Evolving Fully Automated Machine Learning via Life-Long Knowledge Anchors

Xiawu Zheng, Yang Zhang, Sirui Hong, Huixia Li, Lang Tang, Youcheng Xiong, Jin Zhou, Yan Wang, Xiaoshuai Sun, *Member, IEEE*, Pengfei Zhu, Chenglin Wu, and Rongrong Ji, *Senior Member, IEEE*

Abstract—Automated machine learning (AutoML) has achieved remarkable progress on various tasks, which is attributed to its minimal involvement of manual feature and model designs. However, most of existing AutoML pipelines only touch parts of the full machine learning pipeline, e.g., neural architecture search or optimizer selection. This leaves potentially important components such as data cleaning and model ensemble out of the optimization, and still results in considerable human involvement and suboptimal performance. The main challenges lie in the huge search space assembling all possibilities over all components, as well as the generalization ability over different tasks like image, text, and tabular *etc*. In this paper, we present a first-of-its-kind fully AutoML pipeline, to comprehensively automate data preprocessing, feature engineering, model generation/selection/training and ensemble for an arbitrary dataset and evaluation metric. Our innovation lies in the comprehensive scope of a learning pipeline, with a novel "life-long" knowledge anchor design to fundamentally accelerate the search over the full search space. Such knowledge anchors record detailed information of pipelines and integrates them with an evolutionary algorithm for joint optimization across components. Experiments demonstrate that the result pipeline achieves state-of-the-art performance on multiple datasets and modalities. Specifically, the proposed framework was extensively evaluated in the NeurIPS 2019 AutoDL challenge, and won the *only champion* with a significant gap against other approaches, on all the image, video, speech, text and tabular tracks.

Index Terms—Fully automated machine learning, life-long learning, evolutionary algorithm

1 Introduction

In recent years, deep neural networks [1], [2], [3], [4], [5] have achieved extraordinary success on various pattern analysis and machine learning tasks, ranging from image classification and object detection in computer vision to semantic parsing and sentence boundary detection in natural language processing. Despite the remarkable progress, existing deep learning models still heavily involve expert designs throughout the entire pipeline, such as data preprocessing [6], feature engineering [7], model design [3], optimizer selection [8], andmodel deployment [9]. Towards a learning pipeline without heavy handcrafting from experts, automated machine learning (AutoML) has been recently promoted [10], which aims to maximize the model accuracy without consuming human efforts in selecting and implementing aforementioned pipeline components. Despite the early endeavors in the pre-deep-learning era [1], [2], [3], AutoML has recently emerged and largely extends the application scope of deep learning models to handle various specific tasks with an extremely limited expert cost. However, existing AutoML frameworks can be still regarded as a *semi-auto* one (denoted as *Semi-AutoML*), which focuses on automating one or a few isolated parts aforementioned in the entire learning pipeline. On one hand, *Neural Architecture Search* (NAS) only searches for the optimal deep model with a fine-grained search space designed by an expert and covered variations only *within* the model design component. On the other hand, traditional AutoML methods [11], [12] search hyper-parameters of a given model over other isolated aspects, such as data preprocessing [6] and back-propagation [12], as illustrated in Table 1. In these works, most parts of the learning pipeline are still searched isolated. Such a Semi-AutoML paradigm might save computational cost due to the usage of small/partial search space, which however has two fundamental drawbacks: First, searching isolated components over the entire learning pipeline is certainly sub-optimal and may bias the output solution, as there is no guarantee that different parts in the learning pipeline are independent of each other. Second, the fine-grained search space needs to be carefully composed [13], and thus creates a new burden in the design and deployment of learning

- *Xiawu Zheng, Huixia Li, and Lang Tang are with the Media Analytics and Computing Lab, Department of Artificial Intelligence, School of Informatics, Xiamen University, Xiamen 361005, China, and also with Peng Cheng Laboratory, Shenzhen 518066, China. E-mail: {zhengxiawu, hxlee, langt}@stu.xmu.edu.cn.*
- *Yang Zhang, Sirui Hong, Youcheng Xiong, Jin Zhou, and Chenglin Wu are with DeepWisdom, Ltd, Xiamen 361000, China. E-mail: {youngzhang, stellahong, ycxiong, jinzhou, alexanderwu}@fuzhi.ai.*
- *Yan Wang is with the Pinterest, Seattle, WA 98100 USA. E-mail: yanw@pinterest.com.*
- *Xiaoshuai Sun is with the Media Analytics and Computing Lab, Department of Artificial Intelligence, School of Informatics, Xiamen University, Xiamen 361005, China. E-mail: xssun@xmu.edu.cn.*
- *Pengfei Zhu is with the College of Intelligence and Computing, Tianjin University, Tianjin 300072, China. E-mail: zhupengfei@tju.edu.cn.*
- *Rongrong Ji is with the Media Analytics and Computing Lab, Department of Artificial Intelligence, School of Informatics, Xiamen University, Xiamen 361005, China, with the Institute of Artificial Intelligence, Xiamen University, Xiamen 361005, China, and also with the Peng Cheng Laboratory, Shenzhen 518066, China. E-mail: rrji@xmu.edu.cn.*

Manuscript received 15 May 2020; revised 17 Feb. 2021; accepted 23 Mar. 2021. Date of publication 29 Mar. 2021; date of current version 4 Aug. 2021. (Corresponding authors: Pengfei Zhu, Chenglin Wu, and Rongrong Ji.) Recommended for acceptance by H. J. Escalante, J.Vanschoren,W.-W. Tu, Y. Yu, S. Escalera, N. Pillay, R. Qu, N. Houlsby, and T. Zhang. Digital Object Identifier no. 10.1109/TPAMI.2021.3069250

TABLE 1
Comparison Between Our Framework and Other Widely-Used AutoML Frameworks

Method	Search Space	Optimization	Meta learning	Parallelizable
Auto-Weka [12]	FE/MS-HO	SMAC/TPE	×	×
AutoPrognosis [14]	DP/FE/MS-HO	BO	×	×
P4ML [15]	DP/FE/MS/HO/EN	-	×	×
Auto-Sklearn [16]	DP-FE-MS-HO/EN	BO	✓	×
TPOT [17]	DP-FE-MS-HO	EA	×	✓
Ours	DP-FE-MS-HO-EN	EA	✓	✓

The second column denotes the machine learning pipelines in the search space of different frameworks, with abbreviations as follows: DP - *data preprocessing;* FE - *feature engineering;* MS - *model selection;* HO - *hyperparameters optimization;* EN - *ensemble. To better represent the relationship between different components in the search space, '|' and '-' denote two components are searched hierarchically and integrally, respectively.*

systems, which is opposite to the original objective of AutoML. It is thus highly demanded that the AutoML pipeline directly makes use of off-the-shelf machine learning models and needs only negligible human efforts. Besides, the search results also tend to overfit specific datasets or domains, e.g., the architecture found by [13] for classification task performs poorly in other tasks such as object detection or semantic segmentation.

In this paper, we present a *Fully-AutoML* that breaks the isolated design of separated search space, linking all machine learning components in a unified way without *any* human efforts as summarized in Table 1. Essentially, the proposed Fully-AutoML paradigm aims to search within a coarse-grained space covering all learning procedures and all aforementioned components, such as data preprocessing, feature engineering, model search, model training, and ensemble, permitting no or limited human-design and enabling the pipeline to be built *without* ML prior.

Comparing to the existing Semi-AutoML methods, the major challenge of Fully-AutoML lies in the difficulty of searching in a large search space covering all components, which has disabled most existing methods in architecture search and automated feature engineering, like [13], [18]. For example, the work in [19] has found that most NAS methods are only marginally superior to random search [11], [19] when facing a cell-based search space. Typically, EfficientNet [16], i.e., the current state-of-the-art deep learning model for image classification, was built by a simple grid search. To explain, existing methods only need to independently optimize specific components in the full pipeline, such as fine-grained model search [13] or data augmentation [6]. In contrast, the search space in Fully-AutoML is generic and comprised of all possibilities of every component in the entire pipeline, in which the potential optimal solutions could be quite sparse. Given such a huge search space the existing black-box optimization methods are unable to be directly deployed, e.g., optimization algorithms have to find the optimal solution from 2^{80} candidates in our Fully-AutoML search space. On the other hand, we are inspired by the increasing amount of architectures accumulated over different datasets for the same task. For example, ResNet [3] has demonstrated superior performance on various computer vision tasks and there has been a large variety of ResNet models available for tasks like face recognition and image classification. To sum up, for each component in the search space, the state-of-the-art algorithms/strategies/models are prevailed in literature, forming preliminary solutions like *anchors*. How to efficiently and effectively leverage such anchors as prior knowledge to achieve efficient search over the above full search space is an exciting topic but left unexploited.

In view of the above insights, in this paper we propose a novel fully AutoML pipeline with *life-long* knowledge anchors, which is the first of its kind. In particular, we encode every component involved in the learning pipeline, as well as the corresponding operations in each component, as a vector to form a huge and full search space. We then propose to efficiently handle the optimization with a life-long evolutionary algorithm. In particular, we first randomly initialize a set of vectors, each of which represents a specific example in the full search space example. Vectors with high performance are selected to randomly segment and combine for crossover, assign a new value to a certain dimension for mutating, and then inherit to the next population. To improve the search efficacy in finding optimal combinations, a novel knowledge anchor scheme is proposed to accelerate the search procedure. Specifically, knowledge anchors are defined as the optimal examples in the previous search process for a specific task and benchmark, which contains detailed information including data modalities and other statistics such as data size and length.[1] For a specific task, to pick up knowledge anchors for a new dataset, we first compute the similarities between the target and the anchor datasets by quantitatively characterizing their major properties, i.e., datasets labels, sizes, image resolutions, and categorical distributions. After that, knowledge anchors with higher similarities are selected with a higher probability to compose part of the initial population in the evolutionary algorithm [20]. In this way, knowledge anchors can provide a prior for mutation, reproduction, and crossover, as to largely accelerate the optimization process. Furthermore, we have also observed that the optimal pipelines found in different datasets are very similar, which inspires us to maintain knowledge anchors that have good results in multiple data sets in a *life-long* style, as quantitatively demonstrated latter in Section 3.3, can further improve the generalizability of the search algorithm.

By embedding life-long knowledge anchor into the evolutionary based fully AutoML pipeline, the proposed

1. For each dataset in a dataset repository, we first directly use the proposed search algorithm (without any knowledge anchor) to find and store an instantiation of the pipeline with a strong empirical performance for that dataset, which is considered as a knowledge anchor.

method effectively finds good solutions that surpass human designs despite the enormous size and sparsity in the full search space, as validated in Section 4. More importantly, the proposed fully AutoML framework has won the 1st champion on NeurIPS 2019 AutoDL challenge,[2] outperforming all state-of-the-arts and alternatives in all five modalities, including image, video, speech, text and tabular. We have released the submitted code to share our progress on Fully-AutoML with the community at https://github.com/DeepWisdom/AutoDL

In summary, our contributions are listed as follows:

- For the first time, the Fully AutoML paradigm is proposed in the literature, which directly searches the whole learning pipeline covering data preprocessing, feature engineering (extraction/construction/transformation), model selection, training hyperparameter optimization, and ensemble jointly.
- To tackle such a huge full search space, we present an evolutionary algorithm to achieve fully AutoML, and propose a life-long knowledge anchor scheme for search acceleration.
- The found pipelines were extensively evaluated in AutoDL contest, which won the 1st champion in NeurIPS 2019 AutoDL challenge with significantly highest ALC in all 5 modalities. All of the experiments in our paper are reproductive by the released source codes.

2 Related Work

2.1 AutoML

AutoML has became popular for various machine learning tasks. It searches for the best learning configurations, e.g., architectures, and hyper parameters. The earlier AutoML work was proposed in [21], which aims to find hyperparameters and choose algorithms for an ensemble method by using Particle Swarm Model Selection. Later works started to explore different black-box optimization methods for different learning components, such as Auto-WEKA [12] and Hyperopt-sklearn [22]. Feurer *et al.* [23] extended the work in Auto-WEKA and Hyperopt-sklearn by employing meta and ensemble learning. Auto-Prognosis [14] split the whole learning pipeline into smaller optimization problems, which are then solved by Bayesian Optimization. Snoek *et al.* [24] used Bayesian optimization or evolution strategy to find the optimal hyperparameters. Chen *et al.* [25] integrated a hierarchical stacking models with EA to find optimal ensemble models. Also, other works have tried to get the optimal parameters for specific layers [26], full forward pass [27], and data augmentation strategies [6], [28], *etc.* As summarized in Table 1, previous works include similar machine learning components with the proposed fully AutoML search space. However, they did not consider these components integrally, which largely limited the performance of the searched pipelines.

Neural architecture search has become a widely-studied endeavor of AutoML, which automates the manual task of neural network architecture design and has attracted extensive focus previously [13], [29], [30], [31] and recently [10], [32]. Zoph *et al.* [13] proposed to explore and design transformable network building blocks with the reinforcement learning (RL). Motivated to alleviate the huge computational demand of this strategy, another recent trend in NAS is differentiable search, such as DARTS[33]. However, both aforementioned directions need fine-grained expert-designed search space and are limited in specific datasets. In contrast, the proposed approach aims to optimize the overall learning pipeline from end to end.

However, the search space of the above-mentioned methods is isolated from each other, which are termed as Semi-AutoML in this paper. Such a paradigm has two major drawbacks. On one hand, there is no guarantee that each part of the learning pipeline is independent, which may lead to a sub-optimal solution. On the other hand, semi-AutoML still needs a lot of human effort to design search space and other parts of the learning pipeline. In this paper, we propose to do a joint search on every component of the overall learning pipeline, which is demonstrated to have a better performance on various modalities and generalizes well to new datasets.

2.2 Lifelong Learning and Meta Learning

Lifelong learning is often regarded as an online multi-task learning [34], [35], which aims to efficiently learn on different tasks based on previous knowledge while optimizing the performance across all tasks seen so far. It remains as a challenging task [36], and its major challenge is catastrophic forgetting [37], [38], which means the resulting model overfits to the latest task while forgetting the previous tasks. A straightforward fix is to retrain the model with previous tasks by storing old training examples [39], [40], whose storage and training time is however intractable especially in the deep learning era. Another approach is regularization, for example, adding a quadratic penalty on the difference between the weights for previous tasks and the new task [41], [42], [43]. This is also supported by recent findings that a general model may also perform well on multiple tasks with transfer learning [3], [16]. For example, In [44], the model selects a path for the current task by a tournament selection genetic algorithm. When learning for a new task, the weights in paths of previous tasks are frozen so that the knowledge can be preserved. Serra *et al.* [45] proposed a hard attention mechanism to select the path for different tasks. However, this approach also hits a challenge that, as the number of tasks increases, a fixed network with limited capacity can hardly maintain satisfactory performance. Therefore, Hu *et al.* [46] splits the parameters of the model into two parts. The first part is shared by all tasks learned so far, and the second part is dynamically generated to adapt to suit each test example in order to classify correctly. Combined with automated strategies, some recent methods [47], [48], [49] have been proposed for automatically selecting parameters and network sizes. Other methods [50], [51], [52] design dynamic network extensions for task-specific parts (e.g., filters, and neurons). For example, Xu *et al.* [53] assigned different new modules for different tasks by a controller trained using reinforcement learning.

2. https://autodl.lri.fr/competitions/162

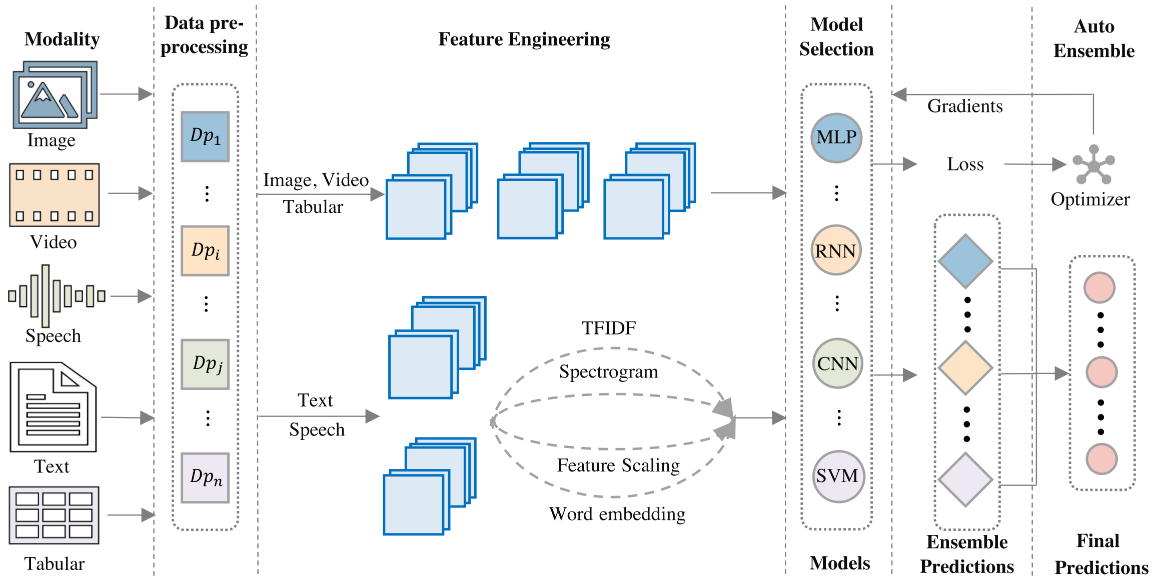


Fig. 1. The search space on a machine learning pipeline consisting of different components for different modalities in the proposed Fully-AutoML paradigm.

Another related topic is meta learning [54], which aims to estimate the performance of learning algorithms across datasets. Specifically, previous works [23], [55], [56] apply meta-features to select instantiations of given machine learning pipelines (similar to knowledge anchors in this paper) that are likely to perform well on a new dataset. For a large number of datasets, they collect both quantitative performance scores and a set of meta-features, i.e., characteristics of the dataset that can be computed efficiently and help determine which algorithm to use on a new dataset. However, these meta-features highly depend on human efforts and tend to overfit in new datasets. For example, the approach in [57] requires 15 different meta-features that are carefully selected and thoroughly studied in previous literatures [58].

The approach proposed in this paper is related yet intrinsically different from both lifelong learning and meta learning. We follow the idea of lifelong learning and meta learning in that our approach has memory on the previously learned tasks, so the AutoML search has a good initialization. However, we introduce novel meta-features together with probabilistic sampling to greatly reduce human efforts, and also avoid overfit in the search process.

3 Approach

In this section, we introduce the proposed approach in detail. First we introduce how we define the search space (Section 3.1). Then we illustrate how the search is performed by solving a joint optimization problem (Section 3.2) and taking advantage of lifelong learning (Section 3.3), followed by more detailed discussions on implementation details (Section 3.4).

Given a dataset $D = \{x_i, y_i\}_{i=1}^n$ with a train/validation split $D = D_{\text{tr}} \cup D_{\text{vl}}$, we aim to automatically construct a classification algorithm machine learning (pipeline) f under an evaluation metric P. Following [59], P is the average validation loss

$$P(\hat{f}; D_{\text{vl}}, L) = -\frac{1}{|D_{\text{vl}}|} \sum_{(x,y)\in D_{\text{vl}}} L\Big(\hat{f}(x), y\Big). \tag{1}$$

Here L is the loss function taking the prediction $f(x)$ and ground truth y as input to measure the performance.

3.1 Search Space Design

Given the above optimization objective, we further define the search space in this subsection. As illustrated in Fig. 1, we divide the search space into 5 components including data preprocessing, feature engineering, model selection, model optimization and ensemble.

For a given component, there are multiple approaches to choose from, which are not necessarily mutually exclusive. Therefore, when making a selection of a specific operation in the search space, e.g., *input resizing*, we use a binary indicator and related hyperparameters to describe whether and how to choose suitable operations. While some hyperparameters are continuous numbers, we discretize the continuous hyperparameter in a few candidates for simplicity. In this way, we encode the selection of data preprocessing and feature engineering components into a fixed-length vector, which is also compatible with all modalities. For example, the binary indicator of *input resize* and *frame selection* is always 0 (not picked) in text and image modalities.

For model selection, ensemble and optimization, these three components are highly correlated, i.e., ensemble determines the number of models, and meanwhile, the model selection also determines the range of optimization methods. Without loss of generality, we set the maximum number of models to be fixed. Each model is generated by three sequential choices. The first choice is whether to use the model, which determines the number of models. Since the select domain of optimization methods is decided by the corresponding models, the second choice is the algorithm in model selection, and the third choose the optimization method and hyperparameters of the selected model. It is worth noting that, for those search algorithms with

TABLE 2
Summary of the Operations and the Number of Hyperparameters That We Consider for Each Component in the Search Space

Data Preprocessing	Feature Engineering	Model Selection	Optimization	Ensemble
Sampling (2)	TFIDF (1)	Logistic regression (5)	SGD (2)	Instance Bagging (1)
Resize (2)	Word embedding (3)	SVM (4)	RMSprop (4)	Prediction Ensemble (3)
Crop (2)	Spectrogram (4)	Catboost (9)	Adam (3)	
Cutoff (2)	Feature scaling (1)	LightGBM (9)	Lr scheduler(4)	
Padding (2)	Tabular features (7)	Xgboost (9)	Epoch (1)	
Test augmentation (1)	Text level (1)	Resnet 9/18 (4)	Batchsize (1)	
Frame selection (2)		DNN (4)		
Scaling (2)		Thin-ResNet (6)		
Clean (1)		Text-CNN (7)		
Data split (1)		GRU (10)		

multiple hyperparameters, we encode the hyperparameters as a tuple, and then select suitable hyperparameters among candidate tuples. In order to ensure there is at least one model in the pipeline, we directly select the corresponding model through the second and the third selections when generating the first model.

Based on the above formulation, we use a discrete vector with fixed dimensions to represent the overall pipeline. And the search space now is determined by what operations to consider in each component. Table 2 gives a summary of the operations we consider for this approach. We will discuss more details in Section 3.4. Note this is by no means an exhaustive list. And adding new operations for any component is easy. In our case, the average of the hyperparameters' candidate number is 5, therefore, the maximum number of the pipeline is $2^{12} \times 5^{12} \times 2^3 \times 10^3 \times 5^3$, which is an extremely large space to search, and thus requires efficient optimization methods.

3.2 Evolutionary Algorithm

We use an evolutionary algorithm [60] to search for the optimal solution due to its simplicity and effectiveness in searching large search spaces [18], [61]. An evolutionary mimics the biology evolution by setting up an initial population, and performing operations such as mutation, reproduction (crossover), and selection. With proper selection pressure towards the optimization objective, the evolutionary algorithm could explore a wide variety of options and avoid local minima, by maintaining a diverse population. The framework of the proposed algorithm is illustrated in Fig. 2.

Our initial population is set up around a set of *knowledge anchors*, which are known as promising ML pipelines derived from existing knowledge and will be introduced in the next subsection. We initialize the rest individuals randomly to form the overall population to ensure a balance between performance and diversity. Then the evolutionary algorithm performs the standard selection, crossover, mutation, and update operations. Specifically, each iteration selects $T < P$ best individuals as the parents. These parents are then crossed over and mutated to produce different children that are appended to the population, while the worst T individuals in the population are removed. After reaching the maximum number of epochs or the top $\frac{P}{2}$ is not updated anymore, we output the best individual in the population as the optimal solution. The detailed steps are described in the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2021.3069250.

3.3 Initialization With Life-Long Knowledge Anchors

While the evolutionary algorithm typically works with a random population, the search process could be significantly accelerated if a high-quality yet diverse population is given. We use the experience from other datasets, noted as life-long knowledge anchor, to guide the generation of the initial population, as illustrated in Fig. 2.

Our life-long learning approach works as follows. In an offline phase, for each dataset in a dataset repository, we first directly use an evolutionary algorithm (without any knowledge anchor) to find and store an instantiation of the pipeline with a strong empirical performance for that dataset as a knowledge anchor. Then, we use a data-driven approach to obtain the initial population in the following search phase. More specifically, given a dataset $\hat{D}$, we extract features from K pre-defined functions $F(.)$ which maps a dataset to a real valued vector or matrix. At the same time, the corresponding distance function is defined as $d(.)$, where the i^{th} mapping and distance function are denoted as F_i and d_i, respectively. Suppose we have $j = 1, \ldots N$ datasets, for the given dataset $\hat{D}$, we define the distance of two datasets $\hat{D}$ and D_j as

$$d(\hat{D}, D_j) = \sum_{i=1}^{K} \frac{1}{K} \cdot \frac{d_i\big(F_i(\hat{D}), F_i(D_j)\big)}{\max_j\Big(d_i\big(F_i(\hat{D}), F_i(D_j)\big)\Big)}, \quad (2)$$

where $\max_j\Big(d_i\big(F_i(\hat{D}), F_i(D_j)\big)\Big)$ is the normalization term. $F_i(.)$ and $d_i(.)$ are defined hierarchically with 5 different level including modality, task, domain, label and meta-model features, a list of which are as follows:

- F_1, F_2, F_3 encode different modalities, tasks, and domains as different one-hot vectors, respectively. For example, MNIST [68] is encoded as $10000/10/0100000$ where each component is the indicator vector of modality (image), task (classification) and domain (digit). The corresponding distance is obtained by the standard euclidean distance, formally, $d_i, i \in \{1, 2, 3\}$ are defined as $\|\cdot\|_2$.
- The label function is defined as F_4, which collects the labels that appear in the dataset and summarizes

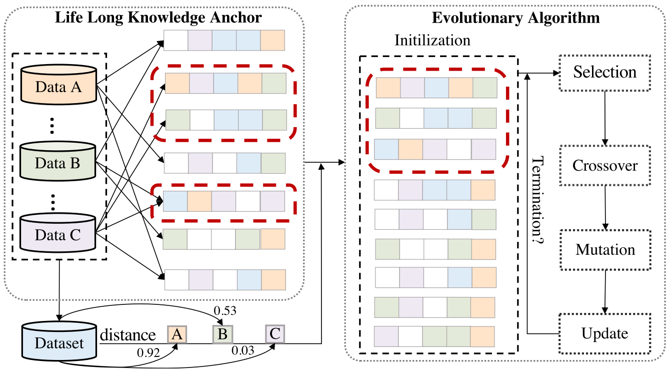


Fig. 2. The proposed evolutionary algorithm with life-long knowledge anchor. We define the optimal solutions searched for previous datasets as knowledge anchors, and record the corresponding meta information (such as labels, length, and size of the datasets). Given a new dataset, we first calculate the distance between the new dataset and the recorded datasets and sample the knowledge anchors according to the distances as a part of the initial population in the evolutionary algorithm.

them into a set of Ω, i.e., $F_4(\hat{D}) = \hat{\Omega}$. The distance function between $\hat{\Omega}$ and Ω_j is then defined as

$$d_4(\hat{\Omega}, \Omega_j) = \frac{|\hat{\Omega} \cap \Omega_j|}{|\hat{\Omega}|}, \tag{3}$$

where $|\cdot|$ is the cardinality operator.

- F_5 first randomly samples M instances from the dataset, and then uses the meta-model pre-trained on the meta-dataset to extract meta-features $\mathcal{M} \in \mathbb{R}^{M\times H}$ (assuming that the length of the feature in the corresponding model is H). Meta-models and meta-datasets for different modalities are summarized in Table 3. Note that we do not use any meta-model and meta-dataset in tabular modality due to the huge diversity between different models and datasets. The distance is defined as the standard Frobenius norm, i.e., $d_5(A,B) = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}(a_{ij}-b_{ij})^2}$.

After obtaining the distance of the datasets, we use the following probability to sample the anchors:

$$p(\eta_{i,j}) = \frac{e^{-d(D_i,D_j)}}{\sum_j e^{-d(D_i,D_j)}}. \tag{4}$$

Practically, as illustrated in Fig. 2, given a dataset, we first use Eq. (4) to obtain the probability. When a dataset is sampled, we define the optimal solutions found on the dataset as a knowledge anchor, which is then equally selected without replacement and employed as a part of the initial population in Section 3.2.

In general, the difference of the proposed life-long knowledge anchors and previous meta-learning AutoML [23], [54] scheme is *probabilistic sampling* (Eq. (4)) and *end-to-end feature extraction*. Therefore, comparing with other methods, the proposed approach is more effective with limited human efforts. In terms of "probabilistic sampling", knowledge anchors are randomly sampled with a probability according to the distance. Without probabilistic sampling, the search process tends to have a lower cost but worse performance, which causes the previous selection scheme [15], [23], [55], [56] to overfit in AutoML. On the other hand, the proposed "probabilistic sampling" method shows a clear performance gap with a negligible cost increase (corresponding experiments are supplied in Section 4.3). In terms of "end-to-end feature extraction", except for the simple meta-features including data modality, task and label, we employ meta-models trained on meta datasets to extract meta-features for different modalities. Table 3 summarizes the meta-models and datasets for different modalities. As illustrated in Table 3, pre-trained meta models and datasets are publicly available, together with the simple features, the proposed meta-feature extraction only needs negligible human efforts while showing better performance (the corresponding experiments are supplied in Section 4.3).

TABLE 3
The Meta-Models and Meta-Datasets for Different Datasets

Modality	Meta-Model	Meta-Dataset
Image	Resnet-50[3]	ImageNet[62]
Video	MC3[63]	Kinetics-400 [64]
Text	RoBERTa[65]	[65]
Speech	Thin_Resnet34[66]	VoxCeleb2 [67]
Tabular	Adopted from Auto-sklearn [23]	

In addition, the proposed meta-features are more effective. On one hand, there are only 5 meta-features in our framework, which is much less compared to other methods (e.g., 38 and 45 in auto-sklearn [23] and meta-HPO [55], respectively). On the other hand, the previous meta-features need extra human efforts for different datasets. For example, Landmarking meta-features in [15], [55], [56] need to build different machine learning models for different modalities, even for different datasets. Meanwhile, the proposed meta-features are general: The simple meta-features are general to different modalities and datasets. And the meta-features extracted from the meta-models are general to different datasets within the same modality.

3.4 Implementation Details

In the following, we provide more details on the components of the proposed AutoML framework.

3.4.1 Data Preprocessing

Some tasks or modalities are especially sensitive to the proper preprocessing of the raw data. For example, there may be many types of data errors (incomplete or redundant data, *etc.*) in a tabular dataset. Recent works [69], [70] also show that applying some augment policies on the raw data largely promotes the performance and helps achieve state-of-the-art performance on multiple datasets. There are some common data preprocessing operations for different modalities, which are listed below:

- *Resize & crop*: It is the process of unifying the input data by resize and crop operations. In this work, we use two parameters, i.e.,, resize ratio ($[0.5, 1.5]$) and crop ratio ($[0.5, 1]$), to control these operations. Generally, we sample a few data examples in the dataset, calculate the average size, and resize the data to mean size $\times$ resize ratio and crop data with mean size $\times$ crop ratio.

- *Padding*: Another alternative unifying method is padding or extension. Specifically, for this operator, we define a size that covers most input data and pad zeros or simply copies the values from the small example.
- *Scaling*: Data scaling is to normalize the range of data. In this operator, we consider four different scaling methods including min-max normalization ($\frac{x-\min(x)}{\max(x)-\min(x)}$), mean normalization ($\frac{x-\text{mean}(x)}{\max(x)-\min(x)}$), standardization ($\frac{x-\mu}{\sigma}$) and L_2 normalization ($\frac{x}{||x||}$).
- *Test time augmentation*: This is an application of data augmentation for the test dataset which is inspired by [2]. In particular, it involves creating multiple augmented copies of each data by flipping, cropping, and shifting.
- *Clean*: As mentioned before, there are a lot of missing values in tabular data and meaningless data in text data. These data will be discarded if needed, by filling with 0 or with error part otherwise.
- *Frame selection*: This search operation determines the number, the interval, and the starting point of frames for a video dataset, which may bring large acceleration without any performance drop.
- *Segmentation:* It is the process of dividing text or speech into meaningful units, such as words and sentences.
- *Samping:* The ratio of the data that we sampled in the training and testing process of the following pipeline.
- *Data split:* The ratio of the split ratio of the training and testing data in the sampled dataset.

3.4.2 Feature Engineering

How to generate good features is an important and fundamental problem in machine learning, which has been studied for decades. For example, before the success of CNNs, a lot of image features such as SIFT[7] and HOG [71], have been successfully applied in computer vision. In many cases, the original data may not be good enough, e.g., most raw data is not discriminative for the corresponding task. Consequently, we employ some widely used feature extraction methods on different modalities to improve learning performance. In image and video modalities, we have investigated a lot of human-designed features and found that the end to end deep learning-based features are quite sufficient in the tested tasks. Therefore, feature engineering is only employed for speech and text modality. The corresponding methods are listed as follows:

- *TFIDF* [72]: It is short for term frequency-inverse document frequency, which uses statistics and intends to reflect the importance of a word in a specific text. The method has been proven to perform well across many NLP tasks.
- *Word embedding:* Another alternative feature extractor in NLP is word embedding with pre-trained neural networks [73], which maps the vocabulary to vectors of real numbers.

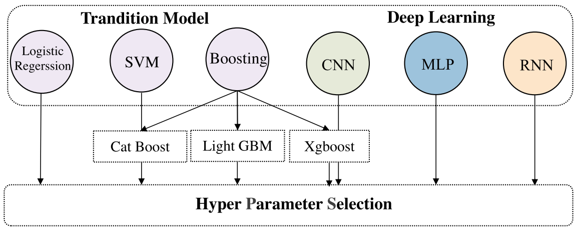


Fig. 3. The model selection search space. The selection process is also divided into two steps, first selecting the model to be used, and then choosing the corresponding hyper-parameters. Note that, during this process, the selection of each model is independent, which means that we may choose multiple models as an ensemble.

- *Spectrogram* [74]: We select two widely-used methods spectrogram [75] and melspectrogram [76] for speech modality. Spectrogram uses two convolution layers, which are initialized by Fourier transform kernels. Melspectrogram is an extended layer based on Spectrogram, which is obtained by multiplying the Spectrogram with a mel-scale conversion matrix trained from linear frequencies.
- *Feature scaling:* Feature scaling is used to normalize features, which contains four different scaling methods similar to data scaling.

3.4.3 Model Selection

Once data or features have been preprocessed, the next step is to find a model to predict the labels. Model generation and selection are usually considered as the most important part of AutoML. In this paper, we aim to automatically generate and select machine learning pipelines for different modalities and datasets under limited computation resource constraints, which means that those resource-consuming generation methods (such as neural architecture search [13], [77], [78]) are not within our consideration. Specifically, we fully investigated different machine learning models and selected several candidates with low computational complexity and good generalizability to construct our model selection search space. As illustrated in Fig. 3, we divide the search space into two parts, the traditional models (logistic regression [79], SVM [80], catboost [81], lightGBM [82] and xgboost [83]) and the deep models (Resnet 9/18 [3], R$(2+1)$D [63], Thin-Resnet [66], Text-CNN [84], GRU [85]). The selection process is also divided into two steps, where we first select the model to be used and then choose the corresponding hyper-parameters. Traditional learning component includes hyper-parameters such as regularization, category prior and loss functions. Similarly, deep learning component contains hyper-parameters such as pretrained datasets, freezing layers, network widths, and loss functions. Note that during this process, the selection of each model is independent, which means that we may choose multiple models as an ensemble.

3.4.4 Optimization

After model selection, the next component is model training, which involves the selection of a good optimization method. For classical learning models, optimization is not a crucial concern, since they usually have a default optimal optimization method. Besides, according to [86], different optimization methods may have similar performance. Therefore, in this paper, we only focus on the choice of the optimization methods that are applied to deep learning models.

For the optimization of deep learning models, gradient descend is one of the most popular choices which was employed by most state-of-the-art architectures. Here we select 3 widely-used gradient descend methods including Momentum [87], RMSprop [88] and Adaptive Moment Estimation (Adam) [8] to construct the search space.

3.4.5 AutoEnsemble

The last step in our search pipeline is ensemble, which employs multiple models to obtain better performance. As mentioned in Section 3.4.3, the model selection process will choose multiple candidate models for ensemble. To further expand the model diversity and enhance the performance, we propose two new methods for ensemble, which are listed as follows:

- *Instance Bagging* [23]: We found that different initialization for the same model can produce different predictions. Therefore, we train the same model multiple times to obtain more model instances.
- *Prediction Ensemble* [89]: We also found that the predictions for different training epochs of the same model are quite diverse, which means that we can save models with different training epoch to form the ensemble.

The ensemble process are directly adopted from [23]. Specifically, the dataset D is split into K-folds $\{(D^{(1)}_{\text{train}}, D^{(1)}_{\text{valid1}}, D^{(1)}_{\text{valid2}}), \ldots, (D^{(k)}_{\text{train}}, D^{(k)}_{\text{valid1}}, D^{(k)}_{\text{valid2}})\}$ to avoid overfitting. To simplify, we propose to represent each model as a parametrized function $f_\lambda(\theta)$ with parameters θ and hyperparameters λ. In each folds we use $D^{(i)}_{\text{train}}$ and $D^{(i)}_{\text{valid1}}$ to find the best parameters θ^* and λ for f, which are then evaluated on $D^{(i)}_{\text{valid2}}$ with predefined metrics M to obtain a score $score_{ij}$. $score_{ij}$ is then used to compute the corresponding normalized weight w_{ij} by $\frac{score_{ij}}{\sum_{i=1}^{k}\sum_{j=1}^{m} score_{ij}}$, which is employed in the final testing phase to ensure the quality of the ensemble.

4 Experiment

In the next three subsections, we first validate the effectiveness of the proposed search space and the searched solutions on typical datasets for five different modalities. Then, in Section 4.3, we perform ablation studies of the proposed method with different numbers of knowledge anchors and further compare with other widely-used black-box optimization methods such as random search. Finally, we validate the generalizability of the searched solutions for different modalities on the AutoDL challenge, where we won the first place in both the blind test[3] and the feedback phase.[4]

3. https://autodl.lri.fr/competitions/169
4. https://autodl.lri.fr/competitions/162

4.1 Experimental Settings

In this section, we describe the datasets and evaluation criteria used in our paper. We first introduce the searching datasets, evaluation metric, and implementation details of the search algorithm and the learning pipeline.

4.1.1 Dataset

As summarized in Table 4, the evaluation datasets contain 24 datasets from multiple domains and 5 modalities including image, video, speech, text and tabular. Datasets are collected and published by the organizers of the competition, and a small part of the datasets are sampled from other datasets, which are listed as follows:

- MNIST [68] has a training set of $60{,}000$ and a test set of $10{,}000$. The image size is 28×28 with 10 classes. All the intensities are either 1 or 0. The MNIST dataset is used in image classification with a 2D image and flattened to one dimension vector used as a tabular dataset.
- CIFAR10 [90] has $50{,}000$ training images and $10{,}000$ testing images. The color image size is 32×32 with 10 classes. All the color intensities of the images are normalized to $[-1, +1]$.
- CIFAR100 [90] is similar with the CIFAR10 dataset, except it has more classes (100).
- PA-100K [91] is constructed by images captured from outdoor surveillance cameras, which contains $100,000$ pedestrian images annotated with 26 attributes such as gender and age.
- NWPU VHR-10 [95] is constructed by images captured from Google Earth with very-high-resolution remote sensing, which contains 800 images with 10 classes.
- Ham10000 [92] is collected from different populations acquired dermatoscopic images. The dataset consists of $10,015$ dermatoscopic images with 7 classes where we split 8050 as the training set and the rest $1,965$ as the test set.
- KTH [93] is a human action recognition dataset that contains $2,391$ sequences. All sequences were taken over homogeneous backgrounds with a static camera with a 25fps frame rate. The sequences were downsampled to the spatial resolution of 160x120 pixels and have a length of four seconds on average. In this paper, we directly use the split provided by the AutoDL challenge.

4.1.2 Evaluation Metric

Area under the Learning Curve (ALC) was employed as the main evaluation criterion. The ALC metric is an official evaluation criterion used by the AutoCV challenge [59], which prefers machine learning pipelines with superior performance during the whole learning process. Specifically, for the prediction made at a timestamp, we compute the Area Under ROC Curve (AUC) [96] for every class, then average over all class as and normalize it through

$$NAUC = 2 \times AUC - 1.$$

Following [59], ALC is obtained through:

TABLE 4
Datasets Used in This Paper With Specific Modality, Size, Train Number, Test Number and Label Length, Details of Which are Also Available in [59]

Name	Dataset	Modality	Domain	Size	Train Num	Test Num	Label Length
Caucase	AutoCV2 [59]	Image	Object	$[1,-1,-1,3]$	24518	6089	257
Beatriz	AutoCV2 [59]	Image	People	$[1,350,350,3]$	4406	1094	15
Chucky	CIFAR-100 [90]	Image	Object	$[1,32,32,3]$	50000	10000	100
Pedro	PA-100K [91]	Image	People	$[1,-1,-1,3]$	80095	19905	26
Hammer	Ham10000 [92]	Image	Medical	$[1,400,300,3]$	8050	1965	7
Kreatur	KTH [93]	Video	Action	$[181,60,80,1]$	1528	863	4
Kraut	KTH [93]	Video	Action	$[181,120,160,1]$	1528	863	4
Katze	KTH [93]	Video	Action	$[181,120,160,1]$	1528	863	6
Monica1	AutoCV2 [59]	Video	Action	$[181,168,168,1]$	10380	2565	20
data01	AutoDL-Speech [59]	Speech	Speaker	$[-1,1,1,1]$	3000	3000	100
data02	AutoDL-Speech [59]	Speech	Emotion	$[-1,1,1,1]$	428	107	7
data03	AutoDL-Speech [59]	Speech	Accent	$[-1,1,1,1]$	796	200	3
data04	AutoDL-Speech [59]	Speech	Genre	$[-1,1,1,1]$	940	473	20
data05	AutoDL-Speech [59]	Speech	Language	$[-1,1,1,1]$	199	597	10
O1	AutoDL-Text [59]	Text	Comments	$[-1,1,1,1]$	7796	1817	2
O2	AutoDL-Text [59]	Text	Emotion	$[-1,1,1,1]$	11308	7538	20
O3	AutoDL-Text [59]	Text	News	$[-1,1,1,1]$	60000	40000	2
O4	AutoDL-Text [59]	Text	Spam	$[-1,1,1,1]$	54990	10010	10
O5	AutoDL-Text [59]	Text	News	$[-1,1,1,1]$	155952	72048	18
Adult	Adult [94]	Tabular	Census	$[1,1,24,1]$	39073	9768	3
Digits	MNIST [68]	Tabular	Hand Writing	$[1,1,1568,1]$	60000	10000	10
Dilbert	AutoDL-Tabulr [59]	Tabular	-	$[1,1,2000,1]$	14871	9709	5
Madeline	AutoDL-Tabulr [59]	Tabular	-	$[1,1,259,1]$	4222	3238	2

Specifically, each dataset is encoded as a 4-dimensional tensor, with sequence length, width, height and channel respectively. -1 in the 'Size' column denotes the corresponding value changes with different examples. We follow [59] to rename the datasets.

- Computing $NAUC$ and $s(t)$ at each timestamp t, where $s(t)$ is a step function w.r.t. t.
- Normalizing the time t between $[0,1]$ through

$$\hat{t}(t) = \frac{\log(1 + t/t_0)}{\log(1 + T/t_0)},$$

  where T is the overall time constraints and t_0 is the reference time amount.
- Computing the ALC using

$$ALC = \frac{1}{\log(1 + \frac{T}{t_0})} \int_0^T \frac{s(t)}{t + t_0} dt.$$

For a specific modality, the final score of a search pipeline is obtained by the average ALC on different datasets.

4.1.3 Implementation Details

Experiments are all conducted in an environment constructed following the challenge's official configurations: 4 vCPUs, with 26 GB of memory, CUDA 10 with drivers of cuDNN 7.5. In the search algorithm settings, the population size is 25, the generation number is 10 for the evolution algorithm. Every experiment is conducted with 4 repetitions and the corresponding means and standard deviations are reported in the corresponding tables or figures.

4.1.4 Search Space for ALC

In this section, we explain our search process under the ALC metric. The metric of ALC measures how well a pipeline could provide a fast and accurate prediction in a limited time. Therefore, we further expand the corresponding search space in the following experiments. More specifically, we add a new search hyperparameter, namely *num_of_pipeline*, to define how many pipelines need to be executed in the entire test process. In the search process, we first randomly decide the *num_of_pipeline* H, and then sample H pipelines defined in Section 3.1. These pipelines are executed in order and will be stopped when the time limit is reached or all pipelines are executed. In this way, the search space is largely expanded, but it usually ends up at solutions with a better ALC score. For instance, we found that all datasets tend to select multiple pipelines in the optimal solution, and the pipeline that in the earlier orders tend to select traditional machine learning or lightweight deep model for a fast prediction.

4.2 Performance Comparison on Different Modalities

The proposed method searches among a long list of operations described in Section 3.1 under the ALC criterion in Section 4.1.2. The extremely large search space makes the search more difficult, yet it also allows the proposed method to explore more possibilities and has a better opportunity to discover state-of-the-art pipelines. In this subsection, we analyze the optimal pipelines found by our method during the AutoDL challenge [59] for different modalities and datasets in detail.

Table 5 illustrates the optimal backbones and the corresponding pre-trained datasets for different modalities that we found by the proposed AutoML algorithm. Part of the results is counterintuitive. We found that there exist a group of similar pipelines that perform consistently better for a specific modality. For example, the widely-used Resnet [3] pre-trained on ImageNet [62] always has a superior performance

TABLE 5
The Searched Model Backbones, Pretrained Datasets for Multi-Modality Data

Modality	Non-neural	Neural	Pretrained
Image	-	ResNet9 ResNet18	ImageNet [62]
Video	-	MC3 [63]	Kinetics [64]
Speech	Logistic Regression	ThinResnet34[66]	VoxCeleb2 [67]
Text	SVM	TextCNN/RCNN GRU/GRUA	-
Tabular	LightGBM Xgboost Catboost	DNN	-

GRUA means GRU with Attention

TABLE 6
Image Task Results, Compared With the Previous AutoCV2 [59] First Solution

Dataset	Ours		AutoCV2	
	ALC	NAUC	ALC	NAUC
Caucase	$\mathbf{0.82 \pm 0.00}$	$\mathbf{0.96 \pm 0.00}$	0.78 ± 0.00	0.92 ± 0.00
Chucky	$\mathbf{0.87 \pm 0.00}$	$\mathbf{0.97 \pm 0.00}$	0.82 ± 0.00	0.92 ± 0.00
Pedro	$\mathbf{0.78 \pm 0.00}$	$\mathbf{0.93 \pm 0.00}$	0.77 ± 0.00	0.90 ± 0.00
Beatriz	$\mathbf{0.80 \pm 0.02}$	$\mathbf{0.85 \pm 0.07}$	0.65 ± 0.00	0.62 ± 0.06
Hammer	$\mathbf{0.85 \pm 0.01}$	$\mathbf{0.90 \pm 0.00}$	0.81 ± 0.01	0.88 ± 0.00

compared with other deep models on all 6 image datasets. This kind of generalizability explains why some classical deep models can still dominant most tasks, even though there are a lot of new models proposed subsequently. By summarizing and comparing different search results on different datasets for the same modality, we found several interesting insights, which are discussed in the following.

4.2.1 Image

For the image modality, we found that except for the searched backbones in Table 5, re-initializing all BN layer weights and bias of model backbone can further boost the ALC. Furthermore, in the later training phase, Resnet 9 without downsampling at the first two convolution layers can achieve more reliable performance. Other key hyperparameters, including splitting 0.1 of the dataset as a validation set, training with 32 batch size, 0.05 initial learning rate, and decay in every 30 steps, are validated to be the best solution for the image task.

By applying the found re-initialized tricks on the specific architectures, we can largely promote the test AUC of the first prediction, leading to a better ALC. Switching to ResNet9 in the latter training phase, we can obtain a more stable prediction in the ensemble phase. As shown in Table 6, this strategy is able to ensure better AUC and ALC on all datasets.

4.2.2 Video

We found that the 3D convolutional model (MC3 [63]) can obtain better performance for video. It uses the 3D convolution to extract the timing information. However, the disadvantage of the architecture lies in the computation speed of

TABLE 7
Video Task Results, Compared With the Previous AutoCV2 [59] First Solution

Dataset	Ours		AutoCV2	
	ALC	NAUC	ALC	NAUC
Kraut	$\mathbf{0.75 \pm 0.01}$	$\mathbf{0.79 \pm 0.00}$	0.68 ± 0.00	0.72 ± 0.01
Katze	$\mathbf{0.93 \pm 0.00}$	$\mathbf{0.97 \pm 0.00}$	0.90 ± 0.00	0.95 ± 0.00
Kreatur	0.72 ± 0.02	$\mathbf{0.76 \pm 0.01}$	$\mathbf{0.73 \pm 0.01}$	0.75 ± 0.01
Monica1	$\mathbf{0.65 \pm 0.02}$	$\mathbf{0.89 \pm 0.00}$	0.48 ± 0.00	0.84 ± 0.00

3D convolution, which is much slower than 2D convolution and could be harmful to ALC. Thus, we consider the optimal number of sampled frames in the data loading phase, where we found that 3 frames are the best choice for different video datasets. Specifically, all of our models are trained on 3,10 and 12th frames. The above models are further enhanced by auto-ensemble. In addition, we also find that freezing the first two layers of the model in the training stage will lead to a more stable state. Other hyperparameters including splitting 0.1 of the dataset for validation, training with 32 batch size, 30 steps per epoch, 0.03 initial learning rate are the best solution of the video task. As shown in Table 7, our solution gets superior results on most datasets. Specifically, our model outperforms AutoCV2 with a clear performance gap.

4.2.3 Speech

The main search results for speech data consist of a series of configs and parameters. Specifically, we set *n_mels* i.e., the number of mel filterbanks of Melspectrogram feature for speech signals to 30, the train/valid split ratio is 1.0, which promote the AUC score in most of the speech datasets, and alleviate overfitting in auto-ensemble. In the model search space, we found that *LogisticRegression* and *ThinResnet34* stably outperform the other models. As for the *ThinResnet34*, we further search two key parameters, the freezing layer number is set to 100 or 124 and the *warmup_epoch_num* is set to 8 or 14, which lead to a higher AUC score.

For implementation, we apply the recommended settings in PANNS's on the found pipeline. We further combine PANNS with the found pipeline and report the TAUC score to explore the single model performance. In our experiments, all the models are trained on the offline datasets with 10 epochs because of the limited time budget.

We compare the found pipeline with the previous AutoSpeech Challenge,[5] denote as "Baseline3". It is an ensemble model with LogisticRegression, LSTM, CRNN, and BiLSTM. The summarized results for our pipeline on speech modality are presented in Table 8. All the deep models are pretrained on VoxCeleb2 dataset [67], the comparisons of NAUC scores show that our model outperforms baseline3 in all datasets. Notably, for data01 and data05 the AUC is improved to 1 within a short time.

4.2.4 Text

The parameters searched for text tasks include train/val set split ratio, batch size, maximum word sequence length, and

5. https://autodl.lri.fr/competitions/48

TABLE 8
Speech Task Results, Compared With the AutoSpeech Winners' Solutions on Data01 to Data05

Dataset	Ours		AutoSpeech	
	ALC	NAUC	ALC	NAUC
data01	$\mathbf{0.70 \pm 0.01}$	$\mathbf{0.99 \pm 0.00}$	0.55 ± 0.00	0.97 ± 0.01
data02	$\mathbf{0.93 \pm 0.01}$	$\mathbf{0.98 \pm 0.00}$	0.89 ± 0.01	0.92 ± 0.00
data03	$\mathbf{0.69 \pm 0.02}$	$\mathbf{0.85 \pm 0.03}$	0.60 ± 0.01	0.85 ± 0.03
data04	$\mathbf{0.76 \pm 0.01}$	$\mathbf{0.82 \pm 0.01}$	0.61 ± 0.00	0.69 ± 0.02
data05	$\mathbf{0.92 \pm 0.02}$	$\mathbf{1.0 \pm 0.00}$	0.90 ± 0.00	0.99 ± 0.00

TABLE 9
Text Task Results, Compared With AutoNLP First Solution (DeepBlue)

Dataset	Ours		AutoNLP	
	ALC	NAUC	ALC	NAUC
O1	$\mathbf{0.89 \pm 0.01}$	$\mathbf{0.95 \pm 0.00}$	0.80 ± 0.00	0.81 ± 0.01
O2	$\mathbf{0.92 \pm 0.00}$	$\mathbf{0.978 \pm 0.00}$	0.84 ± 0.00	0.974 ± 0.00
O3	$\mathbf{0.71 \pm 0.01}$	$\mathbf{0.875 \pm 0.02}$	0.60 ± 0.03	0.875 ± 0.02
O4	$\mathbf{0.86 \pm 0.02}$	$\mathbf{0.999 \pm 0.00}$	0.69 ± 0.00	0.994 ± 0.00
O5	$\mathbf{0.90 \pm 0.00}$	$\mathbf{0.96 \pm 0.00}$	0.62 ± 0.00	0.94 ± 0.00
Tanak	$\mathbf{0.90 \pm 0.00}$	$\mathbf{0.95 \pm 0.00}$	0.78 ± 0.00	0.92 ± 0.00

Dataset O1-O5 are belong to the offline datasets, Tanak is the feedback dataset.

vocab size, where the optimal values are 0.2, 32, 301, and 20,000, respectively. For feature engineering space, we found that a random initialization of 300-dimensional word embedding is efficient within a small time stamp, which is essential for the improvement of ALC. We also found that fastText[6] performs best in terms of AUC, but it is also time-consuming. Therefore, we consider both random_init_emb_300 word embedding and fastText for feature selection. As for model selection, we found that TextCNN [84] is fast and increasing ALC quickly on both short text datasets and long text datasets. Besides, GRU[85] is computationally expensive yet able to capture sentence order and comprehensive semantic representations. Therefore, at the beginning of the pipeline, we first use textCNN for fast training and then use GRU for further AUC improvement. For optimizer and learning rate, we found that the optimal optimizer for TextCNN based models is RMSprop, and utilize a linear decay strategy with an initial learning rate of 0.1. The optimal optimizer for GRU based models is Adam with an exponential decay strategy and an initial learning rate of 0.016. Moreover, we include the order of our models as a search pipeline, which is reported in Table 9. We further search for the ensemble strategy as described in Section 3.4.5 to find an efficient way to boost the ALC score. We also found that the ensemble prediction based on a weighted linear method is quite effective. Table 9 shows the experiment results on 5 datasets. We compare our method with AutoNLP[7] 1'st solution (DeepBlue). We achieve the best ALC in all datasets, outperforming DeepBlue by 9.84, 1.68, 8.30, 7.31, 9.16 percent, respectively.

TABLE 10
Tabular Task Results, Compared With the DNN Model Released by fase.ai With a Default Hyperparams Setting

Dataset	Ours		DNN	
	ALC	NAUC	ALC	NAUC
Adult	$\mathbf{0.75 \pm 0.00}$	$\mathbf{0.76 \pm 0.00}$	0.51 ± 0.00	0.73 ± 0.00
Dilbert	$\mathbf{0.98 \pm 0.00}$	$\mathbf{0.999 \pm 0.00}$	0.85 ± 0.00	0.999 ± 0.00
Digits	$\mathbf{0.97 \pm 0.00}$	$\mathbf{0.998 \pm 0.00}$	0.80 ± 0.03	0.993 ± 0.00
Madeline	$\mathbf{0.88 \pm 0.00}$	$\mathbf{0.89 \pm 0.00}$	0.21 ± 0.00	0.272 ± 0.01

4.2.5 Tabular

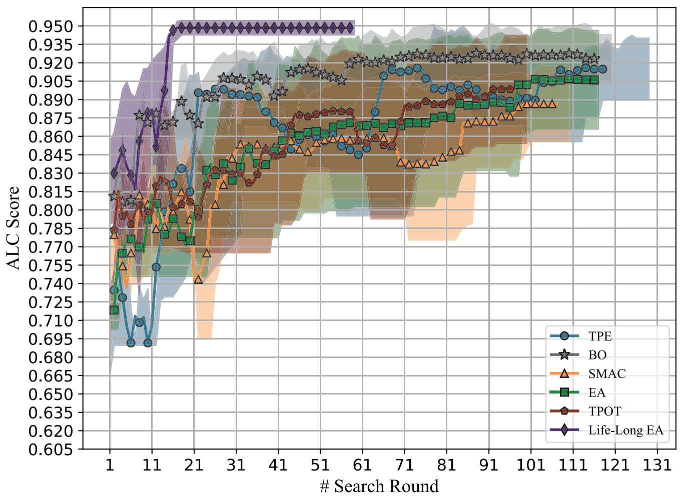
For the tabular modality, we found that boosting methods LightGBM [82], Catboost [81] and Xgboost [83] are fast and effective in the regime of limited training data, little training time and little expertise for parameter tuning. We further search a set of hyperparameters, including max tree depth, number of leaves and learning rates *etc*. We found that unlimited depth with a relatively small number of leaves is more suitable for leaf-wise tree growth, which can achieve good performance and prevents over-fitting. Another useful model on tabular datasets is Multilayer Perceptron (MLP), which automatically learns the classifier for the given features. We compare the found pipeline with two widely-used baselines. The first is an MLP model, which automatically learns a classifier. The second is LightGBM with default hyper-parameter settings. As illustrated in Table 10, we can get superior AUC and ALC on all tabular datasets.

4.3 Ablation Study

In this section, we use two text datasets ChnSentiCorp[8] and toutiao_news_data [97] to illustrate the importance of the evolution algorithm and knowledge anchor.

Search Algorithms and Search Spaces. We first compare the evolution algorithm with widely-used random search [11] and Bayesian optimization (BO) in Fig. 5. The results of random search are obtained from 250 randomly sampled pipelines. As demonstrated in Fig. 5, our method can find a superior pipeline after a few iterations, and it is more stable than other baselines. We attribute the efficiency to meta-learning and the effectiveness of the evolutionary algorithm. Fig. 5 also demonstrate the EA is indispensable in our method. Specifically, as shown in Fig. 5, the vanilla EA does not work well on the proposed search space, even worse performance than the widely-used Bayesian optimization (BO). However, after combining with the life-long knowledge anchors, our method converges quickly and finds a better solution in the search space. Notably, in Fig. 5, we also combine our life-long knowledge anchors with BO, which shows a worse performance compared to life-long EA. Overall, when combined with the life-long knowledge anchors, the evolutionary algorithm achieves better exploration and exploitation than other algorithms. Therefore, we integrate EA with our life-long knowledge anchors. To further demonstrate the effectiveness of the proposed method, we conducted ablation experiments on the search space and search algorithm for the proposed method. As we mentioned in Section 1 and illustrated in Fig. 4 (left), previous

6. https://github.com/facebookresearch/fastText/
7. https://www.4paradigm.com/competition/autoNLP2019
8. https://github.com/SophonPlus/ChineseNlpCorpus



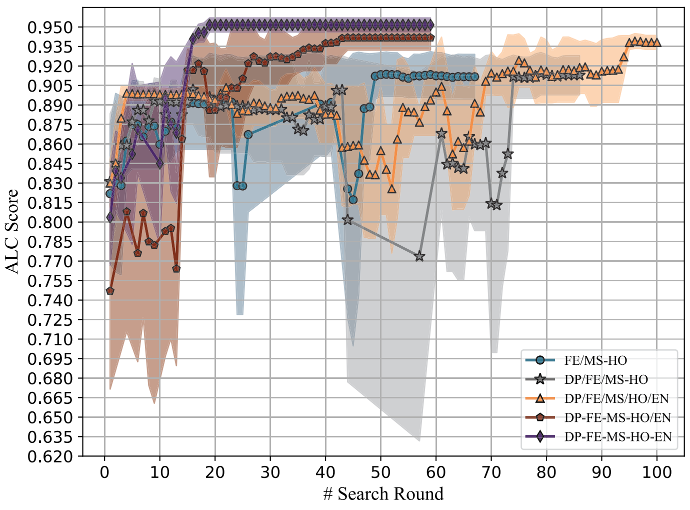


Fig. 4. Mean and standard deviation of the performance in the different algorithms (left) and search spaces (right) on toutiao_news_data. For the left figure, all the search algorithms are compared by employing the proposed joint search space. For the right figure, all the search spaces are compared under the proposed algorithm. The abbreviations have the same meaning as in Table 1. Similarly, '/' and '-' denote two components are searched integrally and hierarchically, respectively.

AutoML frameworks employ isolated and incomplete machine learning components to construct search spaces, which largely limited the search scope thus led to a worse ALC score. At the same time, the learning pipelines found in our search space always show a superior and stale ALC score, which demonstrates the superiority of the proposed search space. As a baseline search algorithm, we compare the ALC score of the proposed method to several widely used search algorithms including TPE [98], TPOT [17], SMAC [12], vanilla evolution algorithm and Bayesian optimization [23]. Fig. 4 (right) shows that the proposed Life-Long EA performs statistically significantly better in terms of search effectiveness and efficiency under the same search space.

Ratio of Knowledge Anchor. As mentioned in Section 3.3, a knowledge anchor is used to accelerate the searching process. We compare different ratios of knowledge anchor deployed in our search algorithm to demonstrate the effectiveness in Fig. 6.

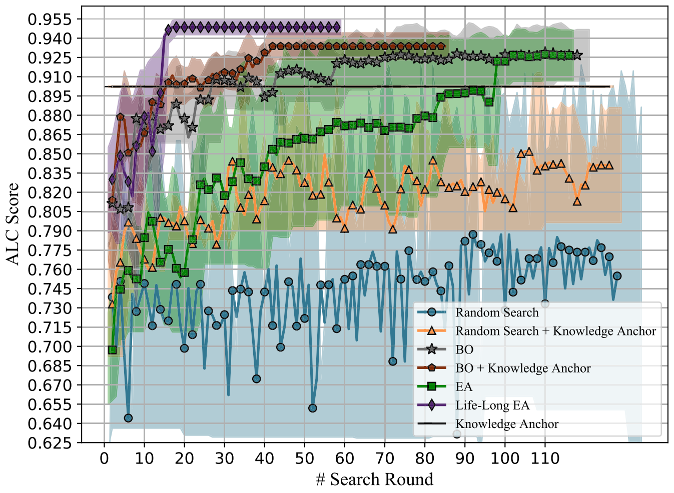


Fig. 5. The ablation study of the proposed method with random search, evolution algorithm (EA) and Bayesian optimization (BO) on toutiao_news_data. "knowledge anchor" denote that we directly employ the sampled knowledge anchors as the final pipeline to obtain the ALC score.

The knowledge anchor is obtained from O4 (described in Table 4), then we randomly replace the individuals with the knowledge anchor in the first generation of the evolution algorithm. As illustrated in Fig. 6, the search time drops quickly with more knowledge anchors appear in the first generation. For ChnSentiCorp dataset, the ALC only drops 0.9 percent but with 128× faster with a knowledge anchor ratio 1.0. For toutiao_news_data, the ALC decreases significantly when the ratio of knowledge anchor is higher than 0.5, which means that a high knowledge anchor ratio also brings overfitting. However, the ALC of toutiao_news_data is comparable when the ratio of the designed knowledge anchor is 0.3, and the search time significantly drops from 2,640s to 1,140s.

Sensitivity of Dataset Similarity. We further investigate the influence of knowledge anchors found on different datasets. Specifically, we select two different dataset O4 and O5 to find knowledge anchors, which is then applied on ChnSentiCorp. The similarity between O4 and ChnSentiCorp is 0.861 and the similarity between O5 and ChnSentiCorp is 0.775. The result is summarized in Fig. 7, knowledge anchors found on more similar datasets tend to be more efficient. In particular, knowledge anchors found on O4 brings a significant speed-up with a tiny performance drop. In

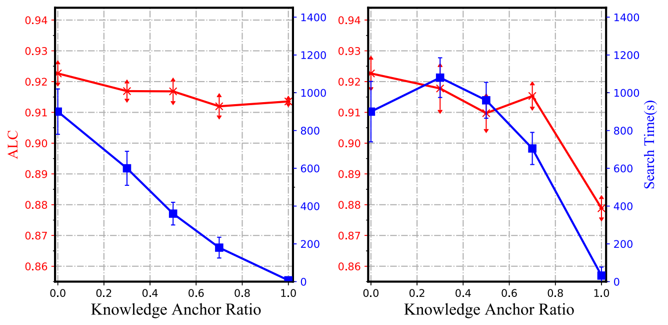


Fig. 6. ALC and search time (s) on ChnSentiCorp (left) and toutiao_news_data (right) with different ratio of knowledge anchor, which is found on O4.

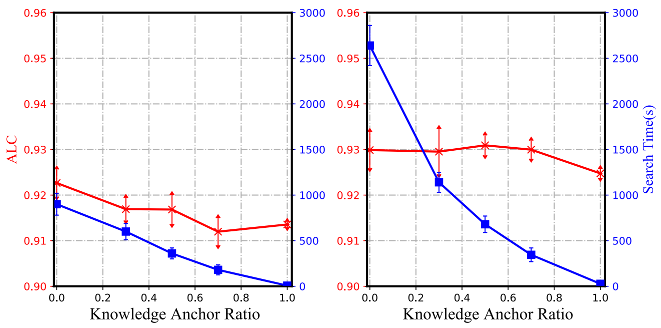


Fig. 7. ALC and search time (s) with different knowledge anchors on O4 (left) and O5 (right). The similarity between O4 and ChnSentiCorp is 0.861 while the similarity between O5 and ChnSentiCorp is 0.775.

TABLE 11
Search Cost, ALC and AUC Scores With Different Meta-Features

Meta-feature	prob Sampling	Search Cost ($\times 10^3$)	ALC	AUC
S, I-T, Stat [23]	$\times$	2.0 ± 0.4	0.943 ± 0.0	0.972 ± 0.0
S	$\times$	2.4 ± 0.4	0.95 ± 0.0	0.974 ± 0.0
S, Meta	$\times$	1.7 ± 0.1	0.951 ± 0.0	0.975 ± 0.0
S, Meta	✓	$\mathbf{0.8 \pm 0.1}$	$\mathbf{0.954 \pm 0.0}$	$\mathbf{0.976 \pm 0.0}$

The first column denotes the type of meta-features employed in the proposed method, where 'S' denotes simple feature, 'I-T' denotes information-theoretic features, 'Stat' denotes statistical meta-features, and 'Meta' denotes the feature extract from meta models.

contrast, knowledge anchors found on O5 will lead to a significant performance drop on ChnSentiCorp.

Comparision of Different Meta-Features. We now evaluate the effectiveness of the proposed meta-feature and probabilistic sampling described in Section 3.3. To demonstrate the effect of the proposed meta-learning scheme, we compare different meta-features and sampling methods in terms of ALC, AUC and search cost. Even without probabilistic sampling, the proposed meta-feature can find more similar pipelines and thus lead to a better performance with less search cost in Table 11. To complement the above meta-feature analysis, we also compare the meta-feature with and without probabilistic sampling in Table 11. As reported in Table 11, the proposed sampling method shows the statistical speed-up with a higher ALC and AUC, which confirms the previous statement that without probabilistic sampling may lead to overfitting thus inefficient in the search process.

4.4 AutoDL Challenge

In this section, we present the evaluation of the found pipelines on the AutoDL 2019 challenge, which is organized by Chalearn, Google, and 4Paradigm. The goal of the AutoDL challenge is to introduce universal learning machines that make algorithms without any human intervention. For the participating teams, all the datasets are invisible. The participating teams submit codes according to the interface provided by the challenge, and then automatically run on different datasets and modalities. Specifically, the challenge is divided into two phases, the feedback phase and the blind test phase: In the feedback stage, participating teams are allowed to submit the code and see their running results. In the blind test phase, the organizer directly ran the code on 10 new datasets, the final ranking of each team was obtained using the average rank over the 10 test datasets. In our paper, we directly use the ALC score as the optimization goal. The found machine learning pipelines are tested and selected in the feedback stage, where the optimal pipelines for each modality are directly submitted to AutoDL challenge.

Fig. 8 shows the comparison of ALC and Rank of different teams. The found pipelines show a superior and stable performance comparing with other teams. For example, the models generated by DeepBlueAI show decent performance on Video and Text, but has the worst ALC score in Speech, which means that their solution may overfit on specific

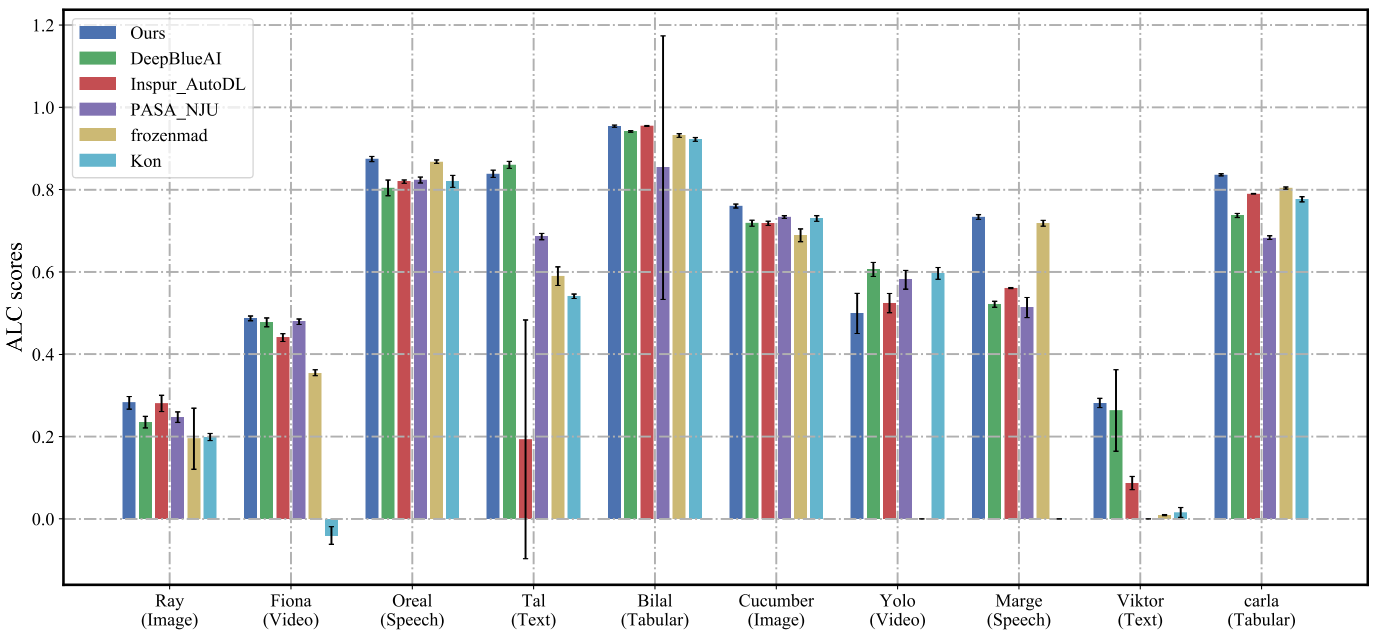


Fig. 8. Mean and standard deviation ALC score accorss ten datasets in AutoDL challenge [59].

TABLE 12
The Mean and std of ALC and AUC Scores for Different Key Hyperparameters

Modality/ Dataset	Key Hyperparameters	ALC Score	AUC Score
Image/Apollon	Baseline (AutoCV2)	0.64 ± 0.03	0.84 ± 0.05
	+ BN re init	0.64 ± 0.09	0.86 ± 0.02
	+ Resnet 9 (later stage)	$\mathbf{0.7 \pm 0.05}$	$\mathbf{0.88 \pm 0.001}$
Video/Kreatur	Baseline (AutoCV2)	0.70 ± 0.03	0.73 ± 0.03
	+ Model: MC3	0.72 ± 0.03	0.75 ± 0.04
	+ Frame Select [3, 10, 12]	$\mathbf{0.75 \pm 0.01}$	$\mathbf{0.77 \pm 0.01}$
Text/O5	Baseline (AutoNLP)	0.75 ± 0.04	0.95 ± 0.005
	+ GRU	0.81 ± 0.02	0.96 ± 0.003
	+ data sampling (0.5)	$\mathbf{0.90 \pm 0.01}$	$\mathbf{0.97 \pm 0.002}$
Speech/Data01	Baseline (AutoSpeech)	0.55 ± 0.00	0.97 ± 0.00
	+ Model: ThinResnet34	0.60 ± 0.00	1.0 ± 0.00
	+ Data sampling	$\mathbf{0.73 \pm 0.01}$	$\mathbf{1.0 \pm 0.00}$
Tabular/Adult	Baseline (DNN)	0.52 ± 0.01	0.82 ± 0.01
	+ Data sampling	0.58 ± 0.02	0.82 ± 0.01
	+ Tree Model (LGB)	0.75 ± 0.001	0.87 ± 0.001
	+ Ensemble	$\mathbf{0.76 \pm 0.002}$	$\mathbf{0.88 \pm 0.001}$

datasets or modalities. On the other hand, there is a significant difference and stable rank in terms of dataset rank and ALC, which proves the effectiveness of the found pipelines. We also note that there is no significant performance on the Yolo dataset, which could be caused by the high diversity of data since the method of frozenmad also crashed on it.

In addition to employing the found pipelines for different modalities, we also made some efforts on the engineering tricks that make further improvement. We summarize the key engineering tricks as follows: (i). Moving dataset to RAM and incremental learning. During the AutoDL competition, we found that an important reason for the low ALC is the slow IO of the dataset, which severely delays the prediction. Therefore, in practice, when the dataset is small, we directly load the entire dataset into RAM. On the contrary, we incrementally sample a small part of the dataset for each model so that they can quickly learn and predict the corresponding dataset. (ii). GPU acceleration. We have written the corresponding GPU acceleration code for every corresponding machine learning component which can do more prediction in a limited time constraint. (iii). Caching feature, model, and prediction. To achieve better acceleration, we cache all the previous outputs including processed data, features and models. In this way, when the following pipeline shares the same component, we can skip this part and run different parts directly. For example, if two pipelines share the same operations and hyperparameters in feature engineering, we directly employ the features that have been processed to achieve better acceleration.

We also summarize the key hyperparameters of different modalities in Table 12. Interestingly, we found that some simple and oblivious components show a great impact on the final performance. For example, in the video modality, the simple frame selection on 3, 10, and 12 improves the ALC score from 0.72 to 0.75. This indicates that the integrated learning search space proposed in our paper may bring a new aspect of the AutoML. It is worth noting that most previous works are only focusing on new algorithm design.

5 Conclusion

In this paper, we presented another ambitious goal of AutoML: a new paradigm for fully automated discovery of the whole ML pipeline, which largely reduces the human intervention. As a start, we demonstrated the potential of this paradigm by constructing a novel evolutionary algorithm with lifelong knowledge anchors. Our method can significantly reduce the computational cost while achieving state-of-the-art performances on 5 different modalities. Furthermore, the searched pipelines can also generalize well to other datasets, based on which we won 1st place in both the feedback and final phases of the 2019 AutoDL challenge.

Acknowledgments

This work was supported in part by the National Science Fund for Distinguished Young Scholars under Grant 62025603, in part by the National Natural Science Foundation of China under Grants U1705262, 62072386, 62072387, 62072389, 62002305, 61772443, 61802324, and 61702136, and in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2019B1515120049. Xiawu Zheng, Yang Zhang, Sirui Hong, Huixia Li, Lang Tang, and Youcheng Xiong contributed equally to this work.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Inf. Process. Syst.*, vol. 60, no. 6, pp. 84–90, May 2017.
[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations*, 2015.
[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[4] G. Huang, Z. Liu, L. Van Der Maaten , and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE Conf. Comput Vis. Pattern Recogni.*, 2017, pp. 2261–2269.
[5] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7132–7141.
[6] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 113–123.
[7] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, pp. 91–110, 2004.
[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, 2015.
[9] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *Comput. Vis. Pattern Recognit.*, 2017, *arXiv:1704.04861*.
[10] Q. Yao *et al.*, "Taking human out of learning applications: A survey on automated machine learning," *Artif. Intell.*, 2018, *arXiv:1810.13306*.
[11] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.
[12] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown , "Auto-weka: Combined selection and hyperparameter optimization of classification algorithms," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2013, pp. 847–855.
[13] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017.
[14] A. M. Alaa and M. van der Schaar, "Autoprognosis: Automated clinical prognostic modeling via Bayesian optimization with structured kernel learning," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 139–148.

[15] Y. Gil *et al.*, "P4ML: A phased performance-based pipeline planner for automated machine learning," in *Proc. Int. Conf. Mach. Learn. Workshop*, 2018, pp. 1–8.

[16] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[17] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore, "Evaluation of a tree-based pipeline optimization tool for automating data science," in *Proc. Genet. Evol. Comput. Conf.*, 2016, pp. 485–492.

[18] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 4780–4789.

[19] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," in *Proc. 35th Conf. Uncertainty Artif. Intell.*, 2019, pp. 367–377.

[20] T. Back, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic algorithms*. New York, NY, USA: Oxford University Press, 1996.

[21] H. J. Escalante, M. Montes, and L. E. Sucar, "Particle swarm model selection," *J. Mach. Learn. Res.*, vol. 10, no. 2, pp. 405–440, 2009.

[22] B. Komer, J. Bergstra, and C. Eliasmith, "Hyperopt-Sklearn: automatic hyperparameter configuration for Scikit-learn," in *Proc. 13th Python Sci. Conf.*, 2014, pp. 32–37.

[23] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 2755–2763.

[24] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 2951–2959.

[25] B. Chen, H. Wu, W. Mo, I. Chattopadhyay, and H. Lipson, "Autostacker: A compositional evolutionary learning system," in *Proc. Genet. Evol. Comput. Conf.*, 2018, pp. 402–409.

[26] M. Kim and L. Rigazio, "Deep clustered convolutional kernels," in *Proc. 1st Int. Workshop Feature Extraction, Modern Questions Challenges*, 2015, pp. 160–172.

[27] A. Gaier and D. Ha, "Weight agnostic neural networks," in *Proc. Neural Inf. Process. Syst.*, 2019, pp. 5365–5379.

[28] D. S. Park *et al.*, "Specaugment: A simple data augmentation method for automatic speech recognition," in *Proc. Int. Speech Commun. Assoc.*, 2019, pp. 2613–2617.

[29] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing Systems 2*. Pittsburgh, PA, USA: School of Computer Science, Carnegie Mellon University, 1990, pp. 524–532.

[30] P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 54–65, Jan. 1994.

[31] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv:1308.3432*.

[32] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, "Designing neural networks through neuroevolution," *Nat. Mach. Intell.*, vol. 1, pp. 24–35, 2019.

[33] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Representations*, 2019.

[34] S. Thrun, "A lifelong learning perspective for mobile robot control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 1994, pp. 23–30.

[35] P. Ruvolo and E. Eaton, "ELLA: An efficient lifelong learning algorithm," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 507–515.

[36] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks (NN)*, vol. 113, pp. 54–71, May 2019.

[37] R. Ranjan, V. M. Patel, and R. Chellappa, "Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 1, pp. 121–135, Jan. 2019.

[38] A. A. Rusu *et al.*, "Progressive neural networks," 2016, *arXiv:1606.04671*.

[39] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "iCaRL: Incremental classifier and representation learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5533–5542.

[40] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6470–6479.

[41] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.

[42] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *Proc. 34thInt. Conf. Mach. Learn.*, 2017, pp. 3987–3995.

[43] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 144–161.

[44] C. Fernando *et al.*, "Pathnet: Evolution channels gradient descent in super neural networks," 2017, *arXiv:1701.08734*.

[45] J. Serrà, D. Suris, M. Miron, and A. Karatzoglou, "Overcoming catastrophic forgetting with hard attention to the task," in *Proc 35th Int. Conf. Mach. Learn.*, 2018, pp. 4548–4557.

[46] W. Hu *et al.*, "Overcoming catastrophic forgetting for continual learning via model adaptation," in *Proc. Int. Conf. Learn. Representations*, 2018.

[47] L. Chen *et al.*, "SCA-CNN: Spatial and channel-wise attention in convolutional networks for image captioning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6298–6306.

[48] Y. Gao, J. Ma, M. Zhao, W. Liu, and A. L. Yuille, "NDDR-CNN: Layerwise feature fusing in multi-task CNNs by neural discriminative dimensionality reduction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3200–3209.

[49] Y. Lu, A. Kumar, S. Zhai, Y. Cheng, T. Javidi, and R. Feris, "Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1131–1140.

[50] X. Jia, B. De Brabandere , T. Tuytelaars, and L. V. Gool, "Dynamic filter networks," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 667–675.

[51] H. Liu, J. Lu, J. Feng, and J. Zhou, "Learning deep sharable and structural detectors for face alignment," *IEEE Trans. Image Process.*, vol. 26, no. 4, pp. 1666–1678, Apr. 2017.

[52] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," in *Proc. Int. Conf. Learn. Representations*, 2017.

[53] J. Xu and Z. Zhu, "Reinforced continual learning," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 907–916.

[54] P. Brazdil, C. G. Carrier, C. Soares, and R. Vilalta, *Metalearning: Applications to Data Mining*. Berlin, Germany: Springer, 2009.

[55] M. Reif, F. Shafait, and A. Dengel, "Meta-learning for evolutionary parameter optimization of classifiers," *Mach. Learn.*, vol. 87, pp. 357–380, 2012.

[56] M. Feurer, J. T. Springenberg, and F. Hutter, "Initializing Bayesian hyperparameter optimization via meta-learning," in *Proc. Twenty-Ninth AAAI Conf. Artif. Intell.*, 2015, pp. 1128–1135.

[57] R. Engels and C. Theusinger, "Using a data metric for preprocessing advice for data mining applications," in *Proc. 13th Eur. Conf. Artif. Intell.*, 1998, pp. 430–434.

[58] J. Gama and P. Brazdil, "Characterization of classification algorithms," in *Proc. Portuguese Conf. Artif. Intell.*, 1995, pp. 189–200.

[59] Z. Liu *et al.*, "AutoCV challenge design and baseline results," in *Proc. Conf. sur l'Apprentissage Automatique*, 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02265053

[60] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," ETH Züürich, Zürich, Switzerland, TIK-Rep 103, 2001.

[61] E. Real, C. Liang, D. R. So, and Q. V. Le, "AutoML-Zero: Evolving machine learning algorithms from scratch," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 8007–8019.

[62] O. Russakovsky *et al.*, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, pp. 211–252, Apr. 2015.

[63] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6450–6459.

[64] W. Kay *et al.*, "The kinetics human action video dataset," 2017, *arXiv:1705.06950*.

[65] Y. Liu *et al.*, "RoBERTa: A robustly optimized bert pretraining approach," 2019, *arXiv:1907.11692*.

[66] W. Xie, A. Nagrani, J. S. Chung, and A. Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.* 2019, pp. 5791–5795.

[67] J. S. Chung, A. Nagrani, and A. Zisserman, "VoxCeleb2: Deep speaker recognition," in *Proc. Int. Speech Commun. Assoc.*, 2018, pp. 1086–1090.

[68] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[69] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit.*, 2019, pp. 113–123.

[70] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim, "Fast autoaugment," in *Proc. Neural Inf. Process. Syst.*, 2019, pp. 6662–6672.

[71] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, 2005, pp. 886–893.

[72] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*. New York, NY, USA: University Press, 2011.

[73] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol.*, 2018, pp. 4171–4186.

[74] K. Choi, D. Joo, and J. Kim, "Kapre: On-GPU audio preprocessing layers for a quick implementation of deep neural network models with Keras," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017.

[75] L. Wyse, "Audio spectrogram representations for processing with convolutional neural networks," *Proc. 1st Int. Workshop Deep Learn. Music Joint with IJCNN*, 2017, pp. 37–41.

[76] Y. Hwang, H. Cho, H. Yang, I. Oh, and S.-W. Lee, "Mel-spectrogram augmentation for sequence to sequence voice conversion," 2020, *arXiv:2001.01401*.

[77] X. Zheng, R. Ji, L. Tang, B. Zhang, J. Liu, and Q. Tian, "Multinomial distribution learning for effective neural architecture search," in *Proc. Int. Conf. Comput. Int. Conf. Comput. Vis.*, 2019, pp. 1304–1313.

[78] X. Zheng *et al.*, "Dynamic distribution pruning for efficient network architecture search," 2019, *arXiv:1905.13543*.

[79] D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein, and M. Klein, *Logistic regression*. New York, NY, USA: Springer, 2002.

[80] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, pp. 293–300, Jun. 1999.

[81] A. V. Dorogush, V. Ershov, and A. Gulin, "Catboost: Gradient boosting with categorical features support," in *Proc. 32nd Conf. Neural Inf. Process. Syst.*, 2018.

[82] G. Ke *et al.*"LightGBM: A highly efficient gradient boosting decision tree," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 3149–3157.

[83] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 785–794.

[84] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1746–1751.

[85] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Proc. NIPS Workshop Deep Learn.*, 2014.

[86] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 161–168.

[87] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Netw.*, vol. 12, no. 1, pp. 145–151, Jan. 1999.

[88] Y. N. Dauphin, H. de Vries, and Y. Bengio, "Equilibrated adaptive learning rates for non-convex optimization," in *Proc. Adv. Conf. Neural Inf. Process. Syst.*, 2015, pp. 1504–1512.

[89] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get M for free," in *Proc. Int. Conf. Learn. Representations*, 2017.

[90] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.222.9220&rep=rep1&type=pdf

[91] X. Liu *et al.*, "Hydraplus-net: Attentive deep features for pedestrian analysis," in *Proc. Int. Conf. Comput. Vis.*, 2017, pp. 350–359.

[92] P. Tschandl, C. Rosendahl, and H. Kittler, "The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," *Sci. Data*, vol. 5, Aug. 2018, Art. no. 180161.

[93] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local SVM approach," in *Proc. 13th Int. Conf. Pattern Recognit.*, 2004, pp. 32–36.

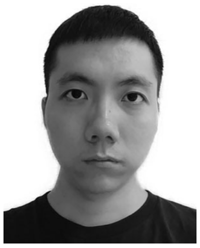
[94] R. Kohavi, "Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid." in *Proc. 2nd Int. Conf. Knowl. Discov. Data Mining*, 1996, pp. 202–207.

[95] G. Cheng, J. Han, P. Zhou, and L. Guo, "Multi-class geospatial object detection and geographic image classification based on collection of part detectors," *J. Photogrammetry. Remote Sensing*, vol. 98, pp. 119–132, Dec. 2014.

[96] I. Guyon, G. C. Cawley, G. Dror, and V. Lemaire, "Results of the active learning challenge," in *Proc. Active Learn. Exp. Des. Workshop Conjunction AISTATS*, 2011, pp. 19–45.

[97] L. Xu *et al.*, "Clue: A Chinese language understanding evaluation benchmark," in *Proc. 28th Int. Conf. Comput. Linguistics*, 2020, pp. 4762–4772.

[98] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proc. 24th Int. Conf. Neural Inf. Process. Syst.*, 2011, pp. 2546–2554.



Xiawu Zheng received the MS degree in computer science in 2018, from the School of Information Science and Engineering, Xiamen University, Xiamen, China, where he is currently working toward the Phd degree. His research interests include computer vision and machine learning. He was involved in automated machine learning.

Yang Zhang received the BSc degree in computer science from the Huazhong University of Science and Technology, China, in 2013, and the MSc degree in computer applications from Peking University Shenzhen Graduate School, China, in 2016. He is currently a research engineer at DeepWisdom. His research interests include information retrieval, recommendation system, and natural language processing.

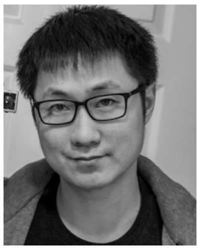
Sirui Hong received the MS degree in telecommunication from the School of Engineering, The Hong Kong University of Science and Technology (HKUST), Hong Kong SAR, China, in 2014. From 2014 to 2016, she was a research assistant with HKUST. She is currently a senior researcher with Deep Wisdom. Her research interests include natural language processing, information retrieval, and machine learning.

Huixia Li is currently working toward the MS degree with the School of Information Science and Engineering, Xiamen University. Her research interests include computer vision and machine learning. She was involved in neural network compression and acceleration.

Lang Tang is currently working toward the MS degree with the School of Information Science and Engineering, Xiamen University. He was an intern with Microsoft Research Asia. His research interests include computer vision and machine learning. He was involved in automated machine learning.

Youcheng Xiong received the MS degree in physics from the School of Physical Science and Technology, Xiamen University, Xiamen, China, in 2019. He is currently a researcher at Deep Wisdom. His research interests include time series forecasting, machine learning, and meta learning. He was involved in automated machine learning.

Jin Zhou received the BSc degree in computer science from the School of Information Science and Engineering, Hangzhou Dianzi University, Hangzhou, China, in 2017. He is currently a researcher at Deep Wisdom. His research interests include recommendation system and machine learning.



Yan Wang received the PhD degree in electrical engineering from Columbia University. He was a software engineer with Search, Pinterest. He authored or coauthored more than 20 papers on top international conferences and journals and holds ten US or international patents. His research interests include deep learning's applications on multimedia retrieval.

Xiaoshuai Sun (Member, IEEE) received the BS degree in computer science from Harbin Engineering University, in 2007, and the MS and PhD degrees in computer science and technology from the Harbin Institute of Technology, in 2009 and 2015, respectively. He is currently an associate professor at the School of Informatics, Xiamen University, China. His research interests include computer vision, pattern recognition, and multimedia analysis. He was the recipient of the Best Ph.D. Thesis Award in Harbin Institute of Technology (2015) and also a winner of Microsoft Research Fellowship (2011). He holds three authorized patents and has authored more than 80 papers in referred journals and conferences, including the *IEEE Transactions on Image Processing*, ACM Multimedia, IEEE CVPR, ICCV, NeurIPS, IJCAI, and AAAI.

Pengfei Zhu received the PhD degree from The Hong Kong Polytechnic University, Hong Kong, China, in 2015, and the BS and MS degrees from the Harbin Institute of Technology, Harbin, China, in 2009 and 2011, respectively. He is currently an associate professor at the College of Intelligence and Computing, Tianjin University. His research interests include machine learning and computer vision.

Chenglin Wu received the BSc degree in computer science from the School of Information Science and Engineering, Xiamen University, Xiamen, China, in 2012. He is currently a CEO of DeepWisdom. His research interests include information retrieval, recommendation system, and machine learning.

Rongrong Ji (Senior Member, IEEE) is currently a Nanqiang distinguished professor at Xiamen University, the deputy director at the Office of Science and Technology, Xiamen University, and the director of Media Analytics and Computing Lab. He has authored or coauthored more than 50 papers in ACM/IEEE Transactions, including the *IEEE Transactions on Pattern Analysis and Machine Intelligence* and the *International Journal of Computer Vision*, and more than 100 full papers on top-tier conferences, such as CVPR and NeurIPS. His publications have got over 10K citations in Google Scholar. His research interests include the field of computer vision, multimedia analysis, and machine learning. He was the recipient of the National Science Foundation for Excellent Young Scholars (2014), the National Ten Thousand Plan for Young Top Talents (2017), the National Science Fundation for Distinguished Young Scholars (2020), and the Best Paper Award of ACM Multimedia 2011. He was the area chair in top-tier conferences, such as CVPR and ACM Multimedia. He is currently an advisory member of Artificial Intelligence Construction with the Electronic Information Education Commitee of the National Ministry of Education.