Meta-data Augmentation based Search Strategy through Generative Adversarial Network for AutoML Model Selection

Yue Liu[1,2,3][0000−0002−2883−5216], Wenjie Tian[1], and Shuang Li[1]

[1] School of Computer Engineering and Science, Shanghai University
[2] Shanghai Institute for Advanced Communication and Data Science
[3] Shanghai Engineering Research Center of Intelligent Computing System
Shanghai, 200444, China
{yueliu,twenjie,aimeeli}@shu.edu.cn

Abstract. Automated machine learning (AutoML) attempts to automatically build appropriate learning model for given dataset. Despite the recent progress of meta-learning to find good instantiations for AutoML framework, it is still difficult and time-consuming to collect sufficient meta-data with high quality. Therefore, we propose a novel method named Meta-data Augmentation based Search Strategy (MDASS) for AutoML model selection, which is mainly composed of Meta-GAN Surrogate model (MetaGAN) and Self-Adaptive Meta-model (SAM). MetaGAN employs Generative Adversarial Network as surrogate model to collect effective meta-data based on the limited meta-data, which can alleviate the dilemma of meta-overfitting in meta-learning. Based on augmented meta-data, SAM self-adaptively builds multi-objective meta-model, which can select the algorithms with proper trade-off between learning performance and computational budget. Furthermore, for new datasets, MDASS combines promising algorithms and hyperparameter optimization to perform automated model selection under time constraint. Finally, the experiments on various classification datasets from OpenML and algorithms from scikit-learn are conducted. The results show that GAN is promising to incorporate with AutoML and MDASS can perform better than the competing approaches with time budget.

Keywords: AutoML · Meta-learning · Data Augmentation · Generative Adversarial Network · Automated Model Selection

1 Introduction

Automated machine learning (AutoML) attempts to automate tedious but core tasks efficiently [22] to enable the wide-spread use of machine learning by non-experts [21]. It has shown promising results in various tasks, such as feature engineering [14], hyperparameter optimization [12], model selection [18, 5, 7].

Despite recent progress, model selection in high-dimension configuration space is still a major challenge in AutoML. To improve the efficiency, meta-learning

is widely adopted [5, 6, 21] to learn how different configuration performs across datasets (meta-knowledge) from previous learning experience of prior datasets (meta-data) [19]. The existing researches pay more attention to how to better extract meta-knowledge. However, meta-data also plays an important role in meta-learning. It is difficult to collect sufficient meta-data with high quality for the following problems: i) Since some configurations may spend several hours or days to evaluate performance, collecting sufficient meta-data endures more computational budget. ii) The datasets provided by repositories such as UCI [2] and OpenML [20] are still limited and their distribution cannot cover the entire distribution of prior datasets. Meta-learning cannot work well when the distribution of new datasets is significantly different from prior datasets.

To efficiently collect meta-data, the existing promising methods limit maximum runtime or memory consumption in model evaluation. The models exceeding runtime or memory remain missing entries in meta-data. Afterward, the techniques such as collaborative filtering [16, 21] or matrix factorization [7] are to complete meta-data. However, these techniques require at least one model on each dataset trained successfully, and become poor when facing numerous missing entries. Reference [11] showed that some datasets are not reliable for being used in hyperparameter optimization, and presented an interesting approach of using probabilistic encoder to generate inexpensive and realistic data for optimization benchmarking. Therefore, it is significant to directly generate meta-data through generative model for meta-learning. Generative Adversarial Network (GAN), a framework to estimate generative models through adversarial training [8, 15, 1], has been prevailing for data augmentation since its origin [9]. One of the key features of GAN is its generic nature in sampling from an unspecified distribution underlying the given data. This feature fits well when augmenting data from datasets collected without knowing a prior distribution, such in the case of meta-learning. Therefore, meta-data augmentation based on GAN is a promising approach to improve the generalization of meta-learning.

In this paper, we investigate the approach of incorporating GAN with meta-learning for meta-data augmentation, named as Meta-GAN Surrogate Model (MetaGAN). We then further propose Meta-data Augmentation based Search Strategy (MDASS) for AutoML model selection, composed of MetaGAN and Self-Adaptive Meta-model (SAM). Our contributions are summarized as follows:

- In order to efficiently collect sufficient meta-data with high quality, we propose MetaGAN for meta-data augmentation. It employs GAN as surrogate model to generate meta-data described by meta-features and specific class that indicates the algorithm performance. Therefore, it drastically reduces the computational budget to collect meta-data and renders meta-learning more comprehensive to exploit the underlying distribution of meta-data.
- In order to effectively extract meta-knowledge, we propose SAM for algorithm selection. It self-adaptively builds meta-model based on augmented meta-data through appropriate classification and regression learners with two meta-tasks: selecting high-performance algorithms and predicting run-

time order respectively. This model can determine which algorithms have the right trade-off between computational budget and performance.
- In order to build appropriate models for new datasets under time constraint, MDASS combines promising algorithms selection and hyperparameter optimization, and employs Bayesian optimization to search the best model with the runtime order in the expected budget.
- Finally, we conduct experiments on a wide range of datasets from OpenML [20] and various machine learning algorithms from scikit-learn [17]. The results show that MetaGAN can significantly improve the performance of meta-learning and MDASS can obtain an appropriate model with the proper trade-off between runtime and accuracy within certain time constraints.

The remainder of this paper is organized as follows. Section 2 describes the main idea of MDASS. Section 3 analyzes the experiments and results. Finally, we present the conclusion and future work in Section 4.

2 Meta-data Augmentation Based Search Strategy for AutoML Model Selection (MDASS)

The framework of MDASS as shown in Fig. 1 is to perform meta-data augmentation through MetaGAN and determine which algorithms have proper trade-off between runtime and accuracy through SAM. We construct accurcacy-oriented meta-data and runtime-oriented meta-data from hand-picked datasets and specific algorithms. Then, accuracy-oriented meta-data is to execute augmentation through MetaGAN (see Section 2.1). Based on augmented meta-data, SAM is built for promising algorithm selection and runtime order prediction (see Section 2.2). Given a new dataset, MDASS computes its meta-feature for SAM, and the promising algorithms are evaluated in this order until exceeding the time budget to build appropriate model. Here, we employ the commonly used Bayesian optimization [18, 5, 12] to evaluate algorithm performance. Notably, when all promising algorithms have been evaluated sequentially but time budget is not exceeding, MDASS executes the process of joint optimization, i.e. combining promising algorithm selection and corresponding hyperparameter optimization.

2.1 MetaGAN for meta-data augmentation

Problem Formulation Meta-data, the foundation of meta-learning, in MDASS can be defined as $M = \langle F, A, R \rangle$, where $F \in \mathbb{R}^{m \times l}$, $A \in \mathbb{R}^{m \times n}$ and $R \in \mathbb{R}^{m \times n}$ are the matrixes of meta-features, accuracy and runtime. l, m and n represent the number of meta-features, hand-picked datasets and specific algorithms.

Let $d_1, ..., d_m$ be a set of hand-picked datasets. They are the instantiations sampled from an unknown datasets distribution $d_i \sim p(d)$. For each dataset d_i, it has an associated objective function $f_d : F_i \in \mathbb{R}^l \rightarrow \mathbb{R}^n$ mapping the meta-features to algorithm performance. Every entry A_{ij} in accuracy matrix records the accuracy of algorithm a_j on dataset d_i. The runtime of corresponding learning model is recorded as R_{ij}. Here, we name the combination of F and A as

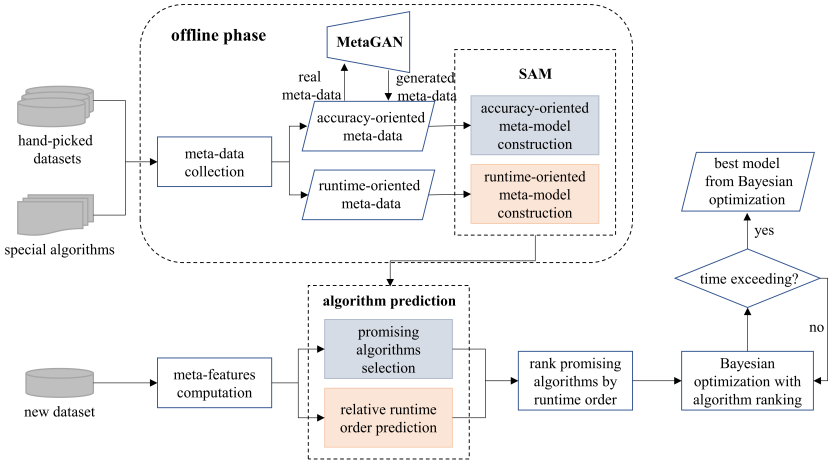


Fig. 1. The framework of Meta-data Augmentation-based Search Strategy (MDASS).

accuracy-oriented meta-data while F and R as runtime-oriented meta-data. For new datasets, most meta-learning methods are to compute the similarity with prior datasets in meta-feature space, or build machine learning model to learn f_d on meta-data and predict the algorithm performance. However, the distribution $p(d)$ is unknown and the hand-picked datasets are hard to represent this distribution. It limits the capacity of meta-learning learning from prior experience. Therefore, the main intention of MetaGAN is to approximate $p(d)$ with GAN and sample new meta-data that belongs to datasets $d^* \sim p(d|F, A)$.

MetaGAN: Conditional WGAN-based Meta-data Augmentation Hand-picked datasets can be viewed as instantiations sampling from unspecified distribution. GAN fits well when augmenting data from datasets sampled without knowing prior distribution. Since unconditional GAN has no control for the generated data mode, MetaGAN is based on conditional setting. Fig. 2 shows the process of MetaGAN. To perform conditional meta-data augmentation, accuracy matrix is first labeled through the following steps: i) Each entry A_{ij} is normalized by $A^*_{ij} = \max\{A_{ij}\}_{j=1}^{n} - A_{ij}$. ii) The normalized accuracy is labeled by predefined threshold ξ. When A^*_{ij} is greater than ξ, it is labeled as $True$, representing algorithm a_j on dataset d_i has promising performance and vice versa.

Afterwards, MetaGAN is conditioned on class to direct accuracy-oriented augmentation under WGAN framework. It is formulated as two adversarial model: meta-generator G and meta-discriminator D. G implicitly learns the distribution of dataset from input meta-data and transforms a prior noise distribution $p_z(z)$ into meta-features space of datasets drawn from the learned distribution. D estimates the Wasserstein distance explicating the minimum cost of transporting mass for transforming the dataset distribution $p(d)$ into the learned distribution of generator p_g. Thus, the objective function of MetaGAN can be

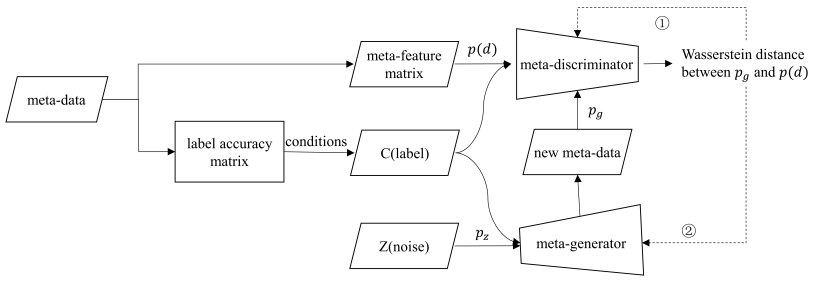


Fig. 2. The process of MetaGAN. It is built with conditional WGAN through gradient descent method. ① represents $\nabla_{\theta_d}\left[\frac{1}{m}\sum_{i=1}^{m} D\left(x^i, c^i; \theta_d\right) - \frac{1}{m}\sum_{i=1}^{m} D\left(G\left(z^i\right), c^i; \theta_d\right)\right]$, and ② represents $-\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m} D\left(G\left(z^i\right), c^i; \theta_g\right)$.

defined as Eq. 1, which also reflects the quality of meta-data augmentation.

$$V(G, D) = \mathrm{E}_{x\sim p(d)} D(x, c) - \mathrm{E}_{G(z)\sim p_g} D(G(z), c) \tag{1}$$

where c is specific class of condition. G and D are trained simultaneously, alternating between the training phases of them as optimizing their parameters shown in Fig. 2. When MetaGAN has been trained, meta-data can be collected with an insignificant cost, bounded only by the computational overhead of MetaGAN.

2.2 Self-Adaptive Meta-model for algorithm selection

The intention of MDASS experienced by users is to efficiently obtain high-performance learning model. Therefore, it requires the algorithms with higher performance but lower runtime in optimization process. We propose SAM to solve these two meta-tasks: high-accuracy algorithm selection and runtime order prediction. It can carry out the trade-off of learning performance and runtime to provide more time for promising algorithms in evaluation process. The key problem for SAM is to achieve global optimum under different algorithm prediction. Since there exists no such universal learner outperforming other learners consistently, different algorithm prediction should be considered separately.

For accuracy-oriented meta-model, it is built on augmented accuracy-oriented meta-data to predict whether an algorithm has promising performance. Therefore, it can be viewed as multi-output classification problem. We consider the classification learners with best performance in accuracy matrix, and base learners of each output is from these learners. The objective function is defined as Eq. 2, which can be achieved by minimizing the loss of each output greedily.

$$f_j^{acc} = \underset{f_j^{acc}\in\Gamma^{cla}}{\arg\min} \frac{1}{m}\sum_{i=1}^{m} \left| f_j^{acc}(F_i) - A_{ij} \right| \tag{2}$$

where f_j^{acc} represents the base learner of each algorithm selection and Γ^{cla} represents the collection of classification learners. For runtime-oriented meta-model, it is built on runtime-oriented meta-data to predict algorithm runtime ranking. Similar to accuracy-oriented meta-model, this meta-model can be viewed as multi-output regression problem, and we consider regression learners from three different categories including ridge regression, support machine vector regression and random forest for base learners. Then, this meta-model can be achieved by minimizing the objective function as Eq. 3.

$$f_j^{time} = \underset{f_j^{time} \in \Gamma^{reg}}{\arg\min} \sum_{i=1}^{m} \left(f_j^{time}(F_i) - R_{ij} \right)^2 \tag{3}$$

where f_j^{time} represents the base learner of each runtime prediction and Γ^{reg} represents the collection of regression learners.

Therefore, SAM applys appropriate learners to adapt each algorithm prediction. When the SAM trained, MDASS can evaluate high-performance algorithms in the order of runtime ranking for new datasets. Notably, SAM is built on augmented meta-data through MetaGAN and this process is performed offline only. Runtime on it does not increase the runtime of MDASS employed by users, bounded only with the automated model selection stage for new given datasets.

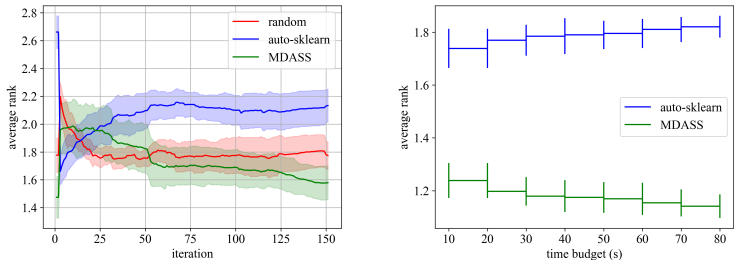
3 Experiments

3.1 Generation of meta-data

We ran all experiments on 405 classification datasets sorted by most runs in OpenML [20]. They were selected by filtering with no more than 10000 samples and 300 features. The same preprocessings were applied to all datasets: removing missing value, one-hot encoding for categorical features and standardizing all features. All involved algorithms were implemented by scikit-learn [17]: Kernel SVM, Linear SVM, Gaussian naive Bayes, Bernoulli naive Bayes, k-Nearest Neighbor (KNN), Decision Tree, Random Forest, Adaboost, Stochastic Gradient Descent (SGD), Latent Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Passive Aggressive, and Extra Tree. We generated accuracy matrix through running Bayesian optimization implemented in Hyperopt [3] to search best performance for a specific algorithm on each dataset and we set the labeled threshold as 0.025. Moreover, the meta-features used in experiments are divided into five groups: simple, statistical, information-theoretic, complexity and landmarkers. More details of meta-feature and algorithm hyperparameter space are listed in supplementary material. We only used six meta-features which are more relevant to runtime prediction for runtime-oriented meta-model: number of instances, number of features, dataset ratio and their log values.

3.2 Experimental Setup

In our experiments, hold-out-validation is used to evaluate MDASS, and repeated 10 times. Therefore, out of the 405 total datasets, 10% ($\approx$ 40) were



(a) iteration (b) time budget

Fig. 3. Comparison of random method and auto-sklearn in a time-constrained setting. (a) the average rank with the increasing of iteration, 1 is the best and 3 is the worst. (b) the average rank with different time budget, 1 is the best and 2 is the worst.

identified to comprise the held-out test set and the remaining datasets were to build MDASS in each hold-out-validation. Six base learners were to formulate accuracy-oriented meta-model: Adaboost, Extra Tree, Random Forest, Decision Tree, Kernel SVM with the highest performance in accuracy matrix and the commonly used learner in literature [5, 13], KNN. For generator and discriminator of MetaGAN, they both had same neural architecture: Multi-layer Perceptron with two hidden layers, and ReLU activation function was applied for all layers except the output layer which was linear combination. For hidden layers, we used dropout strategy with the probability of 0.2. Besides, MetaGAN used RMSProp optimizer with global learning rate of 1e-4 and decay rate of 0.9. We set the weight clipping parameter as 0.01. We augmented the number of meta-data to approximate 600 though MetaGAN. All procedures are ran on Windows server with Intel(R) Xeon(R) CPU E5-2609 v4 @1.79GHZ processor.

3.3 Experimental Evaluation and Analysis

The performance validation of MDASS When tackling a specific domain problem through machine learning, many users tend to quickly construct learning model to mining the knowledge in domain data. MDASS provides a novel AutoML system to efficiently obtain high-quality model with an automatic manner for given datasets. Therefore, it can alleviate the challenge of model design.

In order to evaluate the effectiveness of MDASS, we compare it with the following baselines: i) random: randomly searching algorithm and hyperparameters to evaluate for each dataset in test set. ii) auto-sklearn [5]: Since ensemble and preprocessing are not involved in MDASS, we disabled automated ensemble construction and preprocessing in auto-sklearn. Fig. 3 shows the average rank of these methods from optimization iteration and time budget within a time-constrained setting. From Fig. 3(a), the average rank of MDASS is decreasing consistently, while auto-sklearn is increasing. Besides, auto-sklearn is the best at the beginning 10 iteration, but MDASS can surpass auto-sklearn

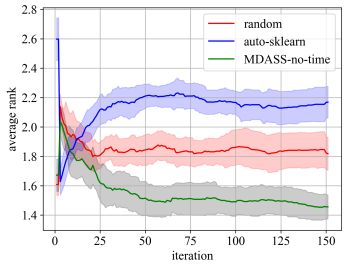


Fig. 4. Comparison of random method and auto-sklearn without time-constrain setting

quickly and become the optimal about 52 iteration. From Fig. 3(b), MDASS is better than auto-sklearn consistently and the performance of MDASS turns better with the increasing of time budget. We can conclude that MDASS performs surprisingly well than other methods with time budget, since random performs optimization in all configuration space and auto-sklearn selects multiple hyperparameter configurations to warm-start optimization according to the distance between datasets, while MDASS only optimizes in promising model space, i.e., higher accuracy and lower runtime algorithms. In addition, Fig. 3 also illustrates that MDASS can search better individual model than auto-sklearn and random method since we both disable ensemble construction in these methods.

In addition, we disable the time-constrain setting and directly joint optimization (CASH) is applied to the promising algorithms with corresponding hyperparameters. The result is shown in Fig. 4. We can observe that MDASS turns lowest in earlier phase and the average rank of MDASS drops faster than that with time-constrain setting, surpassing random method from the beginning of optimization consistently. When considering time budget, the initial evaluated algorithm may not be the best model, which consume a few iterations.

Why dose MDASS work well? MDASS performs well in comparison with random and auto-sklearn method. In this section, we execute ablation studies of MDASS to interpret why MDASS works well and whether its components: MetaGAN and SAM are reasonable.

The improvement of MetaGAN and SAM. MDASS contains two main components: MetaGAN and SAM. In order to evaluate the improvement of MDASS, we compare four variants of our method: original version without MetaGAN and SAM (Original), MetaGAN only, SAM only and combining MetaGAN and SAM on the following five metrics: 1) Single Best Rate (SB): overlap of one best algorithms, i.e. the best algorithm in predicted promising algorithms of SAM. 2) Two Best Rate (TB): overlap of two best algorithms. 3) Single Worst Rate (SW): overlap of one worst algorithm. 4) Global Accuracy (GA): average accuracy of each algorithm prediction. 5) Global BER (GB): the balanced error rate (the average of false negative and false positive rates) of each algorithm prediction.

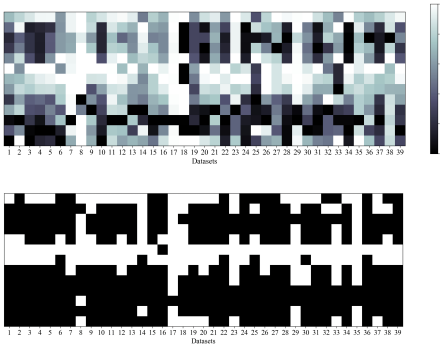


Fig. 5. Algorithm performance heatmap. Top: Each block is colored by the performance of each algorithm on 39 datasets. The lighter block represents the performance of corresponding algorithm is better, and vice versa. Bottom: the selection result of SAM on these 39 datasets. The white block represents the algorithm selected by SAM while the black block represents the algorithm is not selected as promising one. The algorithms represented by y-axis from bottom to top in these two figures are Kernel SVM, Linear SVM, Gaussian naive Bayes, Bernoulli naive Bayes, KNN, Decision Tree, Random Forest, Adaboost, SGD, LDA, QDA, Passive Aggressive, and Extra Tree.

For these metrics, the greater values of SB, TB, GA and lower values of SW, GB indicate better performance of each component. Here, we evaluate the accuracy-oriented meta-model (promising algorithm selection), and the performance of runtime-oriented meta-model is evaluated in the following experiment. We set base learner of Original and MetaGAN variants as Random Forest.

Table 1 shows the comparison of these methods. We observe that MetaGAN+SAM achieves superior performance over all variants. Compared with Original, the SB, TB, GA of MetaGAN+SAM have increased by 0.074, 0.057, 0.035 and SW, GB have reduced by 0.031, 0.018 respectively, which reveals our approach can find out more promising algorithms. Besides, both two components can yield improvement over Original. MetaGAN can improve meta-learning to search more promising algorithms. Compared with Original, SB, TB and GA are increased from 0.795, 0.879 and 0.788 to 0.828, 0.918 and 0.796 while SW and GB are reduced from 0.113 and 0.105 to 0.085 and 0.098 respectively, which demonstrates the promise of GAN to meta-learning. SAM adapts meta-data using multi-learners for each algorithm selection instead of single learner for all outputs, which also improves the performance of MDASS. Compared with Original, SB, TB and GA have increased by 0.026, 0.021 and 0.014 while SW has reduced by 0.013. Fig. 5 shows algorithm selection performance of MDASS on 39 datasets of meta-test set in one of 10 hold-out-validation. We can observe that the lighter algorithms (top) have larger probability to be selected (bottom).

Moreover, in order to demonstrate the superiority of GAN, we compare MetaGAN with the following synthetic data methods: 1) Resample: random sampling from origin meta-data repeatedly. 2) SMOTE [4]: random resampling for one

Table 1. Four variants compasion on held-out test set.

Variants	Metrics				
	SB(↑)	SW(↓)	TB(↑)	GA(↑)	GB(↓)
Original	0.795±0.044	0.113±0.042	0.879±0.057	0.788±0.021	0.105±0.017
MetaGAN	0.828±0.030	0.085±0.036	0.918±0.050	0.796±0.020	0.098±0.017
SAM	0.821±0.055	0.100±0.047	0.900±0.053	0.802±0.019	0.101±0.016
MetaGAN+SAM	**0.869±0.051**	**0.082±0.047**	**0.936±0.052**	**0.823±0.021**	**0.087±0.018**

Table 2. The comparison of different synthetic data.

Methods	Metrics				
	SB(↑)	SW(↓)	TB(↑)	GA(↑)	GB(↓)
Resample	0.785±0.066	0.090±0.015	0.879±0.051	0.794±0.051	0.102±0.015
SMOTE	0.797±0.053	0.121±0.067	0.882±0.040	0.792±0.022	0.099±0.020
ADASYN	0.800±0.045	0.128±0.018	0.885±0.048	0.794±0.022	0.100±0.018
MetaGAN	**0.828±0.030**	**0.085±0.036**	**0.918±0.050**	**0.796±0.020**	**0.098±0.017**

class and SMOTE for another, and vice versa. 3) ADASYN [10]: random resampling for one class and ADASYN for another, and vice versa. The result is shown in Table 2. Resample method only generates same meta-data, which is easy to introduce more variance for algorithm selection. Thus, it is uncompetitive with MetaGAN, even with Original. Since SMOTE and ADASYN are both based on the distance of instances in meta-data, they perform poorer than MetaGAN on all metrics, even for comparison to Resample method with higher SW and GB which are both more than 0.12. MetaGAN generates meta-data through learning the distribution of origin meta-data, maintaining the highest performance and lower variance (more robust for meta-learning) in four competing approach.

The runtime prediction performance of SAM. In order to evaluate the runtime prediction performance in MADSS, we compare the runtime-oriented meta-model in SAM and the polynomial regression models with the factor of 2 to 4 using in OBOE [21] on algorithm runtime prediction. In our experiment, the polynomial regression with the factor of 3 and 4 performs much worse than the factor of 2. Therefore, we only present the performance of the model with factor of 2. Table 3 show the runtime prediction accuracy (the distance of actual time and predicted time less than 1 second) of these method. We can observe that SAM performs better than OBOE in 9 algorithms runtime prediction and the accuracy of other algorithm prediction is similar with OBOE. Although runtime prediction for machine learning algorithm is difficult to predict, the performance of SAM on multiple algorithms is a roughly good prediction and the predicted runtime ranking for MDASS is acceptable.

4 Conclusion

Meta-learning is widely used in AutoML. However, collecting sufficient meta-data with high quality is difficult. In this work, we incorporate GAN with meta-learning for meta-data augmentation, and propose Meta-data Augmentation-based Search Strategy (MDASS) for AutoML model selection, which is composed of Meta-GAN Surrogate Model (MetaGAN) and Self-adaptive Meta-model

Table 3. Runtime prediction accuracy.

Algorithm Type	Methods	
	with factor of 2	SAM
Kernel SVM	0.846±0.055	**0.872±0.053**
Linear SVM	0.635±0.034	**0.685±0.050**
Gaussian naive Bayes	0.972±0.027	**0.977±0.021**
Bernoulli naive Bayes	0.982±0.020	**0.990±0.012**
KNN	**0.967±0.023**	0.967±0.033
Decision Tree	0.751±0.051	**0.851±0.043**
Random Forest	0.849±0.068	**0.892±0.043**
Adaboost	0.454±0.040	**0.523±0.083**
SGD	0.821±0.070	**0.846±0.068**
LDA	**0.941±0.046**	0.923±0.049
QDA	**0.890±0.043**	0.885±0.043
Passive Aggressive	0.895±0.045	**0.928±0.039**
Extra Tree	**0.951±0.035**	0.946±0.045

(SAM). MetaGAN generates effective meta-data through WGAN as surrogate model. SAM applies hybrid learners to self-adaptively build meta-models to search the algorithms with right trade-off of runtime and accuracy. For a new dataset, MDASS evaluate promising algorithms with Bayesian optimization under time constraint. Our experiments, evaluated on 405 classification datasets from OpenML and 13 classifiers from scikit-learn, show that our proposed method both can yield improvement than comparative methods.

This work demonstrates the promise of GAN for AutoML research. However, there still many lefts to future research. The well-known dilemma is the stable training process of GAN, and the neural architecture of GAN using in MDASS should be elaborate to design. Therefore, one obvious direction is to adapt neural architecture search for GAN. Moreover, MetaGAN can incorporate with semi-supervised learning and be extended to semi-AutoML through the adversarial training of meta-model, generator and discriminator.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (No. 52073169) and the State Key Program of National Nature Science Foundation of China (Grant No. 61936001).

References

1. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein gan. arXiv preprint arXiv:1701.07875 (2017)
2. Asuncion, A., Newman, D.: Uci machine learning repository (2007)
3. Bergstra, J., Yamins, D., Cox, D.D.: Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In: Proceedings of the 12th Python in science conference. vol. 13, p. 20. Citeseer (2013)
4. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research **16**, 321–357 (2002)

5. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In: Advances in neural information processing systems. pp. 2962–2970 (2015)
6. Feurer, M., Springenberg, J.T., Hutter, F.: Initializing bayesian hyperparameter optimization via meta-learning. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
7. Fusi, N., Sheth, R., Elibol, M.: Probabilistic matrix factorization for automated machine learning. In: Advances in neural information processing systems. pp. 3348–3357 (2018)
8. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. pp. 2672–2680 (2014)
9. Gui, J., Sun, Z., Wen, Y., Tao, D., Ye, J.: A review on generative adversarial networks: Algorithms, theory, and applications (2020)
10. Haibo He, Yang Bai, Garcia, E.A., Shutao Li: Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE International Joint Conference on Neural Networks. pp. 1322–1328 (2008)
11. Klein, A., Dai, Z., Hutter, F., Lawrence, N., Gonzalez, J.: Meta-surrogate benchmarking for hyperparameter optimization. In: Advances in Neural Information Processing Systems. pp. 6270–6280 (2019)
12. Klein, A., Falkner, S., Bartels, S., Hennig, P., Hutter, F.: Fast bayesian optimization of machine learning hyperparameters on large datasets. In: Artificial Intelligence and Statistics. pp. 528–536. PMLR (2017)
13. Li, Y.F., Wang, H., Wei, T., Tu, W.W.: Towards automated semi-supervised learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 4237–4244 (2019)
14. Luo, Y., Wang, M., Zhou, H., Yao, Q., Tu, W.W., Chen, Y., Dai, W., Yang, Q.: Autocross: Automatic feature crossing for tabular data in real-world applications. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1936–1945 (2019)
15. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
16. Mısır, M., Sebag, M.: Alors: An algorithm recommender system. Artificial Intelligence **244**, 291–314 (2017)
17. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. the Journal of machine Learning research **12**, 2825–2830 (2011)
18. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 847–855 (2013)
19. Vanschoren, J.: Meta-learning: A survey. arXiv preprint arXiv:1810.03548 (2018)
20. Vanschoren, J., Van Rijn, J.N., Bischl, B., Torgo, L.: Openml: networked science in machine learning. ACM SIGKDD Explorations Newsletter **15**(2), 49–60 (2014)
21. Yang, C., Akimoto, Y., Kim, D.W., Udell, M.: Oboe: Collaborative filtering for automl model selection. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1173–1183 (2019)
22. Zöller, M.A., Huber, M.F.: Benchmark and survey of automated machine learning frameworks. arXiv preprint arXiv:1904.12054 (2019)