ADELA: Accelerating Evolutionary Design of Machine Learning Pipelines with the Accompanying Surrogate Model

Yang Gu[1], Jian Cao[1*], Hengyu You[1], Nengjun Zhu[2*], Shiyou Qian[1]
[1]Shanghai Jiao Tong University
[2]Shanghai University
{gu_yang, cao-jian, virusyou, qshiyou}@sjtu.edu.cn, zhu_nj@shu.edu.cn

Abstract

The end-to-end automated design of machine learning (ML) pipelines significantly reduces the workload for data scientists and democratizes ML for non-experts. Evolutionary algorithm (EA)-based automated ML (AutoML) systems, a prominent category of AutoML, often face inefficiencies due to the costly fitness evaluation of candidate ML pipelines. Although surrogate models have been employed to approximate the true performance of pipelines more quickly, a key challenge remains in effectively bridging the semantic gap between the heterogeneous features of datasets and pipelines. To address this issue, we propose ADELA, a novel accompanying surrogate-based optimization strategy that accelerates EA-based AutoML while retaining the performance of the resulting pipelines. ADELA operates in two phases: Offline, leveraging a high-quality curated pipeline corpus to meta-learn an accompanying surrogate model; and Online, selecting the accompanying pipeline and using the learned model to predict the performance of evaluation pipelines instead of executing them. The accompanying mechanism effectively mitigates the semantic gap between datasets and pipelines, enabling ADELA to reduce computation times by an average of 73.66% while retaining 98.78% of the final pipeline performance, as demonstrated in extensive experimental evaluations.

Code — https://github.com/t-harden/ADELA

1 Introduction

Machine Learning (ML) applications in data management typically require a comprehensive approach encompassing the entire ML pipeline (Wu et al. 2022; Yang et al. 2019), including data preprocessing, feature engineering, and model selection. Designing task-specific pipelines manually is time-consuming and error-prone (Lazebnik, Somech, and Weinberg 2022). Automated ML (AutoML) addresses this by automating ML processes to create optimal pipelines (Hutter, Kotthoff, and Vanschoren 2019; Gu et al. 2024).

Existing AutoML systems use various optimization techniques, like Bayesian Optimization (BO), Evolutionary Algorithms (EA), and Reinforcement Learning to automatically synthesize pipelines (Feurer et al. 2022; Olson and Moore 2016; Drori et al. 2021). However, they often suffer from inefficiencies due to large search spaces and costly pipeline evaluations (Zhang et al. 2023), leading to high energy consumption and low efficiency (Tornede et al. 2023), emphasizing the need for improvement.

EA-based AutoML uses evolutionary algorithms to iteratively evolve and evaluate candidate ML pipelines. We focus on accelerating EA-based ML pipeline design for two main reasons: EAs excel in optimizing complex ML pipelines, including large-scale and multi-objective requirements (Parmentier et al. 2019; Olson and Moore 2016), and they offer substantial flexibility, easily adapting to new conditions through innovative evolutionary operations (Nikitin et al. 2022). A primary time constraint in EA-based AutoML is the costly fitness evaluation of candidate pipelines, requiring extensive training and testing across the entire dataset (Gijsbers, Vanschoren, and Olson 2018; Li et al. 2020).

To address this issue, some methods utilize **subsets** of data, such as LTPOT (Gijsbers, Vanschoren, and Olson 2018), which evaluates pipelines on smaller subsets before testing promising configurations on the full dataset. Others rely on **surrogate** models, like AVATAR (Nguyen et al. 2020), which employs Petri nets to eliminate infeasible candidates, and MetaTPOT (Laadan et al. 2020), which predicts and ranks performance using meta-surrogate models to optimize resource usage. Subset-based methods improve efficiency but risk losing data information, potentially affecting precision (Lazebnik, Somech, and Weinberg 2022). In contrast, surrogate-based methods predict pipeline performance on original datasets, reducing evaluation time with minimal accuracy loss compared to direct pipeline execution. Previous surrogate-based approaches (Laadan et al. 2020; Zhang et al. 2023) directly concatenate dataset and pipeline features, using multilayer perceptrons (MLPs) for coarse feature fusion. However, these two heterogeneous features exist in different semantic subspaces, creating a semantic gap that complicates accurate prediction. Indeed, during the same evolutionary process, the evaluated ML pipeline with known performance (i.e., **accompanying pipeline**) that is similar to the pipeline being evaluated (i.e., evaluation pipeline) can naturally serve as a semantic bridge for more precise performance prediction on the same dataset.

Consequently, we propose ADELA, a novel strategy for Accelerating evolutionary DEsign of machine Learning pipelines with the Accompanying surrogate model for tabular

*Corresponding author
Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

data (Saha et al. 2022; Zhang et al. 2023). ADELA consists of two phases: In the offline phase, a pipeline corpus is collected and a surrogate model is trained via meta-learning. In the online phase, ADELA selects the accompanying pipeline and leverages the accompanying surrogate model to predict the performance of candidate ML pipelines instead of executing them, thereby speeding up the evolutionary process.

The fundamental idea of our surrogate model is the **accompanying mechanism**, which facilitates the accurate and efficient performance prediction of evaluation pipelines. This mechanism involves selecting an accompanying pipeline from the recent generation of the evaluation pipeline, executing it on the dataset to obtain its true performance, and linking it to the evaluation pipeline using a siamese GRU-based extractor. By incorporating an accompanying bridge with known performance, the model can effectively learn the semantic correlation between the dataset and evaluation pipelines, whose features are from two different modalities, ultimately improving prediction precision.

The main advantage of ADELA is its ability to significantly reduce computation times while retaining competitive performance in the obtained ML pipelines. As an add-on strategy for existing EA-based AutoML systems, it achieves an average runtime reduction of 73.66% with only a 1.22% performance decrease. This paper contributes the following:

- We present an accompanying surrogate-based optimization strategy for EA-based AutoML, aiming to reduce AutoML computation costs with minimal decrease in final pipeline performance.
- We introduce an accompanying mechanism in the surrogate model, which helps bridge the semantic gap between datasets and pipelines, thereby enhancing the model's precision.
- We implement ADELA and perform an extensive experimental comparison with SOTA baselines across 16 datasets from various domains and shapes. The results demonstrate ADELA's effectiveness and scalability.

2 Related Work

2.1 End-to-End Design of ML Pipelines

End-to-end design of ML pipelines aims to automate the development of complete pipelines, often treated as a black-box optimization problem (Hutter, Lücke, and Schmidt-Thieme 2015). Common approaches include random search (Bergstra and Bengio 2012), BO (Karras et al. 2023), EA (Le, Fu, and Moore 2020), and meta-learning (Müller et al. 2021), with EA-based methods being particularly prominent.

TPOT (Olson and Moore 2016) is the most representative EA-based system, using genetic algorithms to automatically design and optimize a series of data transformations and ML models. ML-Plan (Mohr, Wever, and Hüllermeier 2018) generates sequential ML workflows via hierarchical task network planning, while FEDOT (Nikitin et al. 2022) uses a multi-objective evolutionary approach for composite pipelines. Our ADELA aims to enhance the operational efficiency of existing EA-based AutoML tools.

2.2 Accelerating EA-based Design of ML Pipelines

The efficiency of EA-based design of ML pipelines is primarily limited by the vast search space and costly fitness evaluation. While advancements like space pruning (Saha et al. 2022) exist, evaluation remains a key bottleneck.

Subset-based methods enhance efficiency by assessing pipelines on varied data subsets. SubStrat (Lazebnik, Somech, and Weinberg 2022) uses a genetic-based algorithm to create entropy-preserving subsets by concurrently selecting specific rows and columns. Other strategies like independent feature selection (Le, Fu, and Moore 2020) and row sampling (Cohen and Peng 2015) also effectively reduce dataset size. **Surrogate-based methods** use surrogate models to evaluate ML pipelines, reducing the need for real fitness evaluations. eTOP (Zhang, Freire, and Garg 2023) uses low-cost surrogates to early terminate non-viable pipelines, while AVATAR (Nguyen et al. 2020) uses a Petri net-based surrogate for quality assessment. In the realm of Data-Driven EA (DDEA), various approaches (Jin et al. 2018; Li et al. 2020) approximate and substitute the real evaluations with surrogate models derived from previously evaluated solutions. Building on this concept, ML4ML (Zhang et al. 2023) utilizes the provenance of prior pipeline executions to predict performance. MetaT-POT (Laadan et al. 2020) employs a meta-surrogate to select the most promising candidates. However, these methods often overlook the interactions among individuals within a single evolutionary process, which are crucial for accurate performance prediction. **Other methods** for enhancing general EA, like caching (Przewozniczek and Komarnicki 2021) and distributed computing (Nikitin et al. 2023), are also applied in AutoML but often require stronger hardware (e.g., larger memory, more clusters), leading to higher costs.

Our approach is a surrogate-based acceleration strategy that utilizes the accompanying pipeline to improve the prediction capability of surrogate models.

3 Problem Formulation

In a typical end-to-end ML pipeline design, an AutoML tool A aims to build a pipeline P for predicting target values y in a dataset $D = (X \times Y)$, where X (features) has n rows and m columns, and Y (targets) has n rows and c columns. An ML pipeline $P = [o_f^1, o_f^2, ..., o_f^t, o_m]$ includes t feature engineering (FE) operators (o_f) that transform subsets of X and a model operator (o_m) for learning and prediction. An example ML pipeline is shown in Fig. 1.

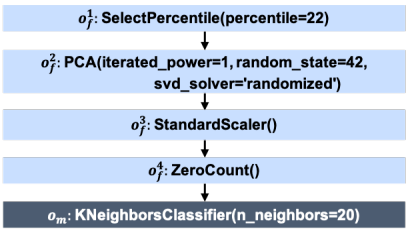


Figure 1: An example ML pipeline with 4 FE operators and 1 model operator.

The search of AutoML tool A over dataset $D = D_{train} \cup$

D_{test} can be defined as: $A(D, y) \rightarrow P_*$, where P_* is the best pipeline (i.e., achieving the highest performance on D_{train}) that A could find. Let $Time(P_*)$ be the time it takes A to generate P_*, with final model accuracy on D_{test} indicated by $\eta(P_*)$. Given a dataset D and a specific predictive task, the goal of ADELA is to reduce computation time of EA-based AutoML systems while maintaining the performance level of the resulting ML pipeline. Namely, to create a new ML pipeline P_{new} s.t. $Time(P_{new}) \ll Time(P_*)$ but $\eta(P_{new}) \approx \eta(P_*)$.

4 Approach

4.1 Overview

As illustrated in Fig. 2, the ADELA framework operates through two stages: In the offline stage, it constructs a corpus of ML pipelines and their corresponding datasets by utilizing an AutoML tool to explore various augmented datasets. This *Pipeline Corpus* is used to meta-train an *Accompanying Surrogate Model*, which rapidly predicts the performance of a given pipeline P_{eval} on dataset D, leveraging insights from an accompanying pipeline P_{acc}. In the online stage, when presented with a new dataset and a predictive task, ADELA selects appropriate accompanying pipelines and employs the accompanying surrogate model to replace traditional fitness evaluations in EA-based AutoML, accelerating the evolutionary process to produce a new pipeline P_{new}.

4.2 Offline1: Creation of Pipeline Corpus

This step aims to curate a high-quality corpus of ML pipelines and datasets to support ADELA's meta-learning. Using an AutoML tool, we search over numerous meta datasets to collect standard pipelines, avoiding noisy and unexecutable ones from open repositories (Cambronero and Rinard 2019).

Meta Datasets and Augmentation We use $R = 58$ representative datasets (IDs available in code repository) from the OpenML-CC18 benchmark (Bischl et al. 2021) as our meta dataset suite, widely used in classification research. To prevent overfitting, we apply bootstrap augmentation (Efron 1992), generating $I = 10$ augmented datasets D^i_{meta} for each n-size dataset D_{meta} by uniformly sampling n examples with replacement. It densely covers the feature space of D_{meta}, offering a more robust training set compared to methods like adding Gaussian noise (Rakotoarison et al. 2022).

Meta Pipelines To train a robust meta-surrogate model for predicting P_{eval} performance online, we employ an EA-based AutoML tool A' to explore augmented meta datasets and collect a substantial number of standardized meta-pipeline tuples $(D^i_{meta}, P^{'}_{eval}, P^{'}_{acc}, perf^{'}_{eval}, perf^{'}_{acc})$.

Evaluation Pipelines. Each pipeline P is represented as an operator list with labels and configurations, e.g., $SelectPercentile$ with $percentile = 22$ (Fig. 1). Using A', we evolve the augmented meta datasets and collect $K = 2000$ generated pipelines (i.e., evaluation pipelines) $P^{'}_{eval}$ for each augmented meta dataset D^i_{meta}. Concurrently, we record their performance $perf^{'}_{eval}$ by calculating the evaluation metric (e.g., F1 score), employing the same computational approach as the AutoML tool A in the online phase.

Accompanying Pipelines. As introduced in Section 1, accompanying pipeline $P^{'}_{acc}$, derived from the same evolutionary process as the evaluation pipeline $P^{'}_{eval}$, can serve as a bridge to facilitate the prediction of $perf^{'}_{eval}$. To this end, we randomly select $J = 10$ accompanying pipelines $P^{'}_{acc}$ from the recent generations of each $P^{'}_{eval}$ and record their corresponding performance $perf^{'}_{acc}$.

In total, $R * I * K * J$ meta-pipeline tuples are collected to meta-train the accompanying surrogate model.

4.3 Offline2: Meta-training of Accompanying Surrogate Model

In this section, we leverage the created pipeline corpus to meta-train the *Accompanying Surrogate Model* for online prediction, as shown in Fig. 3.

Input and Embedding Layer The model inputs include the meta dataset D_{meta}, the evaluation pipeline $P^{'}_{eval}$, and the accompanying pipeline $P^{'}_{acc}$ with its performance score $perf^{'}_{acc}$. To transform each input into a dense vector representation, we first outline the dataset and operator encoding methods in the embedding layer.

Dataset Encoding. Effective dataset encoding captures features that characterize the dataset and discern patterns between them and ML operators. For example, the choice of FE operator often depends on dataset characteristics such as attribute types and missing values. Following (Rakotoarison et al. 2022; Zhang et al. 2023), we compute meta-features using PyMFE (Alcobaça et al. 2020), representing D_{meta} as a d_D-dimensional vector V_D. These meta-features characterize the dataset across general, statistical, information-theoretic, model-based and other perspectives.

Operator Encoding. As shown in Fig. 1, a pipeline operator (whether FE or model type) can be expressed as a string comprising the operator name and its hyper-parameters, essentially forming a single-line code fragment. Inspired by the successful application of pre-trained models in code-related tasks like code completion and search (Liu et al. 2023; Dinh et al. 2024), we utilize the SOTA transformer-based code pre-trained model UniXCoder (Guo et al. 2022) to embed ML operators. Unlike other models that predominantly focus on syntactic patterns, UniXCoder's ability to integrate contextual information allows for capturing the functional characteristics of different operators and their configurations (Zhao and Huang 2018). Consequently, each operator in $P^{'}_{eval}$ and $P^{'}_{acc}$ is represented as a d_O-dimensional vector V_O.

Feature Extraction and Stacking Layer In these layers, we aim to extract and integrate interaction features of datasets and operators using two components:

Attention-based Extractor. Given that different operators in a pipeline focus on different meta-features of the dataset, we introduce an attention net to extract weighted features of the meta dataset w.r.t. the evaluation and accompanying pipeline, respectively.

For a evaluation pipeline $P^{'}_{eval}$ with l operators $V^i_O (i = 1, ..., l)$, we standardize the dimensions of V_D and V^i_O to d'.

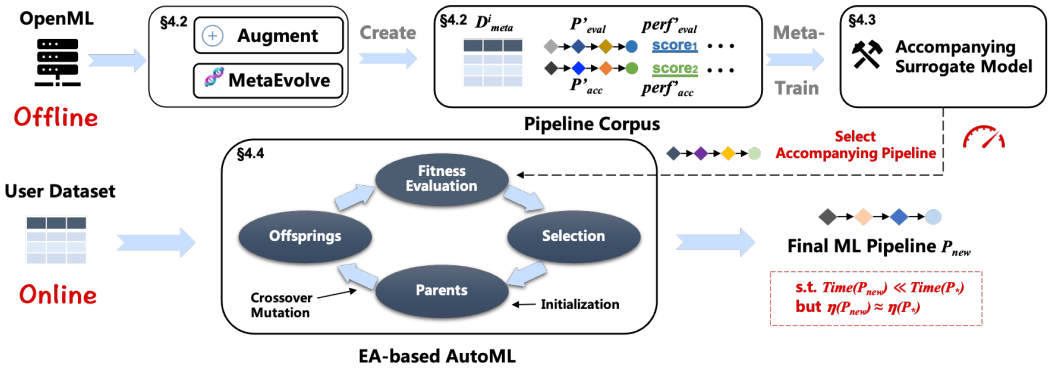


Figure 2: Overview of ADELA.

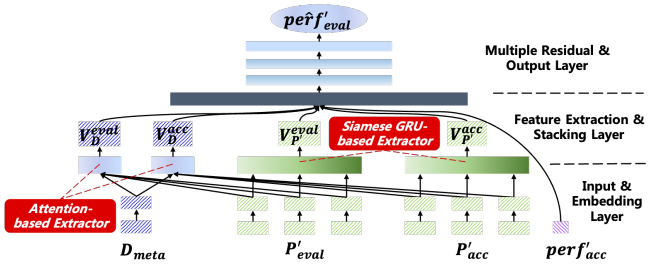


Figure 3: Overview of Accompanying Surrogate Model.

Interaction features between V_D and each V_O^i are modeled using element-wise products, resulting in l interacted vectors $V_D \odot V_O^i$. A *Dataset Attention Network* then computes attention scores for these vectors using a one-layer MLP followed by a softmax function. The d'-dimensional weighted dataset vector V_D^{eval} is finally computed as the weighted average of these interacted vectors. The same process is used to obtain V_D^{acc} for the accompanying pipeline, thereby encapsulating all feature interactions in the embedding space with differentiating their importance.

Siamese GRU-based Extractor. This component aims to generate two d'-dimensional pipeline embeddings for P'_{eval} and P'_{acc} using their operator embeddings.

For each sequence of operators, we employ a dual-layer BiGRU network (Chung et al. 2014) to model the pipeline's sequential information. The BiGRU effectively captures dependencies in sequential data, with the final hidden state serving as the pipeline's embedding (Pineda Arango and Grabocka 2023). For two pipeline sequences, we design a siamese architecture. Notably, the known performance value of P'_{acc} on D_{meta} serve as a reference for predicting the performance of P'_{eval} by establishing a connection between the evaluation and accompanying pipeline. The siamese architecture helps to learn pipeline embeddings that embody invariance and selectivity by incorporating structural and semantic similarities between the evaluation and accompanying pipeline (Qian, Wen, and Kumar 2019). In this way, the evaluation pipeline and the dataset, whose features come from two modalities, are bridged by the accompanying pipeline. The *Siamese GRU-based Extractor* is a dual-branch network with shared parameters (Cho et al. 2014), producing embeddings $V_{P'}^{eval}$ and $V_{P'}^{acc}$. These are concatenated with V_D^{eval}, V_D^{acc}, and $perf'_{acc}$ in the stacking layer.

Multiple Residual and Output Layer After learning and stacking various features, this component integrates them to predict the performance of P'_{eval} on D_{meta}. We employ a multi-layer residual network (He et al. 2016) for deep feature fusion and faster training convergence.

The output layer applies a sigmoid function to normalize the predicted score between 0 and 1. To train the end-to-end surrogate model, we use squared loss with $L2$ regularization to prevent overfitting. The objective function is:

$$Loss = \sum_{\mathrm{x} \in \mathbb{X}} \left(\hat{perf}'_{eval}(\mathrm{x}) - perf'_{eval}(\mathrm{x})\right)^2 + \lambda \sum_{\omega \in \Omega} \omega^2 \quad (1)$$

where $\mathbb{X}$ is the set of training instances, $perf'_{eval}(\mathrm{x})$ is the target performance of instance x, λ controls regularization strength and Ω represents the neural network parameters.

4.4 Online: Accelerating Evolutionary Design

Once the accompanying surrogate model is meta-trained, it can be used to predict performance and accelerate the AutoML tool A on the user dataset D online. For detailed pseudocode of ADELA's online phase, refer to the Appendix.

In the initial population, pipelines are directly executed for evaluation. From the first generation onward, we use the accompanying surrogate model to obtain the predicted performance value for each evolved pipeline instead of executing them. A key step is selecting an appropriate accompanying pipeline P_{acc} and executing it to yield its true performance $perf_{acc}$, which aids the surrogate's prediction.

Due to the stabilizing effect of selection pressure on advantageous genes and limited genetic variation from mutation (Eiben and Smith 2015), pipelines from recent generations are more correlated. Thus, we select the pipeline with the highest fitness score from the most recent generation of P_{eval} as P_{acc}, executing it to obtain $perf_{acc}$. We have also explored alternative strategies for selecting the accompanying pipeline, as discussed in Section 5.6, but these did not improve performance. Upon meeting the stopping criteria of evolution, the pipeline P_{new} with the best fitness score is returned.

5 Experiments

5.1 Methodology and Setup

Experimental Methodology & Metric Given an input dataset and a prediction target, we first directly employ an EA-based AutoML tool A and obtain its output ML pipeline P_*. We use TPOT (ver. 0.12.1) as our testbed, which is the most popular EA-based AutoML system in the data science community (Laadan et al. 2020; Li et al. 2023). We record both the running time (in seconds) and the performance of the resulting model (macro F1 score), which serve as our primary baseline, denoted as vanilla-AutoML.

Next, we investigate whether our accompanying surrogate-based strategy can effectively reduce EA-based AutoML running times while still generating ML pipelines with accuracy comparable to vanilla-AutoML. Specifically, in the offline phase, we also use TPOT to evolve and collect meta-pipeline tuples, aiming to meta-train the accompanying surrogate model. Notably, other AutoML tools are also feasible for meta-evolving, provided they maintain consistent pipeline configurations with the online tool. In the online phase, for each instance, we apply our ADELA and compute the relative running time and performance of P_{new} w.r.t. vanilla-AutoML. Following the existing literature (Lazebnik, Somech, and Weinberg 2022), we report the following metrics, where larger values indicate better performance:

- $\textbf{Time-Reduction}$ indicates how much time is saved:

$$Time\text{-}Reduction = \frac{Time(P_*) - Time(P_{new})}{Time(P_*)} \tag{2}$$

- $\textbf{Relative-F1}$ indicates the proportion of performance of vanilla-AutoML that is successfully retained:

$$Relative\text{-}F1 = \frac{F1(P_{new})}{F1(P_*)} \tag{3}$$

Datasets We evaluate using 16 public and real-world classification datasets from OpenML, UCI, and Kaggle, as summarized in Table 1. These datasets vary in characteristics and fall into four categories: 1) Standard, containing several thousand rows and a few dozen columns; 2) Long, comprising more than 30,000 rows; 3) HighDim, high-dimensional datasets with over 200 columns; and 4) HighDim-W, particularly wide datasets with up to 5,000 columns and a column-to-row ratio of at least 45%. See Appendix for more details.

Type	Name	#Classes
Standard	spambase, phoneme, pendigits, PhishingWebsites, Mushroom, Dropout&Success	{2, 3, 10}
Long	nomao, FMA, Dota2 Games	{2, 16}
HighDim	mfeat-factors, isolet, mfeat-pixel	{10, 26}
HighDim-W	Internet-Ad., Bioresponse, Toxicity, Gisette	{2}

Table 1: Descriptions of datasets used.

Implementation Details In the offline phase, we use TPOT to evolve and collect 11.6 million pipeline tuples corpus by specifying linear templates (Le, Fu, and Moore 2020). The data is split into 70% training, 10% validation, and 20% testing. And the evaluation datasets are completely unseen by ADELA. The vector dimensions for d_D, d_O, and d' are 111, 768, and 100, respectively. Training is conducted on a machine equipped with an NVIDIA A40 GPU, using the Adam optimizer with a learning rate of 10^{-3}, and up to 500 epochs with an early stopping strategy.

In the online phase, we perform five trials for each experiment on each evaluation dataset, evolving with 20 generations and a population size of 50. For each trial, we randomly split the dataset into training and testing data with a 75:25 ratio. All baselines are run using the same train-test split of data in each trial to ensure a fair comparison. We use the mean score of the five runs to compare the results, following the methodology in (Lazebnik, Somech, and Weinberg 2022). The experiments were run on a MacBook Pro with an Apple M1 chip and 16GB of memory.

5.2 Comparative Methods

For clarity, we compare ADELA only to other add-on methods designed to expedite classical EA-based AutoML systems, as these are more relevant for current users, rather than to new, lighter AutoML tools (Li et al. 2023). Additionally, hardware-related acceleration strategies (Przewozniczek and Komarnicki 2021; Nikitin et al. 2023) are excluded, as they are not applicable for common users and scenarios.

We implemented several state-of-the-art baselines including SubStrat (Lazebnik, Somech, and Weinberg 2022), IGKM (Kraskov, Stögbauer, and Grassberger 2004), LTPOT (Gijsbers, Vanschoren, and Olson 2018), MetaTPOT (Laadan et al. 2020), ML4ML (Zhang et al. 2023), and AVATAR (Nguyen et al. 2020). For more detailed descriptions, see Section 1, 2 and Appendix. In short, the first three methods are subset-based acceleration algorithms, while the last three, along with ADELA, are surrogate-based methods.

5.3 Overall Comparison Results

Table 2 displays the evaluation results in terms of Time-Reduction (Time-Red.) and Relative-F1 (Rel.-F1). Each baseline was run 5 times per dataset, with the table showing the mean and standard deviation values across all 16 datasets. Results are compared across different expected pipeline lengths $l = 3, 5, 7$, specified by the user (the actual evolved pipeline length equalling or approximating l due to crossover and mutation). Although theoretically possible to have arbitrarily long pipelines, such extensive pipelines are often undesirable due to tuning difficulties and overfitting risks (Gijsbers, Vanschoren, and Olson 2018). Thus, we balance operator counts and performance, focusing on lengths common in ML pipelines (Pineda Arango and Grabocka 2023).

First of all, it can be observed that ADELA is the only method achieving a Relative-F1 higher than 98%, with mean scores of 99.11%, 98.96%, and 98.27% for $l = 3, 5, 7$, respectively. Although ADELA's time reduction metric is not the most competitive among the baselines, it still saves over 70% of the time compared to the vanilla algorithm. Significantly, there is an ongoing discussion about the acceptability of model accuracy in light of training time savings, with a decrease of more than 5% in accuracy being undesirable for AutoML (Lazebnik, Somech, and Weinberg 2022). Therefore,

Method	l=3		l=5		l=7	
	Time-Red.	Rel.-F1	Time-Red.	Rel.-F1	Time-Red.	Rel.-F1
SubStrat	76.38±9.82%	96.51±1.87%	78.98±9.82%	96.21±1.55%	79.24±9.53%	95.89±1.82%
IG-KM	78.31±8.47%	92.33±2.14%	80.31±8.27%	90.21±2.19%	82.14±8.77%	89.48±2.42%
LTPOT	69.42±10.78%	96.24±2.28%	70.54±10.34%	95.36±2.08%	73.02±10.81%	95.06±2.24%
MetaTPOT	74.90±9.43%	90.14±3.03%	76.92±10.23%	89.24±3.53%	77.99±10.03%	88.47±3.03%
ML4ML	73.11±8.89%	94.25±2.52%	74.12±8.45%	93.02±3.02%	75.32±8.95%	92.78±2.22%
AVATAR	72.55±8.89%	92.79±2.33%	72.95±8.52%	91.29±2.24%	74.01±9.10%	91.89±2.54%
ADELA	72.69±11.02%	**99.11±1.98%**	73.62±10.21%	**98.96±1.67%**	74.66±11.32%	**98.27±1.86%**

Table 2: The overall performance of accelerating evolutionary design of ML pipelines.

the slight difference in time savings can be overlooked considering ADELA's remarkable Relative-F1 compared to other approaches. We now analyze the results of the remaining baselines and compare them with ADELA:

- Among surrogate-based approaches, the significant score gap between ADELA and the others demonstrates the effectiveness of our method in sufficiently extracting and matching the semantics of datasets and pipelines. For instance, ML4ML, the strongest competitor, directly matches meta feature-based dataset encoding with OneHot-based pipeline encoding for performance prediction. Consequently, it lags behind ADELA by 4.86, 5.94, and 5.49 points in terms of Relative-F1 for $l = 3, 5, 7$, respectively.
- Generally, subset-based baselines are more efficient than surrogate-based ones. Although IG-KM excels best in Time-Reduction, its Relative-F1 is often lower. This may be because the data subset generated by IG-KM is biased; the target attribute is correlated with only a portion of the columns selected by IG-KM, rather than representing the entire dataset (Lazebnik, Somech, and Weinberg 2022).
- To further illustrate the performance across each datasets, we analyze the distribution of the number of datasets w.r.t. the Relative-F1 interval, based on the average score for $l = 3, 5, 7$, focusing on two representative methods in each category, SubStrat and ML4ML, alongside our ADELA. As shown in Fig. 4, ADELA surpasses the 98% Relative-F1 threshold in 13 out of 16 datasets, while SubStrat achieves this level in only 2/16 datasets and ML4ML in just 1/16.

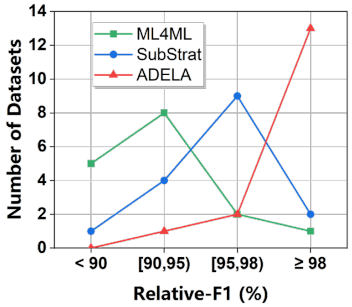


Figure 4: The distribution of #Datasets w.r.t. Relative-F1.

5.4 Scalability Analysis

Pipeline Length Analysis As shown in Table 2, as the expected pipeline length l increases, the Time-Red. metric improves for all algorithms, particularly for subset-based methods. This is because executing longer pipelines on a small-sized dataset saves significantly more time than evaluating them using a similar surrogate model. Conversely, the Rel.-F1 scores of all baselines, including ADELA, tend to decline since more operators introduce complexities in extracting their interaction semantics and matching the dataset with pipelines. Despite this, ADELA maintains a 98% Rel.-F1 level, further validating the accompanying surrogate model's capability to integrate heterogeneous features and accurately predict pipeline performance.

Dataset Shape Analysis We next analyze the performance of ADELA and two representative baselines, SubStrat and ML4ML, across four dataset shape categories for $l = 5$ (similar results were observed for $l = 3, 7$, thus omitted).

Table 3 lists the mean Time-Red. and Rel.-F1 for each dataset category. Firstly, for *Standard* and *Long* datasets, ADELA outperforms SubStrat and ML4ML by an average of 3.59 and 5.59 points in terms of the Rel.-F1 metric, a substantial margin for AutoML. Although SubStrat and ML4ML achieve higher time savings, future work could enhance the inference speed of the accompanying surrogate model by employing quantization (Zhang, Zhou, and Saab 2023) or optimization tools (Xia et al. 2024). For *HighDim* datasets, ADELA performs better than both baselines on all metrics. When it comes to *HighDim-W* datasets, SubStrat's Time-Red. dramatically drops to 55.62%, while ADELA and ML4ML maintain over 70%. This indicates that surrogate-based acceleration methods using deep learning are less constrained by dataset shape compared to subset-based approaches relying on dataset-entropy measures, especially those computed for each column in SubStrat.

In conclusion, our acceleration framework ADELA is scalable in terms of both pipeline length and dataset shape.

5.5 Ablation Analysis

We conduct ablation studies to examine the contribution of the main components/mechanisms in the framework. We present the results for $l = 5$ in Table 4, noting similar trends for $l = 3, 7$. As mentioned in Section 5.3, we focus primarily on the Relative-F1 metric, given that the time savings are relatively comparable across ADELA's variants. Below are our findings:

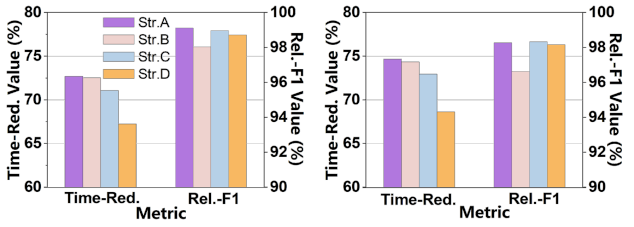
1) Removing the **accompanying mechanism** results in a significant 4.72-point drop in Relative-F1, highlighting the importance of using the accompanying pipeline with true

Method \ Category	Standard		Long		HighDim		HighDim-W	
	Time-Red.	Rel.-F1	Time-Red.	Rel.-F1	Time-Red.	Rel.-F1	Time-Red.	Rel.-F1
SubStrat	85.77±5.26%	96.89±2.23%	72.76±11.32%	94.44±1.12%	75.35±10.88%	96.34±0.75%	55.62±8.04%	94.25±1.58%
ML4ML	81.21±7.55%	94.54±3.56%	70.81±12.74%	92.79±2.60%	77.86±12.33%	93.44±1.12%	72.13±8.23%	91.86±3.53%
ADELA	80.46±7.49%	**99.84±1.23%**	69.74±14.87%	**98.67±1.82%**	78.16±13.76%	**99.10±1.01%**	71.62±8.99%	**98.25±2.02%**

Table 3: The performance of different dataset shape categories.

Method	Time-Red.	Rel.-F1
ADELA	73.62±10.21%	**98.96±1.67%**
\Accompanying	75.88±9.25%	94.24±2.98%
\Attention	74.71±9.21%	96.02±1.86%
↻Siamese GRU	75.13±10.13%	95.35±2.03%
↻Pre-trained	73.91±10.30%	97.69±1.79%

Table 4: Ablation analysis on main components/mechanisms of ADELA. \Accompanying and \Attention denote removing the accompanying mechanism and attention-based extractor, respectively. ↻Siamese GRU denotes replacing siamese GRU-based extractor with average pooling for operators. ↻Pre-trained denotes replacing the code pre-trained model with randomized initialization for operator embedding.



(a) $l = 3$ (b) $l = 7$

Figure 5: Performance analysis of ADELA w.r.t. different selection strategies of accompanying pipeline.

performance to mitigate the semantic gap between dataset and pipeline modalities.

2) Not extracting **attention-based interacted features** leads to a performance decline, providing compelling evidence that our dataset attention network effectively captures the varying importance of interacted semantics between the dataset and pipeline operators.

3) The **Siamese GRU-based Extractor** helps model the interaction features of operators within a pipeline sequence, and the symmetric architecture guides the learning of internal similarities between the evaluation and accompanying pipelines, also reducing the training cost burden.

4) Replacing the **code pre-trained model** with randomized initialization slightly harms performance, confirming the pre-trained domain-specific model's effectiveness in understanding the contextual semantics and functional characteristics of different operators and their configurations.

5.6 Accompanying Pipeline Selection Analysis

Intuitively, the selection of an accompanying pipeline for the evaluation pipeline affects the surrogate model's prediction performance in the online phase. Therefore, we investigate the impact of different selection strategies on the evolutionary results. Specifically, we compare four strategies as follows: A) Highest fitness score from the most recent generation (ADELA's strategy); B) Highest fitness score from all previous generations; C) Most similar pipeline from the most recent generation; D) Most similar pipeline from all previous generations. The pipeline similarity refers to the average cosine similarity of corresponding operator embeddings.

From Fig. 5(a) and Fig. 5(b), we observe:

- The Time-Reduction metric for strategies A and B consistently outperforms that of strategies C and D. This is because the latter two strategies, particularly D, require additional time to compute the semantic similarity between the evaluation pipeline and pipelines from previous generations, especially when the population size is large.
- Strategy B performs worse than strategy A in Relative-F1, indicating that a high-fitness pipeline from a non-recent generation may lack semantic correlation with the evaluation pipeline, which cannot be adequately captured even using the Siamese GRU-based extractor. Notably, the almost equal Relative-F1 results of strategies A and C suggest that in the most recent generation, the most similar pipeline also tends to have a high fitness score due to limited genetic variation and selection pressure (Eiben and Smith 2015). However, strategy C incurs more time to compute similarity.

In summary, strategy A (ADELA) demonstrates the best balance of effectiveness and efficiency among the methods.

6 Conclusion

We propose ADELA, an accompanying surrogate-based acceleration strategy for EA-based AutoML. ADELA utilizes a curated pipeline corpus to meta-learn an accompanying surrogate model offline, enabling pipeline performance prediction online without training or testing. The key to ADELA is the accompanying mechanism, which bridges the semantic gap between datasets and pipelines, enhancing the surrogate's accuracy. Experiments show that ADELA achieves an average Relative-F1 of 98.78% while reducing runtime by 73.66%.

It is important to point out that ADELA is not limited to specific AutoML system, either offline or online, which enables data scientists to continue using their preferred EA-based AutoML tools while significantly reducing running times and computation costs. This contributes to the advancement of "Green AutoML" (Tornede et al. 2023) and energy-efficient AI systems. Future research will expand beyond EA-based approaches to explore the application of (multi-) accompanying mechanism, aiming to further expedite and enhance general AutoML tools.

Acknowledgments

This work is supported by the Interdisciplinary Program of Shanghai Jiao Tong University (project number YG2024QNB05). This work is also supported by the Program of Technology Innovation of the Science and Technology Commission of Shanghai Municipality (Granted No. 21511104700).

References

Alcobaça, E.; Siqueira, F.; Rivolli, A.; Garcia, L. P.; Oliva, J. T.; and De Carvalho, A. C. 2020. MFE: Towards reproducible meta-feature extraction. *Journal of Machine Learning Research*, 21(111): 1–5.

Bergstra, J.; and Bengio, Y. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2).

Bischl, B.; Casalicchio, G.; Feurer, M.; Gijsbers, P.; Hutter, F.; Lang, M.; Mantovani, R. G.; van Rijn, J. N.; and Vanschoren, J. 2021. OpenML Benchmarking Suites. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Cambronero, J. P.; and Rinard, M. C. 2019. AL: autogenerating supervised learning programs. *Proceedings of the ACM on Programming Languages*, 3(OOPSLA): 1–28.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Cohen, M. B.; and Peng, R. 2015. Lp row sampling by lewis weights. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, 183–192.

Dinh, T.; Zhao, J.; Tan, S.; Negrinho, R.; Lausen, L.; Zha, S.; and Karypis, G. 2024. Large language models of code fail at completing code with potential bugs. *Advances in Neural Information Processing Systems*, 36.

Drori, I.; Krishnamurthy, Y.; Rampin, R.; Lourenco, R. d. P.; Ono, J. P.; Cho, K.; Silva, C.; and Freire, J. 2021. AlphaD3M: Machine learning pipeline synthesis. *arXiv preprint arXiv:2111.02508*.

Efron, B. 1992. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics: Methodology and distribution*, 569–593. Springer.

Eiben, A. E.; and Smith, J. E. 2015. *Introduction to evolutionary computing*. Springer.

Feurer, M.; Eggensperger, K.; Falkner, S.; Lindauer, M.; and Hutter, F. 2022. Auto-sklearn 2.0: Hands-free automl via meta-learning. *Journal of Machine Learning Research*, 23(261): 1–61.

Gijsbers, P.; Vanschoren, J.; and Olson, R. S. 2018. Layered TPOT: Speeding up tree-based pipeline optimization. *arXiv preprint arXiv:1801.06007*.

Gu, Y.; You, H.; Cao, J.; and Yu, M. 2024. Large Language Models for Constructing and Optimizing Machine Learning Workflows: A Survey. *arXiv preprint arXiv:2411.10478*.

Guo, D.; Lu, S.; Duan, N.; Wang, Y.; Zhou, M.; and Yin, J. 2022. UniXcoder: Unified Cross-Modal Pre-training for Code Representation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 7212–7225.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hutter, F.; Kotthoff, L.; and Vanschoren, J. 2019. *Automated machine learning: methods, systems, challenges*. Springer Nature.

Hutter, F.; Lücke, J.; and Schmidt-Thieme, L. 2015. Beyond manual tuning of hyperparameters. *KI-Künstliche Intelligenz*, 29: 329–337.

Jin, Y.; Wang, H.; Chugh, T.; Guo, D.; and Miettinen, K. 2018. Data-driven evolutionary optimization: An overview and case studies. *IEEE Transactions on Evolutionary Computation*, 23(3): 442–458.

Karras, A.; Karras, C.; Schizas, N.; Avlonitis, M.; and Sioutas, S. 2023. AutoML with Bayesian optimizations for big data management. *Information*, 14(4): 223.

Kraskov, A.; Stögbauer, H.; and Grassberger, P. 2004. Estimating mutual information. *Physical review E*, 69(6): 066138.

Laadan, D.; Vainshtein, R.; Curiel, Y.; Katz, G.; and Rokach, L. 2020. MetaTPOT: enhancing a tree-based pipeline optimization tool using meta-learning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2097–2100.

Lazebnik, T.; Somech, A.; and Weinberg, A. I. 2022. Substrat: A subset-based optimization strategy for faster automl. *Proceedings of the VLDB Endowment*, 16(4): 772–780.

Le, T. T.; Fu, W.; and Moore, J. H. 2020. Scaling tree-based automated machine learning to biomedical big data with a feature set selector. *Bioinformatics*, 36(1): 250–256.

Li, J.-Y.; Zhan, Z.-H.; Wang, H.; and Zhang, J. 2020. Data-driven evolutionary algorithm with perturbation-based ensemble surrogates. *IEEE Transactions on Cybernetics*, 51(8): 3925–3937.

Li, Y.; Shen, Y.; Zhang, W.; Zhang, C.; and Cui, B. 2023. VolcanoML: speeding up end-to-end AutoML via scalable search space decomposition. *The VLDB Journal*, 32(2): 389–413.

Liu, C.; Lu, S.; Chen, W.; Jiang, D.; Svyatkovskiy, A.; Fu, S.; Sundaresan, N.; and Duan, N. 2023. Code execution with pre-trained language models. *arXiv preprint arXiv:2305.05383*.

Mohr, F.; Wever, M.; and Hüllermeier, E. 2018. ML-Plan: Automated machine learning via hierarchical planning. *Machine Learning*, 107: 1495–1515.

Müller, S.; Hollmann, N.; Arango, S. P.; Grabocka, J.; and Hutter, F. 2021. Transformers can do bayesian inference. *arXiv preprint arXiv:2112.10510*.

Nguyen, T.-D.; Maszczyk, T.; Musial, K.; Zöller, M.-A.; and Gabrys, B. 2020. Avatar-machine learning pipeline evaluation using surrogate model. In *Advances in Intelligent Data Analysis XVIII: 18th International Symposium on Intelligent Data Analysis, IDA 2020, Konstanz, Germany, April 27–29, 2020, Proceedings 18*, 352–365. Springer.

Nikitin, N. O.; Teryoshkin, S.; Pokrovskii, V.; Pakulin, S.; and Nasonov, D. 2023. Improvement of Computational Performance of Evolutionary AutoML in a Heterogeneous Environment. In *2023 IEEE Congress on Evolutionary Computation (CEC)*, 1–8. IEEE.

Nikitin, N. O.; Vychuzhanin, P.; Sarafanov, M.; Polonskaia, I. S.; Revin, I.; Barabanova, I. V.; Maximov, G.; Kalyuzhnaya, A. V.; and Boukhanovsky, A. 2022. Automated evolutionary approach for the design of composite machine learning pipelines. *Future Generation Computer Systems*, 127: 109–125.

Olson, R. S.; and Moore, J. H. 2016. TPOT: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on automatic machine learning*, 66–74. PMLR.

Parmentier, L.; Nicol, O.; Jourdan, L.; and Kessaci, M.-E. 2019. Tpot-sh: A faster optimization algorithm to solve the automl problem on large datasets. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, 471–478. IEEE.

Pineda Arango, S.; and Grabocka, J. 2023. Deep Pipeline Embeddings for AutoML. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1907–1919.

Przewozniczek, M. W.; and Komarnicki, M. M. 2021. Fitness caching-from a minor mechanism to major consequences in modern evolutionary computation. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, 1785–1791. IEEE.

Qian, C.; Wen, L.; and Kumar, A. 2019. TraceWalk: Semantic-based process graph embedding for consistency checking. *arXiv preprint arXiv:1905.06883*.

Rakotoarison, H.; Milijaona, L.; Rasoanaivo, A.; Sebag, M.; and Schoenauer, M. 2022. Learning meta-features for automl. In *ICLR 2022-International Conference on Learning Representations (spotlight)*.

Saha, R. K.; Ura, A.; Mahajan, S.; Zhu, C.; Li, L.; Hu, Y.; Yoshida, H.; Khurshid, S.; and Prasad, M. R. 2022. SapientML: synthesizing machine learning pipelines by learning from human-writen solutions. In *Proceedings of the 44th International Conference on Software Engineering*, 1932–1944.

Tornede, T.; Tornede, A.; Hanselle, J.; Mohr, F.; Wever, M.; and Hüllermeier, E. 2023. Towards green automated machine learning: Status quo and future directions. *Journal of Artificial Intelligence Research*, 77: 427–457.

Wu, Y.; Lentz, M.; Zhuo, D.; and Lu, Y. 2022. Serving and optimizing machine learning workflows on heterogeneous infrastructures. *arXiv preprint arXiv:2205.04713*.

Xia, C.; Zhao, J.; Sun, Q.; Wang, Z.; Wen, Y.; Yu, T.; Feng, X.; and Cui, H. 2024. Optimizing deep learning inference via global analysis and tensor expressions. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, 286–301.

Yang, Y.; Zhan, D.-C.; Wu, Y.-F.; Liu, Z.-B.; Xiong, H.; and Jiang, Y. 2019. Semi-supervised multi-modal clustering and classification with incomplete modalities. *IEEE Transactions on Knowledge and Data Engineering*, 33(2): 682–695.

Zhang, H.; Freire, J.; and Garg, Y. 2023. eTOP: Early Termination of Pipelines for Faster Training of AutoML Systems. *arXiv preprint arXiv:2304.08597*.

Zhang, H.; López, R.; Santos, A.; Piazentin Ono, J.; Bessa, A.; and Freire, J. 2023. Using Pipeline Performance Prediction to Accelerate AutoML Systems. In *Proceedings of the Seventh Workshop on Data Management for End-to-End Machine Learning*, 1–11.

Zhang, J.; Zhou, Y.; and Saab, R. 2023. Post-training quantization for neural networks with provable guarantees. *SIAM Journal on Mathematics of Data Science*, 5(2): 373–399.

Zhao, G.; and Huang, J. 2018. Deepsim: deep learning code functional similarity. In *Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*, 141–151.