard-coded classifier or already trained classifier</td>
<td>parameters, hyper-parameters (if any) and meta-parameters (if any)</td>
</tr>
<tr>
<td>$\beta$-level</td>
<td>task/dataset (e.g. MNIST, CIFAR-10)</td>
<td>$\alpha$-level algorithm</td>
<td>learning algorithms (e.g. SVM, CNN); HPO algorithms (e.g. grid search cross-validation, SMAC [56], NAS [124])</td>
<td>hyper-parameters and meta-parameters (if any)</td>
</tr>
<tr>
<td>$\gamma$-level</td>
<td>meta-dataset (e.g. OpenML [115])</td>
<td>$\beta$-level algorithm</td>
<td>meta-learning algorithms (e.g. meta-learning part in Auto-sklearn [36]); <strong>algorithms from this thesis.</strong></td>
<td>meta-parameters</td>
</tr>
</tbody>
</table>
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

Auto ML

AUTO KERAS

Auto-Sklearn
AutoML methods
with application to Deep Learning
We'll focus on the simplest case

\[ \mathcal{D}_{tr} = \varnothing \] (initially) and \[ \mathcal{D}_{te} = \{(D_{tr}, D_{te})\} \] (single dataset)

Hyperparameter Optimization

single fixed training dataset: \( D_{tr} \)

we only need to focus on \( \beta_\lambda, \lambda \in \Lambda \)

Reminder:

\[
\max_{\gamma} \sum_{D_{tr}, D_{te} \in \mathcal{D}_{te}} P(\hat{\alpha}; D_{te}) \quad \text{where } \hat{\alpha} = \hat{\beta}(D_{tr}) \text{ and } \hat{\beta} = \gamma(\mathcal{D}_{tr})
\]
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: \( P(\hat{\alpha}; D_{te}) \) \( \Rightarrow \) use an estimation (e.g. CV): \( \hat{P}(\lambda) \)

so usually the problem becomes

\[
\max_{\lambda \in \Lambda} \hat{P}(\lambda)
\]

black-box optimization

expensive to compute

\( \Rightarrow \) surrogate model

(not discussed)

where

\[
\hat{P} : \Lambda \rightarrow \mathbb{R}
\]

\( \lambda \mapsto s = \hat{P}(\lambda) \approx P(\beta_\lambda(D_{tr}), D_{va}) \)

is an estimation of the test score

Remark: some approaches optimize \( \lambda \) and \( \theta \) at the same time

\( \Rightarrow \) bi-level optimization

(ex. DARTS H. Liu et al., 2018)
$\beta_\lambda, \lambda \in \Lambda$ encodes an architecture $A$

3 ingredients in HPO (NAS):

- Search space
- Search strategy
- Performance estimation strategy
Search Space (for DL)

\[ \beta, \lambda \in \Lambda : \text{architecture, optimizer, regularization, etc} \]

chain-structured (feed-forward)

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

\[ L_{i}^{\text{in}} = L_{i-1}^{\text{out}} \]

multi-branch

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

Different layer types are visualized by different colors.

Search Space (for DL)

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

"NASNet search space" only uses two building blocks

Search Strategy

- Model-Free
  - Grid Search
    - Experiential
      - Response Surface Methods
    - Design of Experiments
  - Random Search
- Model-Based
  - Bayesian Optimization
  - Evolutionary Algorithms
  - Reinforcement Learning
  - Other Methods
Grid Search (exhaustive 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

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...?)

Evolutionary Algorithms

Population-based derivative-free optimization methods

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

- 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
Evolutionary Algorithm: an example


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

Bayesian Optimization

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

Original idea:
\(\lambda\) 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
Bayesian Optimization (cont'd)

\[
\max_{\lambda \in \Lambda} \hat{P}(\lambda) \quad \text{with} \quad \hat{P} : \Lambda \rightarrow \mathbb{R} \\
\lambda \mapsto s
\]

usual acquisition function: \( \text{Expected Improvement (EI)} \)
\[
a_{\text{EI}}(\lambda \mid D_n) = \mathbb{E}[\max(\hat{P}(\lambda) - s_{\text{max}}, 0)]
\]

usual prior model: \( \text{Gaussian Process (GP)} \)

but state-of-the-art tends to use \text{tree-based} classifier such as \text{Random Forest} to model
\[
\hat{P}(\lambda) \quad (\text{or} \quad p(s \mid \lambda))
\]

(thus not so Bayesian anymore...), see Auto-sklearn
Bayesian Optimization: an example
Swersky K, Snoek J, Adams 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

\[ f(x) \mapsto y \] instead of \[ \hat{P} : \lambda \mapsto s \]
Reinforcement Learning

A reminder:

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

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

Goal: Learn a policy: $\pi(s, a) = p(a | s) : S \times A \rightarrow [0, 1]$ that maximizes the (discounted) expected return

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

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
Reinforcement Learning: an example

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

Objective:

\[ J(\theta_c) = \mathbb{E}_{P(a_{1:T};\theta_c)}[R] \]

REINFORCE rule:

\[ \nabla_{\theta_c} J(\theta_c) = \sum_{t=1}^{T} \mathbb{E}_{P(a_{1:T};\theta_c)}[ \nabla_{\theta_c} \log P(a_t|a_{(t-1):1};\theta_c) R ] \]

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

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

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)

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

Find

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

I.e. We want to learn some transferable knowledge across datasets (a meta-learning model $\gamma$), 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


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 \( D_i \); with \( \omega_i \) the optimal model for \( D_i \)

MAML finds a generally good solution:

\[
\omega = \arg\max \sum_{D_i} s_{D_i} (\omega - \alpha \nabla_\omega s_{D_i})
\]

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

---

**Diagram:**

- \( \omega \): Parameter vector being meta-learned
- \( \omega_i^* \): Optimal parameter vector for task 1
- Meta-learning
- Adaptation
AutoML challenges
The AutoML challenge (Guyon et al., 2015-2016)

Task variabilities:
- classification / regression
- various scoring functions
- various time budget
- etc.

Goal: Find a process to identify the best $\beta_\lambda$ 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/

auto-sklearn is an automated machine learning toolkit and a drop-in replacement for a scikit-learn estimator:

```python
>>> import autosklearn.classification
>>> cls = autosklearn.classification.AutoSklearnClassifier()
>>> cls.fit(X_train, y_train)
>>> predictions = cls.predict(X_test)
```
AutoDL

https://autodl.chalearn.org/
AutoDL challenge 2019-2020

(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.

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

<table>
<thead>
<tr>
<th>Approach</th>
<th>Image</th>
<th>Video</th>
<th>Speech</th>
<th>Text</th>
<th>Tabular</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Meta-learning</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offline meta-training</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>transferred with AutoFolio [25] based on meta-features (<a href="#">automl freiburg</a>)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offline meta-training generating solution agents, searching for optimal sub-operators in predefined sub-spaces, based on dataset meta-data. (<a href="#">DeepWisdom</a>)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAML-like method [17] (team zhaw)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Preprocessing</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Image cropping and data augmentation (<a href="#">PASANJU</a>), fast autoaugment (<a href="#">DeepBlueAI</a>)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sub-sampling keeping 1/6 frames and adaptive image size (<a href="#">DeepBlueAI</a>)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adaptive image size</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mel Spectrogram, STFT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MFCC, MFCC extractors with stemmer, meaningless words filtering (<a href="#">DeepBlueAI</a>)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Hyperparameter Optimization</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offline with BOHB [26] (Bayesian Optimization and Multi-armed Bandit) (<a href="#">automl freiburg</a>) Sequential Model-Based Optimization for General Algorithm Configuration (SMAC) (<a href="#">automl freiburg</a>)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Transfer learning</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pre-trained on ImageNet [28] (all teams except Kon)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pre-trained on ImageNet [28] (all top-8 teams except Kon) MC3 model pretrained on Kinetics (<a href="#">DeepWisdom</a>)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Ensemble learning</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adaptive Ensemble Learning (ensemble latest 2 to 5 predictions) (<a href="#">DeepBlueAI</a>)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ensemble Selection [29] (top 5 validation predictions are fused) (<a href="#">DeepBlueAI</a>, Ensemble models sampling 3, 10, 12 frames (<a href="#">DeepBlueAI</a>)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FastText pre-trained on Common Crawl (<a href="#">frozenmad</a>)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last best predictions ensemble strategy (<a href="#">DeepWisdom</a>) averaging 5 best overall and best of each model: LR, CNN, CNN+GRU (<a href="#">DeepBlueAI</a>)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weighted Ensemble over 20 best models [29] (<a href="#">DeepWisdom</a>)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team</td>
<td>Image</td>
<td>Video</td>
<td>Speech</td>
<td>Text</td>
<td>Tabular</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DeepWisdom</td>
<td><strong>ResNet-18</strong> and ResNet-9 models [pretrained on ImageNet]</td>
<td>[MC3 model] [pretrained on Kinetics]</td>
<td>[fewshot learning] [LR, ThinRestnet34 model][pretrained on VoxCeleb2]</td>
<td>[fewshot learning] [task difficulty and similarity evaluation for model selection] [SVM, TextCNN, fewshot learning] RCNN, GRU, GRU with Attention</td>
<td>[LightGBM, Xgboost, Catboost, DNN models] [no pretrained]</td>
</tr>
<tr>
<td>DeepBlueAI</td>
<td>[data augmentation with Fast AutoAugment] [ResNet-18 model]</td>
<td>subsampling keeping 1/6 frames [Fusion of 2 best models]</td>
<td>[iterative data loader (7, 28, 66, 90%)] [MFCC and Mel Spectrogram preprocessing] [LR, CNN, CNN+GRU models]</td>
<td>[Samples truncation and meaningless words filtering] [Fasttext, TextCNN, BiGRU models] [Ensemble with restrictive linear model]</td>
<td>[3 LightGBM models] [Ensemble with Bagging]</td>
</tr>
<tr>
<td>PASA NJU</td>
<td><strong>ResNet-18</strong> and SeResnext50; preprocessing: shape standardization and image flip (data augmentation)</td>
<td><strong>ResNet-18</strong> and SeResnext50; preprocessing: shape standardization and image flip (data augmentation)</td>
<td>[data truncation(2.5s to 22.5s)][LSTM, VggVox ResNet with pretrained weights of DeepWisdom(AutoSpeech2019) ThinRestnet34?]</td>
<td>[data truncation(300 to 1600 words)][TF-IDF and word embedding]</td>
<td>[iterative data loading] [Non Neural Nets models] [models complexity increasing over time] [Bayesian Optimization of hyperparameters]</td>
</tr>
<tr>
<td>frozenmad</td>
<td>[images resized under 128x128] [progressive data loading increasing over time and epochs] [ResNet-18 model] [pretrained on ImageNet]</td>
<td>[Successive frames difference as input of the model] [pretrained ResNet-18 with RNN models]</td>
<td>[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]</td>
<td>[TF-IDF and BERT tokenizers] [SVM, RandomForest, CNN, tinyBERT ]</td>
<td>[progressive data loading] [no preprocessing] [Vanilla Decision Tree, RandomForest, Gradient Boosting models applied sequentially over time]</td>
</tr>
</tbody>
</table>
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

**Alpha level:** predict() in sklearn, a classifier

- Input: $x$ (example/sample, e.g. an image)
- Output: $y$ (labels)
- Code Jam LeetCode

**Beta level:** fit() in sklearn, a learning algo.

- Input: $T$ (ML task, dataset)
- Output: $\alpha$ (alpha-level algo)
- (Auto)ML challenges & AutoDL

**Gamma level:** meta_fit() on a meta-dataset

- Input: $\mathcal{D}$ (Meta-dataset)
- Output: $\beta$ (beta-level algo)

Check and stay tuned [https://metalearning.chalearn.org/](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
\implies Cross-domain meta-learning yet to be studied

Any-time learning aspect to be studied further
Stay tuned! autodl.chalearn.org

AutoDL challenges

AutoDL challenges

Following the success of AutoDL 2019-2020 (which was part of the competition selection of NeurIPS 2019, see our workshop page), we are continuing to organize a series of challenges.

Coming soon KDD 2020 will be held in San Diego, CA, USA from August 23 to 27, 2020. The Automatic Graph

Sign up

You will be notified of our new challenges

* Required

Email *

Your answer