AMI-Class: Towards a Fully Automated Multi-view Image Classifier

Mahmoud Jarraya[1], Maher Marwani[1], Gianmarco Aversano[2], Ichraf Lahouli[1], and Sabri Skhiri[2]

[1] EURANOVA TN, Les berges du lac, Tunisia
[2] EURANOVA BE, Mont-Saint-Guibert, Belgium
{mahmoud.jarraya,maher.marwani, gianmarco.aversano, ichraf.lahouli, sabri.skhiri}@euranova.eu

Abstract. In this paper, we propose an automated framework for multi-view image classification tasks. We combined a GAN-based multi-view embedding architecture with a scalable AutoML library, DeepHyper. The proposed framework is able to, all at once, train a model to find a common latent representation and perform data imputation, choose the best classifier and tune all necessary hyper-parameters. Experiments on the MNIST data-set show the effectiveness of our solution to optimize the end-to-end multi-view classification pipeline.

1 Introduction

In real-world, multi-view images are quite common and are often complementary which make them exploited in various computer vision applications such as medical applications, [1], surveillance [2, 3] and agriculture [4]. Multi-view representation learning is concerned with the problem of learning common (latent) representations of multi-view data, usually by either aligning or fusing view-specific latent features. One popular approach, called multi-view embedding (MVE), is to project these views in a common latent space which can then be used to solve several machine learning tasks. However, the remaining challenge is how to deal with missing views. Mapping different views to a common latent space, instead of concatenating, facilitates the process of data imputation. Typical techniques include kernel-based methods such as Deep Canonical Correlation Analysis (DCCA) [5] or non-negative matrix factorization (NMF)-based methods [6]. However, these two kinds of techniques suffer from several limitations, mainly complexity, inefficiency to scale with large scale databases and wastefulness to compensate for missing views due to regularization and added constraints [7] which makes them less attractive. Recently, generative models such as VAEs [8] or GANs [9] have gained popularity in this field [7, 10] because of their ability to map any given distribution to a target one (e.g. the distribution of the missing data) and their effectiveness in narrowing the difference between the distributions of different views [11]. For instance, the implementation of a multi-modal adversarial representation network (MARN) with an

attention mechanism has led to state-of-the-art performance for click-through rate prediction [12]. For these reasons, the present work proposes a GAN-based architecture for MVE, inspired by [7, 10], that is able to impute missing views.

From a machine learning (ML) perspective, we have recently seen the emergence of frameworks, called AutoML, that automate the whole pipeline, starting from the feature extraction to the model deployment. AutoML attempts to solve either Neural Architecture Search (NAS) problem or a Combined Algorithm Selection and Hyper-parameter tuning (CASH) problem, i.e. find the best configuration for a ML program, within limited computational budget [13]. For image classification tasks, we do not want to rely only on deep neural architectures as DL techniques represent only a subset of machine learning methodologies and some classic classifiers such as K-nearest neighborhood might perform well with certain datasets. Thus, in this paper, we will focus on resolving the CASH problem. The optimization process over an algorithm's hyper-parameters can be black-box (e.g. random-search, grid-search or model-based) or multi-fidelity [14]. Despite the increasing research efforts in AutoML [15, 16], there are still several open challenges such as scalability, data preparation, choice of optimization techniques and multi-modality of data [13]. For instance, Auto-Sklearn [15] introduced meta-learning and ensemble methods to improve the performance of vanilla SMAC[17] optimization, however it is built on Scikit-Learn which makes it unsuited to multi-modal/view data. However, as far as we know, even the recent autoML frameworks such as DeepHyper [18] or VEGA [19] can not handle multi-view datasets.

Through this work, we propose a multi-view CASH solver, called AMI, at its basic definition (i.e what classifier to select and what are the adequate hyper-parameters values). Indeed, the main contributions of this work are (1) the implementation of a modified version of a GAN-based MVE architecture for the visual feature extraction and data imputation based on [7] (2) the MVE model's integration in the mono-modal autoML framework i.e DeepHyper [18], resulting in an end-to-end pipeline that can be optimized all at once using an asynchronous Bayesian Optimization technique. Consequently, the best architecture for feature extraction from the multi-view images and their embedding in a common latent space and the best classifier are all selected with their optimal hyper-parameters.

2 PROPOSED METHODOLOGY

Our solution consists of two main components: the first one ensures the MVE while the second one solves the classification task within a CASH problem by integrating the MVE model architecture within DeepHyper.

2.1 Notations

For the sake of clarity, we start by formalising the input data to be used for the rest of the paper. The multi-view data is represented by:
$\mathbf{X} = \left\{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \cdots, \mathbf{X}^{(V)}\right\}$ where $\mathbf{X}^{(v)} = \left\{x_1^{(v)}, x_2^{(v)}, \cdots, x_N^{(v)}\right\} \in \mathbf{R}^{N \times d_v} \forall v \in$

$[1, V]$, V is the number of views, N is the number of samples, and d_v is the feature dimension of v-th view. d is the feature dimension of the common latent space. Besides, since our data might have missing views that need to be imputed, we split our data into 2 subsets: paired data $\left\{x^{(1)}, x^{(2)}, \cdots, x^{(V)}\right\}$, in which all views exist, and unpaired data, $\left\{x'^{(1)}, x^{(2)}, \cdots, x'^{(V)}\right\}$, in which at least one view is missing. We denote by $r \in [0, 1]$ the missingness ratio as the percentage of the samples with missing data according to the total number of samples N (e.g if $r = 0.1$ so 10% of the samples have at least one missing view). We denote by $\mathcal{Y}$ the labels space and each $x_i^{(v)} \forall v \in [1, V]$ and $\forall i \in [1, N]$ is associated to $y_i \in \mathcal{Y}$.

2.2 MVE component

Architecture The purpose of this component, denoted by $\mathcal{M}$, is to derive a representative feature vector in a common latent space of the multi-view input data. This can be achieved by relying on an adapted version of the architecture proposed in [7] where we have removed the clustering layer, changed the fusion one and forced the alignment between the view-specific feature vectors. The MVE component architecture consists of four networks:

Encoder $\{E_v\}_{v=1}^{V}$: $R^{d_v} \rightarrow R^d$.
V encoders, projecting each input view $X^{(v)}$ to a corresponding view-specific subspace through many stacked convolution-batch normalization-ReLu layers. Each encoder outputs a d-dimensional vector $Z^{(v)}$.

Fusion F: $R^{d \times V} \rightarrow R^d$.
The intuition from the fusion network is to capture the shared semantics of the multiview data from the resulting V view-specific representation vectors. It takes the output of the encoders and derives the target feature vector through a fully connected layer.

Decoder $\{G_v\}_{v=1}^{V}$: $R^d \rightarrow R^{d_v}$.
Each G_v has a mirrored architecture to the E_v. It takes a vector from the common sub-space and generates the corresponding view. It acts both as decoder and a generator.

Discriminator $\{D_v\}_{v=1}^{V}$: $R^{d_v} \rightarrow \{0, 1\}$.
Each D_v takes a sample from its corresponding view distribution, which can be either the generated view $\tilde{x}_i^{(v)}$ by G_v or the real view $x_i^{(v)}$, and outputs the probability for this sample to be real, using a stack of convolution-batch normalization-LeakyReLu layers and a final *sigmoid* activation.

Objective Function The objective function we tend to optimize includes two terms; one to train the AE networks, and the other for the GANs networks.

Auto-Encoder loss :
The purpose is that the multi-view feature vector is representative enough and holds the cross-view information. The reconstruction loss is defined as follow:

$$L_{rec} = \Sigma_{v=1}^{V} \|X^{(v)} - G_v(Z_v)\|^2 \tag{1}$$

with Z_v being the latent vector of the v-th view. Each decoder $\{G_v\}_{v=1}^{V}$ has to generate its corresponding view from a sample $\{z^{(v)}\}_{v=1}^{V}$ of view-specific latent space. However, we want to fuse those into a common latent space. Thus, we also minimize the distance between the different $\{z^{(v)}\}_{v=1}^{V}$ using the following alignment loss:

$$L_{align} = \Sigma_{v=1}^{V} \Sigma_{v\prime>v}^{V} \|E_v(X^{(v)}) - E_v\prime(X^{(v\prime)})\|^2 \tag{2}$$

Therefore, the final Auto-Encoder loss is:

$$L_{AE} = L_{rec} + a * L_{align} \tag{3}$$

with $a \geq 0$ being a hyper-parameter. This loss is used to train $\{E^{(v)}\}_{v=1}^{V}$, $\{G^{(v)}\}_{v=1}^{V}$ and F, differently from [7] where a different loss is used to F. During the first few epochs when minimizing Eq. (3), the fusion layer F is not used. The views are reconstructed by the decoders, which take the resulting vector from the fusion operation of $\{Z^{(v)}\}_{v=1}^{V}$. Therefore, the training of the MVE component as Auto-Encoder on minimizing L_{AE} is not enough in the case of unpaired data where the fusion can not be realized. To tackle this, we refine the model by means of adversarial training.

Adversarial Training loss :
We employ this loss to train the GANs' part of the model, which is composed of the decoders $\{G_v\}_{v=1}^{V}$ and the discriminators $\{D_v\}_{v=1}^{V}$. The goal of $\{G_v\}_{v=1}^{V}$ is to fool $\{D_v\}_{v=1}^{V}$ by producing realistic images from vectors from common space distribution while the goal of $\{D_v\}_{v=1}^{V}$ is to distinguish whether the input is real or generated (fake). The GAN loss is expressed as follow:

$$L_{GAN} = \Sigma_{v=1}^{V} [\log D_v(X^{(v)}) + \log (1 - D_v \circ G_v(Z_v))] \tag{4}$$

In fact, the generators learn to map a given latent vector to a sample from its target distribution. Yet, the generated sample should be paired with the existing one. Therefore, we add the cycle loss defined in Eq. (5).

This way, $\{G_v\}_{v=1}^{V}$ are trained also on imputing data from the existing one.

$$L_{cyc} = \Sigma_{v=1}^{V} \Sigma_{v\prime\neq v}^{V} \|X^{(v)} - G_v \circ E_{v\prime} \circ G_{v\prime} \circ E_v(X^{(v)})\| \tag{5}$$

Also, the final adversarial training loss is:

$$L_{AT} = L_{cyc} + L_{GAN} \tag{6}$$

Total loss :

$$L = \lambda_{AE}\, L_{AE} + \lambda_{AT}\, L_{AT} \tag{7}$$

Implementation We employ a two-stage training procedure where we first pre-train the AE networks on paired data, and then we train all the model components on both paired and unpaired data.

Step 1 : The paired data is loaded to $\{E_v\}_{v=1}^{V}$ to get V vectors in the latent space. First, these vectors are passed directly to their corresponding $\{G_v\}_{v=1}^{V}$ to reconstruct the input. This way, thanks to Eq. (2), the different vectors are forced to be close to each other in the latent spaces. Then, after a portion ϕ of the total number of epochs, the V vectors are fused by F before being passed to $\{G_v\}_{v=1}^{V}$. Finally, we update the networks by Eq. (3).

Step 2 : We train $\mathcal{M}$ on all the data. In the case of paired data, the input is projected by $\{E_v\}_{v=1}^{V}$ to the latent space, and regenerated by $\{G_v\}_{v=1}^{V}$. These fake views with the real ones are passed to the discriminator then to compute the adversarial loss with Eq. (4) . When encountering unpaired data, we take one of the one existing view each time, project it to the latent space and input the resulting vector to the generator corresponding to the missing view to perform the imputation. Therefore, we compute the cycle consistency loss (5). Finally, we update all the model's parameters by Eq. (7).

2.3 multi-view CASH solver AMI

Architecture DeepHyper [18] is a scalable package for hyper-parameter search, with a generic interface between an asynchronous model-based search method (AMBS) using Bayesian Optimization (BO) ideas (a dynamically updated surrogate model that tries to learn the relationship between the hyper-parameter configurations and their validation errors), and parallel task execution engines. DeepHyper offers: a portable workflow system for parallel, asynchronous evaluations of hyper-parameter configurations at HPC scale; the use of AMBS which has proved to be superior to batch-synchronous methods [18]; a flexible and generic interface that allows for easy extension to a multi-modal framework as DeepHyper is not tied to any ML libraries that can only work on a single node. Besides, as this work combines DeepHyper with a flexible multi-modal embedding model, the problem of data preparation is solved by tuning the hyper-parameters of this model. At First, we have the MVE component's hyper-parameters. This includes: $\mathcal{M}$ HP (AE number of layers, common latent space's dimension d, ϕ etc.) and the training HP (batch size, learning rate, number of epochs, etc). In the other hand, we have a set of hyper-parameters related to the classification problem. We also introduced classification model as a categorical hyper-parameter. Therefore, the hyper-parameters of the whole pipeline are

optimized simultaneously using the AMBS technique of DeepHyper. The classification accuracy is the objective that we tend to maximize. We illustrate our proposed architecture in Fig. 1.

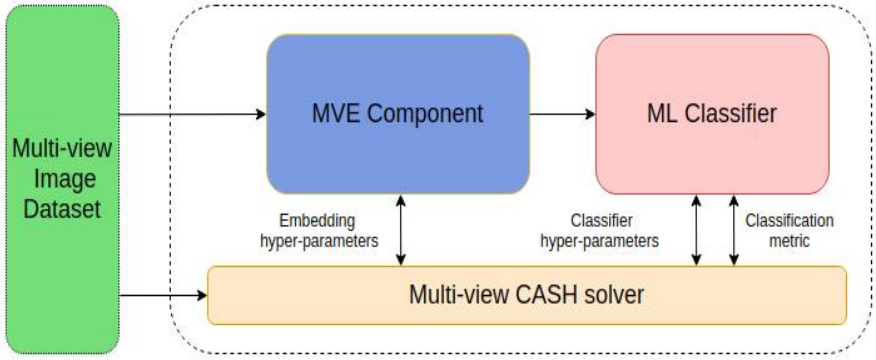

Fig. 1. Proposed End-to-end Multi-view CASH Solver AMI

Implementation First, we split the data into train and validation, we define the search space ζ and the objective function we tend to optimize and also the budget (i.e max number of iterations of the optimization process). Then, at the beginning of each iteration and based on the previous iterations objective, a new configuration c from ζ is sampled. We pick $\mathcal{M}$ HP from c, train it and extract the multi-view data representation for both train and validation in the latent space. After that, we train the selected classifier on the obtained vectors and compute the accuracy on the validation set. Finally, we save the weights of the trained $\mathcal{M}$ whenever a new best objective is achieved. At the end of the process, in addition to the returned best configuration, we obtain a file containing the different configurations associated with their objective.

3 Experiments and Results

In order to validate the aforementioned methodology, we first validate the GAN-based MVE architecture for a classic classification task using the MNIST dataset. To get the multi-view aspect, we are considering the original digits' images as the first view while we compute the corresponding edge images to simulate the second view like done in [10, 7]. Secondly, we aim to prove that the use of DeepHyper can solve the CASH problem by substituting the work of an expert data-scientist and offering the best configuration of the MVE architecture and of the classifier at once.

3.1 MVE Component Validation

We quantitatively test the robustness of the MVE component to deal with missing data by measuring the classification accuracy with respect to different missingness ratios. Indeed, we run $\mathcal{M}$ five times with the same configuration except for the missing data proportion r in training and testing sets. At each run, we step up r by 0.2, train $\mathcal{M}$ on the training set, extract the features representations in the common latent space for both training and testing sets and then train and evaluate them on the multi-classification task using several machine learning models. The results of the classification performance of the different experiences are summed up in Fig. 2.

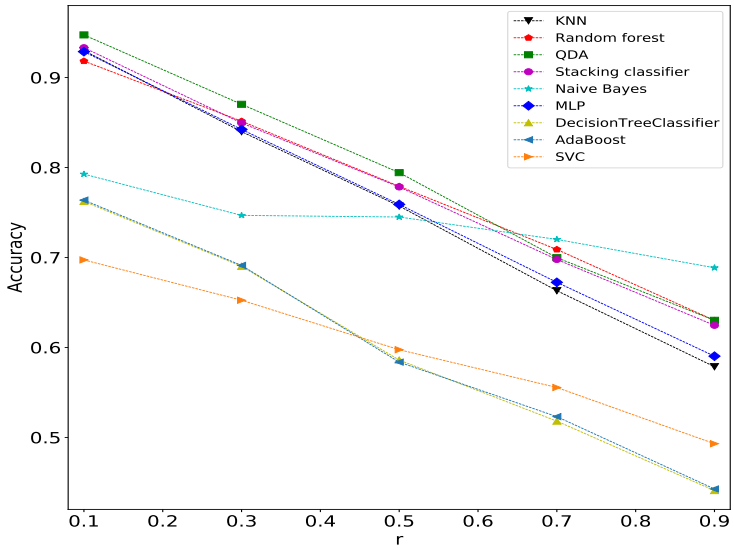

Fig. 2. Classification Accuracy w.r.t the Missingness Ratio r.

Despite the fact that the classifiers have not been tuned, most of them achieved more than 92% of accuracy when r is at its least value. However, this metric drops linearly when r increases which gives an insight on how the missing data could affect the MVE model. From these preliminary results, We can also notice that, in the worst case (i.e $r = 0.9$), the accuracy didn't get lower than 0.4 while it exceeds 0.6 with most of the classifiers. Also, these classifiers perform somehow similarly. Thus, we can not determine which is the best one especially if they have not been tuned yet. As a result from the foregoing, the use of a CASH solver is a crucial step.

3.2 CASH Solver Validation

We are checking if AMI can substitute data-scientist interventions and solve the CASH problem by offering the best configuration of the MVE architecture and of the classifier at once. First, we test its effectiveness to automatically tune the hyper-parameters and select the best classifier in a simultaneous way. To achieve that, we design the search space by integrating hyper-parameters from the MVE component and also a bunch of classifiers with their specific hyper-parameters. Then, we let DeepHyper explore, exploit and evaluate the sampled configuration by measuring its classification accuracy (r is fixed to 0.1). The accuracy results of the exploration process done by DeepHyper are exposed in Fig. 3.

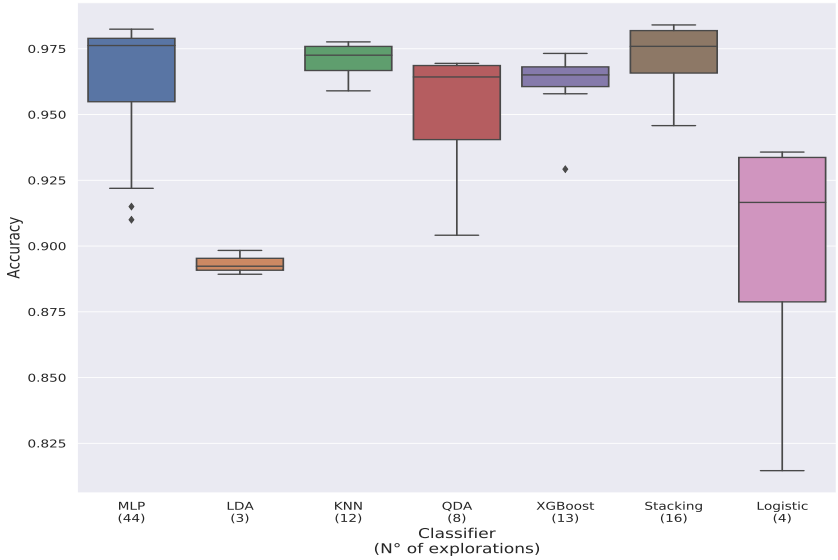

Fig. 3. AMI Explorations for Solving the CASH Problem.

We notice that, despite the large search space, AMI has examined favorable, if not the best, configurations for the most of the classifiers. Also, the more explored classifiers, the better it performs. For instance, MLP and the Stacking classifier achieved more than 98% accuracy, which assert that the optimization process is doing well.

Table 1 reports a comparison between AMI's solution (MVE+Stacking classifier) with two different MVE models: a default-parameters model (non-tuned one following some parameters given by [7]) and a $\mathcal{M}$ model which has been extensively tuned by a human expert via a trial and error process. The missingness ratio r is 0.1 for the three aforementioned experiments. First, we notice that AMI's returned solution surprisingly outperformed the human-tuned solutions. Second, this solution indicates that the stacking classifier is the best for

Table 1. Classification performance comparison of the 3 best classifiers chosen and tuned by AMI and the same models when they are either non-tuned or extensively-tuned.

Default-values	**Extensive Tuning**	**AMI Tuning**
94.57	97.60	**98.40**

this classification problem, which is the best classifier found by human tuning. Therefore, we conclude that the way we have configured DeepHyper leads to a reliable tool that can suggest the best classifier and automate the tuning of the hyper-parameters of the end-to-end pipeline. Indeed, AMI outputs the optimal hyper-parameters of the MVE $\mathcal{M}$ model and the ones for the Stacking classifier.

4 Conclusion & Discussion

In this paper, we propose a multi-view CASH solver for an image classification task which outputs the optimal classifier with the adequate hyper-parameters. In the first hand, we have suggested a GAN-based MVE architecture which takes as input multi-view images and embed them in a common latent space while imputing the missing view if any. In the second hand, we combined this architecture with a scalable mono-view AutoML library called DeepHyper. The proposed framework is able to, all at once, train the multi-view embedding model, automatically choose the best classifier and tune all the hyper-parameters including the ones related to the neural networks that extract the visual high-level features. Consequently, the end-to-end pipeline is flexible and adaptable to data variety (image resolution, number of views etc.) and can even be extensible to other modalities such as text or audio. In future works, we aim to let DeepHyper choose between several MVE techniques such as graph-based or kernel-based. We also tend to support supplementary autoML features such as data pre-processing (e.g data cleaning, data augmentation), to support meta-learning to benefit from prior knowledge (e.g., a warm-start could be valuable in reducing the optimization process time) and also support NAS, to further parameterize the neural-based components.

References

1. Hasan Nasir Khan, Ahmad Shahid, Basit Raza, Amir Dar, and Hani Alquhayz, "Multi-view feature fusion based four views model for mammogram classification using convolutional neural network," *IEEE Access*, vol. PP, pp. 1–1, 11 2019.
2. Lichen Wang, Zhengming Ding, Zhiqiang Tao, Yunyu Liu, and Yun Fu, "Generative multi-view human action recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6212–6221.
3. Yue Bai, Zhiqiang Tao, Lichen Wang, Sheng Li, Yu Yin, and Yun Fu, "Collaborative attention mechanism for multi-view action recognition," *arXiv preprint arXiv:2009.06599*, 2020.

4. Puneet Mishra, Ittai Herrmann, and Mariagiovanna Angileri, "Improved prediction of potassium and nitrogen in dried bell pepper leaves with visible and near-infrared spectroscopy utilising wavelength selection techniques," *Talanta*, vol. 225, pp. 121971, 2021.
5. Zhongkai Sun, Prathusha Sarma, William Sethares, and Yingyu Liang, "Learning relationships between text, audio, and video via deep canonical correlation for multimodal language analysis," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.
6. N. Rai, S. Negi, S. Chaudhury, and O. Deshmukh, "Partial multi-view clustering using graph regularized nmf," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2016, pp. 2192–2197.
7. Qianqian Wang, Zhengming Ding, Zhiqiang Tao, Quanxue Gao, and Yun Fu, "Generative partial multi-view clustering," 2020.
8. Samuel K Ainsworth, Nicholas J Foti, and Emily B Fox, "Disentangled VAE representations for multi-aspect and missing data," *arXiv preprint arXiv:1806.09060*, 2018.
9. Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath, "Generative adversarial networks: An overview," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018.
10. Chao Shang, Aaron Palmer, Jiangwen Sun, Ko-Shin Chen, Jin Lu, and Jinbo Bi, "Vigan: Missing view imputation with generative adversarial networks," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 766–775.
11. Y. Li, M. Yang, and Z. Zhang, "A survey of multi-view representation learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 10, pp. 1863–1883, 2019.
12. Xiang Li, Chao Wang, Jiwei Tan, Xiaoyi Zeng, Dan Ou, Dan Ou, and Bo Zheng, "Adversarial multimodal representation learning for click-through rate prediction," *Proceedings of The Web Conference 2020*, Apr 2020.
13. Quanming Yao, Mengshuo Wang, Yuqiang Chen, Wenyuan Dai, Hu Yi-Qi, Li Yu-Feng, Tu Wei-Wei, Yang Qiang, and Yu Yang, "Taking human out of learning applications: A survey on automated machine learning," *arXiv preprint arXiv:1810.13306*, 2018.
14. R Elshawi, M Maher, and S Sakr, "Automated machine learning: state-ofthe-art and open challenges," *arXiv preprint arXiv:1906.02287*, 2019.
15. Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter, "Auto-sklearn 2.0: The next generation," *arXiv preprint arXiv:2007.04074*, 2020.
16. Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown, "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*, 2013, pp. 847–855.
17. Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *International conference on learning and intelligent optimization*. Springer, 2011.
18. P. Balaprakash, M. Salim, T. D. Uram, V. Vishwanath, and S. M. Wild, "Deephyper: Asynchronous hyperparameter search for deep neural networks," in *2018 IEEE 25th International Conference on High Performance Computing (HiPC)*, 2018.
19. Bochao Wang, Hang Xu, Jiajin Zhang, Chen Chen, Xiaozhi Fang, Ning Kang, Lanqing Hong, Wei Zhang, Yong Li, Zhicheng Liu, et al., "Vega: Towards an end-to-end configurable automl pipeline," *arXiv preprint arXiv:2011.01507*, 2020.