IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 3, 2020, accepted March 11, 2020, date of publication March 13, 2020, date of current version March 26, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2980798

FrImCla: A Framework for Image Classification Using Traditional and Transfer Learning Techniques

MANUEL GARCÍA-DOMÍNGUEZ, CÉSAR DOMÍNGUEZ, JÓNATHAN HERAS, ELOY MATA, AND VICO PASCUAL

Department of Mathematics and Computer Science, University of La Rioja, 26004 Logroño, Spain

Corresponding author: Manuel García-Domínguez (manuel.garciad@unirioja.es)

This work was supported in part by the Ministerio de Economía, Industria y Competitividad under Grant MTM2017-88804-P, in part by the Agencia de Desarrollo Económico de La Rioja under Grant ADER-2017-I-IDD-00018, and in part by the Formación de Personal investigador (FPI) Grant through the Community of La Rioja 2018.

ABSTRACT Deep learning techniques are currently the state of the art approach to deal with image classification problems. Nevertheless, non-expert users might find challenging the use of these techniques due to several reasons, including the lack of enough images, the necessity of trying different models and conducting a thorough comparison of the results obtained with them, and the technical difficulties of employing different libraries, tools and special purpose hardware like GPUs. In this work, we present FrImCla, an open-source and free tool that simplifies the construction of robust models for image classification from a dataset of images, and only using the computer CPU. Given a dataset of annotated images, FrImCla automatically constructs a classification model (both for single-label and multi-label classification problems) by trying several feature extractors (based both on transfer learning and traditional computer vision methods) and machine learning algorithms, and selecting the best combination after a thorough statistical analysis. Thus, this tool can be employed by non-expert users to create accurate models from small datasets of images without requiring any special purpose hardware. In addition, in this paper, we show that FrImCla can be employed to construct accurate models that are close, or even better, to the state-of-the-art models.

INDEX TERMS AutoML, deep learning, image classification, transfer learning.

I. INTRODUCTION

Nowadays, there exists an increment in the use of deep learning methods in a wide variety of computer vision applications, for example, in image classification within X-ray baggage security [1] or in the classification of gastrointestinal diseases [2]. However, using deep learning techniques presents several challenges for non-expert users. First of all, deep learning methods are usually data demanding, and acquiring such an amount of images in problems related to, for instance, object classification in biomedical images, might be difficult [2]–[4]. Moreover, there is not a silver bullet solution to solve all the image classification problems [5], and for each particular problem, it is necessary to conduct a thorough analysis of several algorithms to determine the improvement of one method with respect to the others.

The associate editor coordinating the review of this manuscript and approving it for publication was Juan A. Lara.

To overcome the problem of limited amount of data, one of the most fruitful approaches in the literature consists in applying *transfer learning* [6]. This technique re-uses a model trained in a source task in a new target task [7]–[10]. As explained in [6], transfer learning can be employed in different ways; one of them consists in using the output of a trained source network as ''off-the-shelf'' features that are employed to train a complete new classifier for the target task [7]. However, there are several source models (for instance, DenseNet [11], GoogleNet [12], or Resnet 50 [13] among others) that can be employed to extract the features; hence, the second problem — conducting a thorough analysis of different methods still remains. In order to tackle such a challenge, the methodology presented in [14], [15] can be employed to carry out a statistical analysis.

Even if the combination of transfer learning, machine learning algorithms and statistical analysis can be used to create models for image classification; it is necessary to combine several tools, and connecting those libraries is a time-consuming, tedious and error-prone task, that leads to ''pipeline jungles'' that are difficult to use by non-expert users [16]. In this work, we try to simplify the access and use of those tools in the line of Automated Machine Learning (AutoML) research [17], see Section II. To achieve such an aim, we have built FrImCla, a tool that allows users to experiment with deep learning and traditional computer vision techniques for single-label and multi-label image classification in a simple way, see Section III. In order to illustrate the use of FrImCla and its feasibility to construct useful image classification models for a variety of contexts and using different dataset sizes, we study several single-label and multi-label datasets in Section IV; and in addition, we compare FrImCla with other AutoML tools in Section V. The paper ends with a Conclusions section.

FrImCla is available at https://github.com/ManuGar/FrImCla where the interested reader can find a complete documentation, and several examples showing how to use this tool.

II. RELATED WORK

The goal of Automated Machine Learning (AutoML) is the democratisation of machine learning techniques [17]. This aim is achieved by simplifying the construction and usage of machine learning models, mainly supervised models, for domain experts that have a limited machine learning background.

Typically, a researcher approaches a supervised machine learning problem as follows. First of all, a set of features is extracted from a raw dataset — this step is known as *feature extraction*. Subsequently, the researcher selects a machine learning algorithm to fit the data (that is, *model selection*) and chooses a set of hyperparameters to make the model as accurate as possible (that is, *hyperparameter tuning*). Finally, the researcher validates the model to check whether it generalises properly. There are dozens of alternatives for each one of the steps of the machine learning pipeline, and AutoML techniques try to find the best combination for each particular problem [18].

The most well-known AutoML techniques have been focused on hyperparameter tuning; that is, finding the best hyperparameters for a model. These techniques include methods such as grid search [19], random search [19] or Bayesian optimisation [20]. Since there is not a single model that works best for every problem [21], hyperparameter tuning has to be conducted for several models. This issue has been addressed by tools like SMAC [22], Auto-WEKA [23], RapidMiner [24] or Auto-Sklearn [25] that discover the best combination of model and hyperparameters. Another interest of AutoML has been feature construction; for instance by creating features from relational databases via deep feature synthesis [26]. Finally, there are also end-to-end tools, such as TPOT [27] (that is built on top of the library scikit-learn and automatically designs and optimises machine learning pipelines), MLBox [28] (a library with components for preprocessing, optimising and predicting), the caret package [29] (a set of several tools for automatically developing predictive models using the R language), or the Automatic Statistician [30] (a tool that automates many aspects of data analysis, model discovery and explanation).

The aforementioned methods and tools are focused on constructing shallow models from structured data. In the case of unstructured data, such as images or text, AutoML techniques are mainly focused on automatically designing neural deep architectures [31]. This approach, known as *Neural Architecture Search* (NAS), usually involves reinforcement learning [32] or evolutionary algorithms [33] to discover new neural networks, and has outperformed manually designed architectures on tasks such as image classification or object detection [34]. However, even if NAS techniques avoid the manual design of neural architectures for image classification, the adoption of these techniques by non-expert users to construct recognition models is far from trivial; mainly, because they are computationally intensive (for instance, NasNet takes 1800 GPU days [34], and AmoebaNet takes 3150 GPU days [35]), require some prior knowledge about typical properties of architectures to simplify the search [31], and also require some coding experience to use the libraries that implement the NAS methods.

Nevertheless, non-expert users can take advantage of AutoML techniques to construct image classification models thanks to tools like Auto-Keras [36], an open-source package on top of the Keras library [37] that uses Bayesian optimisation for NAS by efficiently adapting itself to different GPU memory limits; DEvol [38], a library also on top of Keras that employs genetic architecture search; or Ludwig [39], a toolbox built on top of Tensorflow [40] that allows anyone to train deep learning models for different tasks. Even, if these tools are less computationally demanding that usual NAS techniques, the selection and construction of the best model is time-consuming, and distributed and cloud services, like Google cloud vision or Azure custom vision, have arisen [41], [42].

However, all the mentioned AutoML approaches have drawbacks when working in the context of image classification. First of all, most of the tools are designed to work with structured data; hence, the issue of extracting features from raw images remains an important challenge — it is worth noting that the efficacy of machine learning algorithms heavily relies on the employed features [43]. Tools that can train classification models from raw images, such as Auto-Keras [36] and AutoML cloud services [42], also have drawbacks. Auto-Keras is both data–demanding and requires the usage of GPUs to search the best architecture in a reasonable time. In the case of AutoML cloud services, these services are not free to use, might be difficult to configure, and raise concerns about privacy.

In this paper, we address all these challenges with the development of FrImCla, an open-source tool that can automatically construct image classification models from a small dataset of annotated images using transfer learning and

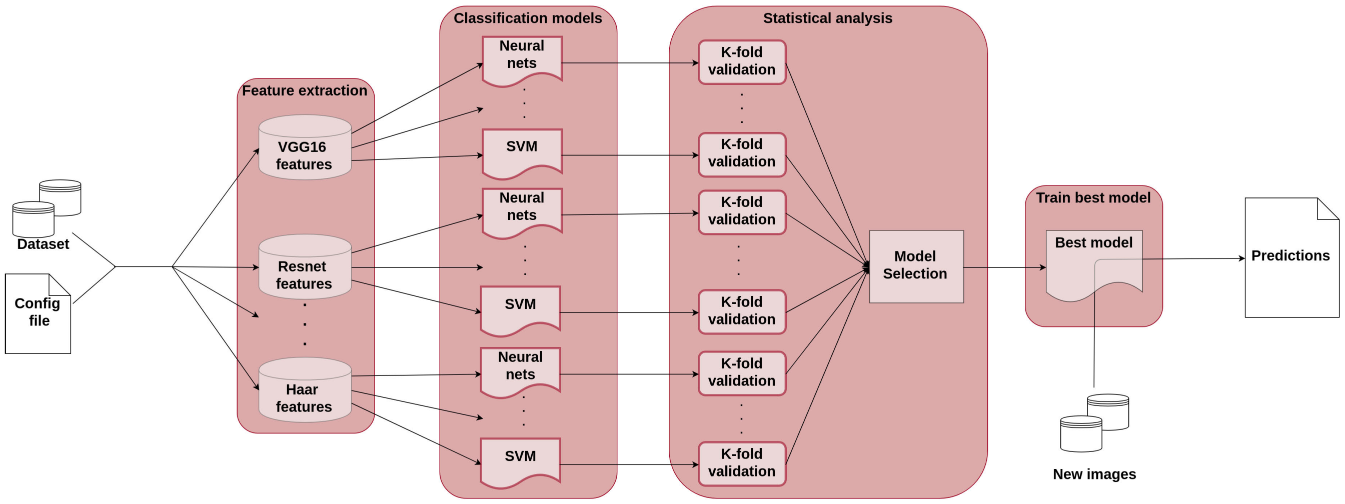

FIGURE 1. Workflow of the FrImCla framework.

traditional computer vision techniques. Namely, the contributions of this work are:

- We have developed FrImCla, an AutoML tool for image classification based on transfer learning. Moreover, we have designed FrImCla as a modular tool (see Section III). Therefore, it is possible to easily incorporate new feature extractors, classification algorithms, or tuning algorithms without modifying the current code.
- In addition, FrImCla tackles two of the main drawbacks to employ deep learning methods: the amount of data and the usage of special-purpose hardware like GPUs (see Section IV).
- Finally, we have conducted a comprehensive evaluation where we have analysed the performance and features of FrImCla with respect to other AutoML tools (see Section V).

III. FRIMCLA DESCRIPTION

FrImCla (that stands for Framework for Image Classification) is an open-source library that has been implemented in Python. This framework relies on several third-party open-source libraries that have been employed and tested by large communities; namely, scikit-learn [44], for machine learning methods and parallelisation techniques; scikit-multilearn [45], for multi-label classification algorithms; OpenCV [46], to extract traditional computer vision features; Keras [37], to extract deep learning features; and cPickle [47], to serialize objects.

The workflow of the FrImCla framework is depicted in Figure 1 and can be summarised as follows. First of all, the user selects the image dataset to be studied and some configuration parameters (mainly the feature extraction and machine learning methods to be used). Subsequently, FrImCla extracts features from the dataset using the set of fixed feature extractor methods given by the user; and a dataset of features is created for each of them. On these datasets of features, FrImCla trains the selected machine learning algorithms, and a statistical analysis is conducted to select the best combination of features and machine learning algorithm. From such a winner combination, a model is created for further usage. Apart from the first step — that is, the selection of feature extraction methods and algorithms to be tried — the rest of the process is carried out automatically by FrImCla without any user intervention.

The rest of this section is devoted to explain these steps more thoroughly.

A. FRIMCLA INPUT

FrImCla has been developed for different types of users. Non-expert users have at their disposal tools that make its execution easier, whereas more advanced users can integrate FrImCla in their own programs. For all users, documentation is available in the project web page explaining the different parts of the framework, the dependencies of the framework and so on.

Expert users can use FrImCla as a Python library, and invoke the methods to create their own model for predictions. For this kind of users this framework can be installed via pip [48]. Non-expert users can use FrImCla in a different way. In this case, the users have to create a JSON file with the parameters of the program where they have to fix seven parameters: the dataset of images, the output path, a selection of methods for feature extraction, the supervised learning algorithms, the metric to measure the performance of the generated models, the kind of problem (single-label or multi-label), and whether the output will be the best model or an ensemble of models.

Despite the simplicity of JSON files, there may still be users who find it difficult to create. For this type of users, we have developed a set of Jupyter notebooks [49]. This is a very useful tool since the user has a detailed explanation of each step that must be executed to use FrImCla.

B. iFEATURE EXTRACTION

FrImCla offers several feature extractor methods (employing both traditional and transfer-learning techniques) to describe the dataset of images. In the case of traditional computer vision feature extractors, FrImCla includes LAB and HSV histograms [50], Haralick features [51] and HOG features [52] — this functionality has been implemented using the OpenCV library. In the case of transfer learning methods, we employ the output from the last layer of several deep learning networks, trained for the ImageNet challenge [53], as feature extractors; namely, the user can select among VGG16 [54], VGG19 [54], DenseNet [11], ResNet [13], Inception [55], GoogleNet [12], Overfeat [56] and Xception [57]. Moreover, FrImCla has been designed to easily include other feature extraction methods. In particular, we distinguish two ways of adding a new feature extractor to FrImCla. The first option consists in defining a function that given an image returns a list of features — this approach should be employed when the feature extractor is based on the expert knowledge. The second option takes as input any trained Keras model and applies transfer learning using such a network — as explained for the previous networks trained for the ImageNet challenge. The features extracted from images are used to train the classification models.

C. CLASSIFICATION MODELS

Similarly to the case of feature extraction methods, FrImCla users can select among several supervised learning algorithms including SVM [58], KNN [59], Neural networks [60], Gradient Boost [61], Logistic Regression [62], Random Forest [63] and Extremely Randomised Trees [64] for single-label datasets. In the case of multi-label datasets, FrImCla users can employ all the aforementioned algorithms through the binary relevance [65], the classifier chain [66], and the label powerset [45] methods; and, additionally, the ML-KNN [67] and the MLTSVM [68] methods are provided. In both cases, the list of algorithms can be easily extended to incorporate other techniques.

The machine learning algorithms are trained using a stratified 10-fold cross validation. The cross-fold validation consists in dividing the dataset into 10 parts, one of them is used for testing and the other parts are used for training the model. The hyperparameters of each machine learning algorithm are chosen using a randomised search on the parameters distributions. The framework uses the test part to know the performance of the model. The process is repeated until each of the 10 parts are used for testing. The user can also choose among several metrics to measure the performance of the models including accuracy, F1-score, recall, precision and AUC.

The information obtained from the cross-validation is employed to select the best combination of feature extractor and classification algorithm using a statistical analysis.

D. STATISTICAL ANALYSIS

The 10-fold cross validation procedure explained previously is enough for selecting the overall best combination of feature extractor and machine learning algorithm. In addition, FrImCla provides a statistical method, that does not produce any computational overhead, to find whether there are significant differences among the constructed models. This procedure is widely employed in several fields [69]–[71].

In order to determine whether the difference between the results of the different combinations of feature extractors and classification algorithms are statistically significant, several null hypothesis tests are performed using the methodology presented in [14], [15]. In order to choose between a parametric and non-parametric test to compare the models, three conditions are checked: independency, normality and heterocedasticity; the use of a parametric test is only appropriate when the three conditions are satisfied.

The independence condition is fulfilled because different runs following a stratified 10-fold cross-validation approach are applied before separating the data with a prior random reshufling of the samples. We use the Shapiron and Wilk [72] to check normality — with the null hypothesis being that the data follow a normal distribution — and a Levene test [73] to check heteroscedasticity — with the null hypothesis being that the results are heteroscedastic.

When comparing two models, the Student's t-test [14] is employed when the parametric conditions are satisfied, and a Wilcoxon's test [14] is employed otherwise. In both cases the null hypothesis is that the two models have the same performance. If we compare more than two models, ANOVA test [15] is employed when the parametric conditions are fulfilled, and a Friedman test [15] otherwise. In both cases, the null hypothesis is that all the models have the same performance. Once the test for checking whether the models are statistically different among them has been conducted, a post-hoc procedure is employed to address the multiple hypothesis testing between the different models. A Bonferroni-Dunn post-hoc procedure [15], in the parametric case, or a Holm post-hoc procedure [74], in the non-parametric case, is used for detecting significance of the multiple comparisons [14], [15] and the p values should be corrected and adjusted. We perform our experimental analysis with a level of confidence equal to 0.05. In addition, the size effect is measured using Cohen's [75] and Eta Squared [76]. This process is performed twice. The first analysis collects the best machine learning algorithm for each feature extractor and the second analysis is focused on knowing which is the overall best combination.

E. OUTPUT OF THE FRAMEWORK

FrImCla has two output modes: best classifier and ensemble of models. In the former, once the results of the statistical analysis are obtained, FrImCla analyses which one is the best combination of feature extractor and classification algorithm. From that combination, the algorithm is retrained with the whole dataset to be further used. FrImCla generates documentation about the choice of the best model such as the accuracy or the measure selected for the training process, and it also informs the user about the time taken throughout the execution. In the latter mode, the best algorithm for each feature extractor is retrained, and it is used as a component of an ensemble based on the majority voting scheme [77] — this technique uses all the models to predict the classes of the images; and, the class with the majority of the votes is the one that results from the prediction. Normally, the ensemble mode gives better results than using just one model for prediction [77]; however, it takes more time because it uses several models to obtain the final result.

The framework generates a model that can be used in several ways. For expert users, they can use the command line or the Python functions provided for such an aim. These users can also exploit the model within their own libraries. Another option to interact with the generated model is using the Jupyter notebooks. With this option the user can read and learn what the model needs. The last option is a simple web application. This option is focused on the users that only want to predict the class of their images. FrImCla asks the users if they want to generate automatically a web application using the model resulting from the previous process.

IV. RESULTS

In order to test the suitability of FrImCla, we consider four different scenarios. First, we perform a thorough analysis of two biomedical dataset of images: the MIAS dataset of images [78] (devoted to study whether an abnormality appears in a mammographic image) and a malaria dataset [79]. Subsequently, we analyse the performance of different feature extractors for a variety of single-label datasets coming from different fields and with various sizes. Finally, we perform a similar analysis for multi-label datasets. All the experiments presented in this section were run in a computer under Linux OS on a machine with CPU Intel Core i5-8250U 3.60GHz and 7.7 GiB of RAM.

A. MIAS

The MIAS dataset consists of a set of 161 pairs of mammographic images. It has 322 photos that can be classified in several ways. The photos are in grayscale and they have an average size of 150 × 150. The images can be classified into two classes (normal or abnormal) or they can be classified into three classes (normal, benign or malign) depending on the severity of the abnormality. There are 113 abnormal images and 209 normal. The dataset is divided into 80% for training and 20% for testing to know the performance of the framework.

Using FrImCla, we have performed a thorough study by combining all the available feature extractors and machine learning models. Namely, we employed as featured extractors: VGG16, VGG19, Resnet, Inception, GoogleNet, Overfeat, DenseNet, LAB444, LAB888, HSV444 and HSV888 (where LABXYZ and HSVXYZ are respectively LAB and HSV histograms using X,Y,Z bins per channel); and as

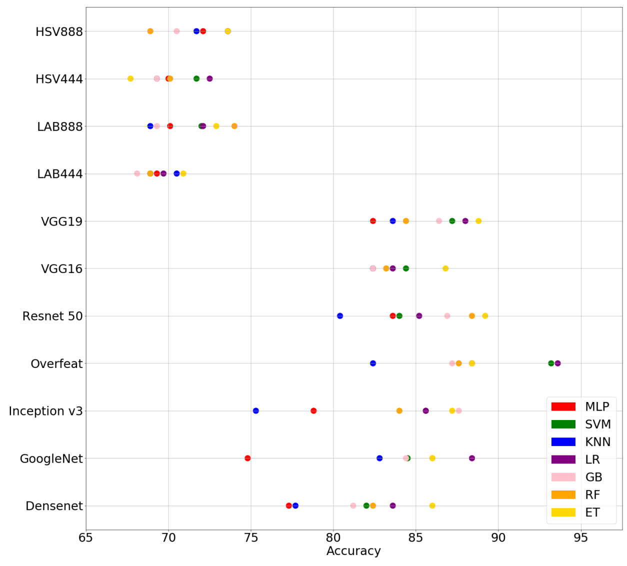

FIGURE 2. Scatter diagram of the MIAS dataset.

machine learning algorithms: SVM, KNN, Multilayer perceptron (MLP), Gradient Boost (GB), Logistic Regression (LR), Random Forest (RF), and Extremely Randomised Trees (ET).

In order to compare all the possible combinations, we analyse the statistical study performed by FrImCla. The analysis has two steps, the former serves to determine the best machine learning algorithm for each feature extractor; whereas, the latter determines which is the best overall combination. From the first analysis, see Table 1 and Figure 2, we can observe that using transfer learning features, we can easily obtain an accuracy higher than 80% but only with a few of them we can achieve an accuracy over 90%. On the contrary, models trained using traditional features are far from the 80% accuracy. This shows the importance of trying different models. In the second analysis, see Table 2 and Figure 3, we compare the best model for each feature extractor, and we check whether the three conditions (independence, normality and heteroscedasticity) are fulfilled. As the conditions are not fulfilled, a Friedman test is performed and gives us a ranking of the compared models assuming as null hypothesis that all the models have the same performance. We obtain differences ($F = 11.36; p = 6.97 \times 10^{-9}$) among the models, with a large size effect $\eta^2 = 0.593969$ (see Table 2). As a result of the analysis, we see that the best combination of feature extractor and machine learning algorithm is *Overfeat* with *Logistic Regression* (although similar results are obtained with other methods) with an accuracy of 93.6%. The result with the test set is a 92.53% accuracy and a 90.95% AUC.

Now, we compare our results with the state-of-the-art models for the MIAS dataset. The best model in the literature for the MIAS dataset was presented in [80] and achieved an accuracy of 98% using fine-tuning — a transfer learning technique. In [80], they fine-tuned the networks VGG16, Resnet50 and Inception v3 for the MIAS dataset;

TABLE 1. Mean accuracy (and standard deviation) of the different studied models for the MIAS dataset. The best result for each model in *italics*, the best result in bold face. $^{**}p < 0.01$;$^{***}p < 0.001$; $>$: there are significant differences; $\simeq$: there are not significant differences.

Network	MLP	SVM	KNN	LR	GB	RF	ET	Test (ANOVA or Friedman)	After post-hoc procedure
Densenet	77.3 (1.3)	82.0 (7.2)	77.7 (9.8)	83.6 (5.4)	81.2 (9.6)	82.4 (1.0)	*86.0 (9.1)*	0.43	ET $\simeq$ MLP, SVM, KNN, LR, GB, RF
GoogleNet	74.8 (2.8)	84.5 (5.3)	82.8 (4.5)	*88.4 (2.8)*	84.4 (5.1)	86.0 (2.7)	86.0 (4.5)	4.37**	LR $\simeq$ SVM, KNN, LR, GB, RF; LR $>$ MLP
Inception v3	78.8 (2.6)	85.6 (2.3)	75.3 (6.6)	85.6 (3.4)	*87.6 (2.9)*	84.0 (5.6)	87.2 (4.6)	4.59**	GB $\simeq$ SVM, LR, ET, RF; GB $>$ KNN, MLP
Overfeat	88.4 (5.7)	93.2 (2.6)	82.4 (4.1)	***93.6 (3.8)***	87.2 (5.7)	87.6 (3.2)	88.4 (4.6)	2.95*	LR $\simeq$ MLP, SVM, ET, GB, RF; LR $>$ KNN
Resnet 50	83.6 (1.9)	84.0 (2.7)	80.4 (2.2)	85.2 (3.3)	86.9 (5.7)	88.4 (1.8)	*89.2 (6.5)*	2.42	ET $\simeq$ MLP, SVM, KNN, LR, GB, RF
VGG16	82.4 (4.5)	84.4 (2.8)	82.4 (4.6)	83.6 (4.8)	82.4 (6.8)	83.2 (1.0)	*86.8 (4.1)*	0.51	ET $\simeq$ MLP, SVM, KNN, LR, GB, RF
VGG19	82.4 (5.1)	87.2 (2.8)	83.6 (3.6)	88.0 (2.4)	86.4 (5.8)	84.4 (2.8)	*88.8 (3.8)*	1.44	ET $\simeq$ MLP, SVM, KNN, LR, GB, RF
LAB444	69.3 (4.0)	68.9 (5.9)	70.5 (7.0)	69.7 (5.1)	68.1 (8.8)	68.9 (7.4)	*70.9 (8.5)*	0.07	ET $\simeq$ MLP, SVM, KNN, LR, GB, RF
LAB888	70.1 (5.6)	72.0 (7.5)	68.9 (9.1)	72.1 (6.4)	69.3 (8.2)	*74.0 (9.7)*	72.9 (5.0)	0.26	RF $\simeq$ MLP, SVM, KNN, LR, GB, ET
HSV444	70.0 (7.6)	71.7 (4.8)	69.3 (6.4)	*72.5 (7.3)*	69.3 (6.7)	70.1 (6.8)	67.7 (4.1)	0.18	SVM $\simeq$ MLP, LR, KNN, RF, GB, ET
HSV888	72.1 (6.0)	*73.6 (7.4)*	71.7 (9.6)	73.6 (8.0)	70.5 (8.1)	68.9 (11.1)	73.6 (8.3)	0.24	LR $\simeq$ MLP, SVM, KNN, RF, GB, ET

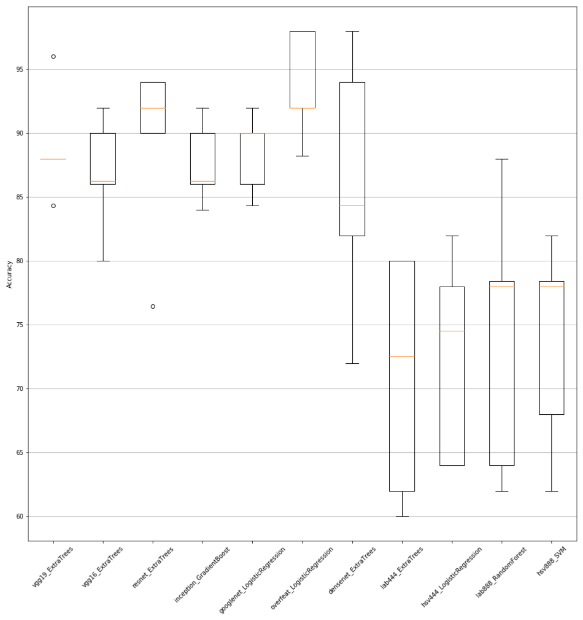

FIGURE 3. Box diagram of the MIAS dataset.

but, instead of applying transfer learning from natural images, they fine-tuned the networks previously trained in other mammographical datasets. This shows the importance of transferring the knowledge from closer domains. However, such an approach requires networks trained in similar datasets with more than 6100 images and the use of GPUs, two restrictions that are not always fulfilled.

Up to the best of our knowledge, the best model for the MIAS dataset built only from images of this dataset was presented in [81], and also used transfer-learning from natural

TABLE 2. Adjusted *p*-values with Holm, and Cohen's d for best models in MIAS dataset. Control technique: Overfeat-LR.

Technique	Z value	p value	adjusted p value	Cohen's d
Densenet-ET	1.23	0.21	0.83	0.96
Resnet-ET	1.23	0.21	0.83	0.72
VGG19-ET	1.33	0.18	0.83	1.11
GoogleNet-LR	1.38	0.16	0.83	1.37
Inception-GB	1.38	0.16	0.83	1.58
VGG16-ET	1.62	0.10	0.63	1.53
LAB888-RF	3.48	0.0005	0.0035	2.36
HSV888-SVM	3.52	0.0004	0.0033	3.01
HSV444-LR	3.71	0.0002	0.0018	3.23
LAB444-ET	4.14	3.36×10^{-5}	0.0003	3.06

images, as in our case. In [81], the networks Inception, Xception and MobileNet were fine-tuned achieving an accuracy of 90% in the test set using the same percentage split employed in our work. Such an accuracy is lower than ours, and was obtained using GPUs.

B. MALARIA PARASITE CLASSIFICATION

In our second case study, we consider the Malaria parasite classification dataset that consists of 27,558 cells images. The images are in colour and have an average size of 150 $\times$ 150. These images can be classified into two classes: parasitized and uninfected. There are 13,779 for each class. For this example, we use only 2,000 images (1,000 images per class) for training and the rest of the dataset for testing.

We apply the two-stage analysis employed previously. When comparing the best model for each feature extractor, as the parametric conditions are not fulfilled, a Friedman

TABLE 3. Mean accuracy (and standard deviation) of the different studied models for the Malaria dataset. The best result for each model in *italics*, the best result in **bold** face. $^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$; $>$: there are significant differences; $\simeq$: there are not significant differences.

Network	KNN	LR	MLP	RF	SVM	GB	Test (ANOVA or Friedman)	After post-hoc procedure
DenseNet	64.8 (1.7)	75.7 (2.9)	64.1 (1.9)	70.5 (1.1)	*75.7 (2.6)*	72.2 (3.3)	18.3***	SVM $\simeq$ LR, GB, RF; SVM$>$ MLP, KNN
GoogleNet	87.2 (2.1)	92.3 (2.3)	92.6 (2.4)	89.5 (1.8)	*92.8 (1.8)*	90.5 (1.4)	4.92***	SVM $\simeq$ MLP, LR; SVM $>$ RF, GB, KNN
Inception v3	71.0 (1.6)	*85.5 (1.5)*	84.6 (2.2)	85.3 (2.0)	84.8 (2.0)	85.6 (2.4)	3.25*	LR $\simeq$ RF, GB, MLP, SVM; LR$>$ KNN
Overfeat	75.0 (4.2)	92.7 (1.2)	93.5 (1.1)	85.0 (1.1)	*93.5 (1.4)*	87.1 (2.1)	32.46***	SVM $\simeq$ MLP, LR, GB; SVM $>$ RF, KNN
Resnet 50	62.6 (1.8)	73.2 (3.4)	73.3 (2.2)	77.8 (3.2)	70.2 (4.7)	*79.7 (3.1)*	14.29***	GB $\simeq$ LR, MLP, RF; GB$>$ SVM, KNN
VGG16	70.0 (1.8)	84.6 (2.9)	82.7 (3.1)	84.9 (1.3)	83.3 (3.1)	*86.2 (0.9)*	5.54**	GB $\simeq$ MLP, RF, LR, SVM; GB $>$ KNN
VGG19	75.2 (2.5)	82.9 (1.5)	*83.4 (1.8)*	81.1 (2.7)	81.7 (1.6)	80.0 (2.4)	7.59***	MLP $\simeq$ LR, RF, GB, SVM; MLP $>$ KNN
Xception	73.5 (3.3)	*87.5 (2.0)*	86.8 (1.9)	85.7 (2.0)	86.8 (2.0)	86.7 (1.9)	13.24***	LR $\simeq$ GB, MLP, SVM, RF; LR $>$ KNN
LAB888	72.4 (2.7)	89.6 (2.2)	86.8 (7.0)	*94.7 (1.2)*	89.3 (3.6)	***94.8 (1.6)***	33.23***	RF $\simeq$ GB, SVM, MLP; GB $>$ LR, KNN
LAB444	77.9 (2.9)	69.7 (2.1)	78.1 (7.2)	90.8 (1.7)	69.7 (1.2)	*91.8 (0.6)*	43.3***	GB $\simeq$ RF, KNN, MLP; GB $>$ SVM, LR
HSV888	69.2 (1.7)	86.7 (3.5)	84.1 (3.8)	*94.6 (1.0)*	88.2 (1.7)	94.4 (1.2)	29.98***	RF $\simeq$ GB, SVM, LR; RF $>$ MLP,KNN
HSV444	70.4 (1.7)	89.8 (2.8)	91.8 (1.9)	94.4 (1.5)	89.8 (3.4)	93.9 (1.7)	48.23***	RF $\simeq$ GB, MLP; RF $>$ SVM, LR, KNN

TABLE 4. Adjusted *p*-values with Holm, and Cohen's d for best models in Malaria dataset. Control technique: LAB888-GB.

Technique	Z value	p value	adjusted p value	Cohen's d
HSV888-GB	0.10	0.92	1.00	0.12
HSV444-RF	0.33	0.92	1.00	0.54
Overfeat-SVM	0.50	0.62	1.00	0.95
GoogleNet-MLP	0.66	0.51	1.00	0.88
LAB444-GB	1.19	0.23	1.00	2.26
Xception-GB	1.79	0.07	0.48	3.20
Vgg16-MLP	2.19	0.03	0.20	4.99
Inception-LR	2.32	0.02	0.16	5.84
Vgg19-LR	2.72	6.46×10^{-3}	0.06	6.34
Resnet 50-GB	3.09	2.01×10^{-3}	0.02	5.50
DenseNet-LR	3.82	1.34×10^{-4}	1.60×10^{-3}	7.80

TABLE 5. Datasets used for single-label classification.

Dataset	♯ images	♯ classes	Type of images
Blindness [82]	3662	5	Fundus images
Chest X Ray [83]	2355	2	Chest X Ray images
Fungi [84]	1204	4	Fungi decolorisation images
German Traffic Signs [85]	39209	43	Traffic signs
HAM10000 [86]	100015	7	Dermatoscopic images
Open Sprayer [87]	6697	2	Dock images
RESISC45 [88]	31500	45	Geospatial images
Tobacco [89]	3492	10	Documents
UCMerced [90]	2100	21	Geospatial images

test is performed and give us a ranking of the compared models assuming as null hypothesis that all the models have the same performance, see Table 3. We obtain differences ($F = 52.53$; $p = 1.11 \times 10^{-16}$) among the models, with a large size effect $\eta^2 = 0.942912$ (see Table 4). As a result

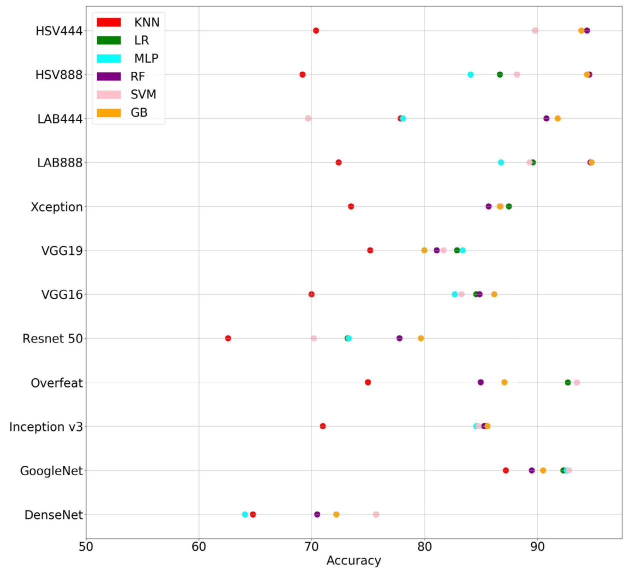

FIGURE 4. Scatter diagram of the Malaria dataset.

of the analysis, the best combination of feature extractor and machine learning algorithm is *LAB888* with *Gradient Boost* (although similar results are obtained with other methods) with an accuracy of 94.8%. In this example, we can see how deep learning models do not always give us the best results and, therefore, it is important to study other alternatives. As we can see in the Figure 4, classical computer algorithms obtain better results than deep learning algorithms. This result highlights the importance of testing all models, because there is not one that produces the best results for all problems.

TABLE 6. Results obtained by FrImCla in the single-label datasets.

	VGG16	VGG19	Resnet	Inception	GoogleNet	Overfeat	Xception	HOG
Blindness	56.2(3.2)	56.0(4.2)	58.2(4.1)	56.0(5.0)	**60.9(3.1)**	60.4(4.3)	55.5(4.4)	55.7(3.8)
Chest X ray	90.4(2.5)	91.2(2.3)	92.2(2.4)	88.7(1.7)	94.2(1.5)	**95.3(1.6)**	90.7(2.4)	94.6(1.0)
Fungi	79.8(3.7)	81.3(3.2)	86.7(2.0)	83.1(1.9)	93.6(2.2)	91.8(2.7)	79.5(4.7)	**95.4(1.7)**
German Traffic Signs	44.3(2.6)	43.7(5.0)	51.9(2.9)	45.2(3.8)	78.8(3.1)	**85.5(1.8)**	52.3(4.7)	48.5(3.1)
HAM10000	41.1(2.9)	43.6(3.5)	48.1(3.7)	40.2(3.3)	48.5(2.8)	**59.0(2.2)**	42.0(3.6)	49.9(3.2)
Open Sprayer	87.9(1.8)	**88.3(1.9)**	85.9(4.1)	88.1(3.3)	83.6(2.0)	83.1(2.2)	88.2(2.9)	76.9(2.9)
RESISC45	46.8(3.5)	46.5(2.7)	48.2(1.3)	42.7(2.6)	67.2(1.5)	**67.6(4.2)**	48.1(2.2)	36.5(3.5)
Tobacco	59.2(3.8)	58.4(3.8)	59.4(4.7)	59.9(4.1)	68.1(5.1)	**73.5(3.2)**	57.2(3.5)	64.9(3.6)
UCMerced	77.0(2.8)	76.6(2.2)	76.9(2.5)	74.0(2.1)	89.8(1.3)	**92.4(2.1)**	77.1(3.7)	71.4(5.4)

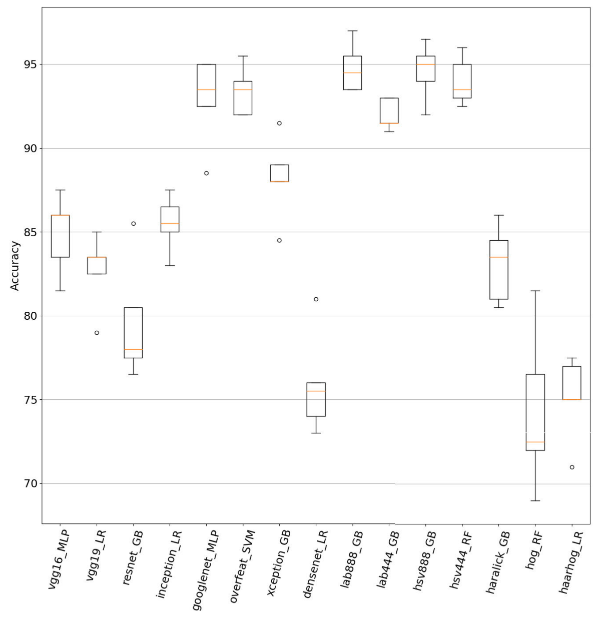

FIGURE 5. Box diagram of the Malaria dataset.

TABLE 7. Results obtained by FrImCla in the multi-label datasets.

	Apparel	Scenes	UCMerced
VGG16	56.8(2.3)	68.9(3.0)	63.3(2.7)
VGG19	57.3(2.0)	65.8(4.1)	56.4(2.0)
Resnet	58.2(1.1)	72.8(3.0)	64.6(3.5)
Inception	54.2(2.1)	71.1(2.8)	59.3(3.2)
GoogleNet	80.3(2.2)	69.7(2.8)	**68.4(3.3)**
Overfeat	**86.4(1.6)**	**75.5(2.6)**	68.1(1.4)
Xception	56.9(2.0	64.9(3.9)	54.9(2.2)
Haralick	54.1(2.1)	64.2(3.3)	59.0(3.1)

After conducting the statistical analysis using 2000 images, we employed the rest of the dataset for testing, obtaining a 95.4% accuracy using *LAB888* as feature extractor and *Gradient Boost* as classifier model.

The malaria dataset was released recently, and the best model for this dataset was presented in [79] also using transfer learning. In that work, they used the networks AlexNet, VGG16, Xception, Resnet and DenseNet as feature extractors to subsequently train a fully-connected network. They use the entire dataset with a five-fold cross-validation obtaining a 95.7% accuracy employing GPUs. On the contrary, we only needed 2000 images of the dataset to obtain a similar accuracy of 95.4% and without using special purpose hardware.

C. SINGLE-LABEL DATASETS

In the two previous sections, we have performed a thorough analysis of the performance of FrImCla on two biomedical datasets. Now, we consider 9 datasets coming from different fields and that have a variety of sizes and number of classes — a summary of those datasets is provided in Table 5. Using the two-stage procedure previously presented, we have analysed the performance of FrImCla on those datasets using VGG16, VGG19, Resnet, Inception, GoogleNet, Overfeat, Xception, and HOG as feature extractors; and SVM, KNN, Multilayer perceptron, Gradient Boost, Logistic Regression, Random Forest, and Extremely Randomised Trees as machine learning algorithms.

In this study, we provide a summary of the results obtained using FrImCla; namely, in Table 6, we include the best model constructed with each feature extractor. From that table, we can notice that, even if there is not a single feature extractor that obtains the best results for all datasets, the Overfeat feature extractor stands out since it achieves the best accuracy in 6 out of the 9 datasets. Something similar happens with the classification algorithm, since in 6 out of 9 datasets the SVM classifier produces the best results. Therefore, we can claim that, in general, the Overfeat feature extractor captures better the underlying representation of images from different fields.

D. MULTI-LABEL DATASETS

We finish this section by analysing the results obtained by FrImCla in three multi-label datasets. Namely, we employ the Scenes dataset [91] (2000 natural scene images, with 5 distinct class labels and with 1.24 class labels per image on average), the UCMerced archive [92] (2100 geospatial images, with 17 distinct class labels and with 3.33 class labels per image on average), and the Apparel dataset [93]

TABLE 8. Features of AutoKeras, Devol, FrImCla and Ludwig for constructing image classification models.

	Technique	Selection of best model	Hardware	Installation	Usage	Configuration
FrImCla	Transfer learning	Statistical Analysis	CPU/GPU	pip	Python library, CLI, Jupyter notebooks	Feature extractors and classification algorithms
AutoKeras	NAS	Holdout	GPU	pip	Python library	Time
Devol	NAS	Holdout	GPU	Git	Python library	Population size and generations
Ludwig	Transfer learning	None	CPU/GPU	pip	CLI	Encoder
WND-CHARM	Traditional features	Cross validation	CPU	Git	CLI	None

(11385 images of cloths, with 11 distinct class labels and with 2 class labels per image).

As in the previous sections, we have applied a two-stage analysis to find the best model for each dataset. In this scenario, we have employed 8 feature extractors (VGG16, VGG19, Resnet, Inception, GoogleNet, Overfeat, Xception, and Haralick) and 23 classifiers (the binary relevance, the classifier chain, and the label powerset methods applied to SVM, KNN, Multilayer perceptron, Gradient Boost, Logistic Regression, Random Forest, and Extremely Randomised Trees; and, the ML-KNN and MLTSVM classifiers).

The results obtained in the second stage of the procedure are summarised in Table 7. Similarly to the results presented in the previous section, the Overfeat feature extractor produces the best results for both the Apparel and Scenes datasets, and there are not significant differences between the results obtained with this feature extractor and those obtained with the best extractor for the UCMerced archive. On the classifier side, the best algorithm depends on the concrete problem, binary relevance combined with Logistic regression for the Apparel dataset, label powerset using a Multilayer perceptron for the Scenes dataset, and label powerset using Logistic regression for the UCMerced archive.

V. DISCUSSION

FrImCla is aligned with the research related to AutoML. The main difference of FrImCla with respect to the majority of AutoML tools, such as Auto-sklearn [25] or Auto-WEKA [23], is that, instead of working with structured data, FrImCla works with raw datasets of images. Such a feature is only available, at least up to the best of our knowledge, in other four open-source tools: AutoKeras [36], Devol [38], Ludwig [39], and WND-CHARM [94]. In this section, we compare the characteristics, and performance, of these five tools. We start by analysing the methods employed by each of these systems.

Despite the fact that the five tools can be employed to construct image classification models, they take different paths to achieve such an aim. The only tool that employs traditional image features and machine learning algorithms is WND-CHARM; the rest of the tools employ deep learning methods. The underlying technique of both AutoKeras and Devol is neural architecture search — using Bayesian optimisation to conduct the search in the case of AutoKeras, and a genetic algorithm in the case of Devol. In the case of

TABLE 9. Datasets used for comparing AutoML tools.

Dataset	♯ classes	♯ training images	♯ test images
Binucleate	2	30	11
C. Elegans	4	189	63
Cho	5	245	82
Hela	10	508	181
Liver aging	4	637	213
Liver gender (AL)	2	391	131
Liver gender (CR)	2	192	64
Lymphoma	3	280	94
Pollen	7	472	158
RNAi	10	150	50
Terminal Bulb Aging	7	727	243

FrImCla and Ludwig, they use transfer learning — the main difference being that FrImCla carries out a thorough analysis of different architectures to find the best possible solution for each particular problem, and, on the contrary, Ludwig only uses either the ResNet or VGG architecture as encoder to construct a model. In order to decide the best explored model, AutoKeras and Devol use the holdout method (that is, part of the dataset is only used for validating the performance of the models), WND-CHARM uses k-fold cross validation, and FrImCla employs the thorough statistical analysis discussed in Section III.

The underlying techniques of these systems have an impact on the computational resources that are required to run them: AutoKeras and Devol can only be executed using a computer with a GPU, due to the computational intensive nature of the neural architecture search algorithms; while FrImCla, Ludwig and WND-CHARM can be run just using a CPU; in addition, both FrImCla and Ludwig can take advantage of a GPU if it is available in the user's computer. Since Ludwig does not explore different alternatives and WND-CHARM explores less alternatives, they are faster than FrImCla. It is worth mentioning that in spite of using different techniques for constructing image classification models, the four tools that employ deep learning methods (that is, AutoKeras, Devol, FrImCla, and Ludwig) rely on the Keras library to implement them, a point that facilitates the installation of these libraries.

An instrumental aspect of AutoML tools is their usability, and this starts with their installation. AutoKeras, Devol, FrImCla, and Ludwig can be installed on any operating system which has Python available on it — that is, they can be

TABLE 10. **Comparison of the performance of AutoKeras, Devol, FrImCla, Ludwig, and WND-CHARM for constructing image classification models in 11 image classification problems. The best result is highlighted in bold face.**

	Binucleate	C. elegans	Cho	Hela	Liver Aging	Liver Gender (AL)	Liver Gender (CR)	Lymphoma	Pollen	RNAi	Terminal
FrImCla	**1.00**	**0.85**	**0.98**	0.82	0.87	**0.98**	**1.00**	0.86	**0.96**	**0.69**	**0.59**
AutoKeras	0.55	0.66	0.96	0.47	0.91	**0.98**	**1.00**	**0.89**	0.81	0.24	0.47
Devol	0.54	0.69	0.75	0.68	0.43	0.82	**1.00**	0.55	0.89	0.28	0.36
Ludwig	0.54	0.48	0.64	0.51	0.33	0.65	0.93	0.57	0.58	0	0.53
WndCharm	**1.00**	0.7	0.95	**0.88**	**0.93**	0.98	0.97	0.79	**0.96**	0.66	0.45

easily installed in Windows, Linux and Mac. In the case of AutoKeras, FrImCla and Ludwig by using the pip package installer; and by cloning a Git repository in the case of Devol. The installation of WND-CHARM might be more challenging, especially for Windows users, since it requires a C++ compiler and several C++ libraries.

With respect to their usage, AutoKeras and Devol can only be invoked as Python libraries; so, they require some coding experience. On the contrary, Ludwig only requires a JSON file that should be provided as a parameter of a command invoked from the console; and, similarly, WND-CHARM is invoked from a command line interface, and it only requires as parameters the paths to the folders containing the images. In the case of FrImCla, we support different execution modes: FrImCla can be used as a library, as AutoKeras and Devol; by means of a JSON file, as Ludwig; or by means of Jupyter notebooks, that provide a detailed explanation of each step that must be executed. Finally, the last usability aspect that we consider is the interaction required from the users. In WND-CHARM, users do not provide any configuration parameter; in AutoKeras, users provide the maximum time that the search can be run; in Devol, users fix the population size and the number of generations that the genetic algorithm is run; in Ludwig, they select the encoder algorithm, either ResNet or VGG; and, in FrImCla, users choose the feature extraction and classification algorithm, but it is also possible to explore all the possible alternatives automatically.

We finish this section by comparing the performance of AutoKeras, Devol, FrimCla, Ludwig, and WND-CHARM by using 11 datasets from different types of image classification problems (these datasets were studied in [95]) — the sizes of the training sets, test sets, and the number of classes in each dataset are listed in Table 9. The accuracy achieved by each tool is presented in Table 10. The execution environment used for the tests has been Google Colab with the GPU option activated for AutoKeras, Devol, and FrimCla; and for Ludwig and WND-CHARM, the experiments were run in a computer under Linux OS on a machine with CPU Intel Core i5-8250U 3.60GHz and 7.7 GiB of RAM.

As we can see, FrImCla obtains better results in most datasets, and in several cases the improvement is very noticeable. There are only three datasets where other tools achieve a better accuracy, and even in those cases, the best accuracy and FrImCla's accuracy are close. It is worth noting that in most datasets, AutoKeras, Devol, and Ludwig tend to overfit, whereas the models constructed by FrimCla and WND-CHARM generalise properly.

VI. CONCLUSION

In this paper, we have presented FrImCla, an open-source framework that allows users to easily create image classification models. This is achieved by automating all the steps required to train an image classification model. Namely, FrImCla is able to select the best combination of feature extractor and classification algorithm by conducting a comprehensive statistical study.

FrImCla can be framed in the context of AutoML tools, but it has several advantages with respect to the existing tools. First of all, FrImCla automatises the whole pipeline to construct classification models from raw images. In addition, it reduces the amount of data and computational resources that are required to train those classification models thanks to the use of transfer learning. Furthermore, the accuracy achieved by the models constructed with FrImCla is superior to the accuracy obtained using other AutoML tools. In fact, FrImCla models can achieve close results to state-of-the-art models specific for concrete problems by using a general method that can be applied to any dataset.

To summarise this work, FrImCla aims to help users without a deep understanding about machine learning or computer vision, but that are interested in building image classification models for their problems. Thanks to FrImCla, users from several contexts can use state-of-the-art techniques and build accurate models from small datasets, and without requiring any special hardware. In the future, we plan to extend the FrImCla framework to deal not only with the problem of classification, but also with other important problems such as localisation, detection and semantic segmentation.

REFERENCES

[1] S. Akcay, M. E. Kundegorski, M. Devereux, and T. P. Breckon, "Transfer learning using convolutional neural networks for object classification within X-ray baggage security imagery," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2016, pp. 1057–1061.

[2] A. Asperti and C. Mastronardo, "The effectiveness of data augmentation for detection of gastrointestinal diseases from endoscopical images," in *Proc. 11th Int. Joint Conf. Biomed. Eng. Syst. Technol.*, vol. 2, 2018, pp. 199–205.

[3] E. Valle, M. Fornaciali, A. Menegola, J. Tavares, F. V. Bittencourt, L. T. Li, and S. Avila, "Data, depth, and design: Learning reliable models for skin lesion analysis," 2017, *arXiv:1711.00441*. [Online]. Available: http://arxiv.org/abs/1711.00441

[4] A. Galdran, A. Alvarez-Gila, M. Ines Meyer, C. L. Saratxaga, T. Araújo, E. Garrote, G. Aresta, P. Costa, A. M. Mendonça, and A. Campilho, "Data-driven color augmentation techniques for deep skin image analysis," 2017, *arXiv:1703.03702*. [Online]. Available: http://arxiv.org/abs/1703.03702

[5] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. Cambridge, MA, USA: MIT Press, 2012.

[6] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Columbus, OH, USA, Jun. 2014, pp. 512–519.

[7] S. Jialin Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[8] S. Christodoulidis, M. Anthimopoulos, L. Ebner, A. Christe, and S. Mougiakakou, "Multisource transfer learning with convolutional neural networks for lung pattern analysis," *IEEE J. Biomed. Health Inform.*, vol. 21, no. 1, pp. 76–84, Jan. 2017.

[9] M. Ghafoorian, "Transfer learning for domain adaptation in MRI: Application in brain lesion segmentation," in *Medical Image Computing and Computer Assisted Intervention—MICCAI*. Cham, Switzerland: Springer, 2017, pp. 516–524.

[10] A. Menegola, M. Fornaciali, R. Pires, F. V. Bittencourt, S. Avila, and E. Valle, "Knowledge transfer for melanoma screening with deep learning," in *Proc. IEEE 14th Int. Symp. Biomed. Imag. (ISBI)*, Apr. 2017, pp. 297–300.

[11] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[14] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Inf. Sci.*, vol. 180, no. 10, pp. 2044–2064, May 2010.

[15] D. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*. Boca Raton, FL, USA: CRC Press, 2011.

[16] D. Sculley, "Hidden technical debt in machine learning systems," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 2503–2511.

[17] I. Guyon, K. Bennett, G. Cawley, H. J. Escalante, S. Escalera, T. K. Ho, N. Macià, B. Ray, M. Saeed, A. Statnikov, and E. Viegas, "AutoML challenge 2015: Design and first results," in *Proc. AutoML @ICML*, 2015, pp. 1–10. [Online]. Available: https://drive.google.com/file/d/0BzRGLkqgrI-qWkpzcGw4bFpBMUk/view

[18] F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., *Automated Machine Learning: Methods, Systems, Challenges*. Gewerbestrasse, Switzerland: Springer, 2018. [Online]. Available: http://automl.org/book

[19] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.

[20] S. Snoek, H. Larochelle, and R. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 2951–2959.

[21] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[22] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential Model-Based Optimization for General Algorithm Configuration," in *Proc. Int. Conf. Learn. Intell. Optim.*, in Lecture Notes in Computer Science, vol. 6683. Berlin, Germany: Springer, 2011, pp. 507–523.

[23] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 847–855.

[24] M. Reif, F. Shafait, M. Goldstein, T. Breuel, and A. Dengel, "Automatic classifier selection for non-experts," *Pattern Anal. Appl.*, vol. 17, no. 1, pp. 83–96, Feb. 2012.

[25] M. Feurer, "Efficient and Robust Automated Machine Learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 2962–2970.

[26] J. M. Kanter and K. Veeramachaneni, "Deep feature synthesis: Towards automating data science endeavors," in *Proc. IEEE Int. Conf. Data Sci. Adv. Anal. (DSAA)*, Oct. 2015, pp. 1–10.

[27] R. S. Olson, R. J. Urbanowicz, P. C. Andrews, N. A. Lavender, L. C. Kidd, and J. H. Moore, "Automating biomedical data science through tree-based pipeline optimization," in *Proc. 19th Eur. Conf. Appl. Evol. Comput.* Cham, Switzerland: Springer, Mar./Apr. 2016, pp. 123–137, doi: 10.1007/978-3-319-31204-0_9.

[28] A. de Romblay. (2017). *MLBox, Machine Learning Box*. [Online]. Available: https://mlbox.readthedocs.io

[29] M. Kuhn, "Building predictive models in R using the caret package," *J. Statist. Softw.*, vol. 28, no. 5, pp. 1–26, 2008.

[30] C. Steinruecken, "The automatic statistician," in *Automated Machine Learning: Methods, Systems, Challenges* (The Springer Series on Challenges in Machine Learning), F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Cham, Switzerland: Springer, 2019.

[31] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 55, pp. 1–21, 2019.

[32] Y. Jaafra, J. Luc Laurent, A. Deruyver, and M. Saber Naceur, "Reinforcement learning for neural architecture search: A review," *Image Vis. Comput.*, vol. 89, pp. 57–66, Sep. 2019.

[33] J. Liang, E. Meyerson, and R. Miikkulainen, "Evolutionary architecture search for deep multitask networks," in *Proc. Genetic Evol. Comput. Conf. (GECCO)*, 2018, pp. 466–473.

[34] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1–14.

[35] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, Jul. 2019, pp. 4780–4789.

[36] H. Jin, Q. Song, and X. Hu, "Auto-keras: An efficient neural architecture search system," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2019, pp. 1946–1956.

[37] F. Chollet. (2017). *Keras*. [Online]. Available: https://keras.io/

[38] J. Davison. (2018). *DEvol—Deep Neural Network Evolution*. [Online]. Available: https://github.com/joeddav/devol

[39] P. Molino, Y. Dudin, and S. Sumanth Miryala, "Ludwig: A type-based declarative deep learning toolbox," 2019, *arXiv:1909.07930*. [Online]. Available: http://arxiv.org/abs/1909.07930

[40] Google. (2018). *TensorFlow: An Open-Source Software Library for Machine Intelligence*. [Online]. Available: https://www.tensorflow.org

[41] T. Kraska, A. Talwalkar, and J. Duchi, "MLbase: A distributed machine-learning system," in *Proc. Conf. Innov. Data Syst. Res.*, 2013, pp. 2.

[42] Google. (2018). *Google Cloud AutoML*. [Online]. Available: https://cloud.google.com/

[43] P. Domingos, "A few useful things to know about machine learning," *Commun. ACM*, vol. 55, no. 10, pp. 78–87, Oct. 2012.

[44] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, and D. Cournapeau, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[45] P. Szymański and T. Kajdanowicz, "A scikit-based Python environment for performing multi-label classification," 2017, *arXiv:1702.01460*. [Online]. Available: http://arxiv.org/abs/1702.01460

[46] G. Bradski, "The OpenCV library," *Dr. Dobb's J. Softw. Tools*, vol. 25, pp. 122–125, Nov. 2000.

[47] *Cpickle Library*, Python Softw. Found., Wilmington, DE, USA, 2018.

[48] Python Packaging Authority. (2011). *Pip*. [Online]. Available: https://pypi.org/project/pip/

[49] *Damion Avila*, Project Jupyter, Fernando Pérez, Brian Granger, 2018.

[50] R. S. Hunter, "Photoelectric color-difference meter," *J. Opt. Soc. Amer.*, vol. 38, no. 7, p. 661, 1948.

[51] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, no. 6, pp. 610–621, Nov. 1973.

[52] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Jun. 2005, pp. 886–893.

[53] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, and A. C. Berg, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[54] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[55] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[56] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Apr. 2014.

[57] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1800–1807.
[58] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[59] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theor.*, vol. 13, no. 1, pp. 21–27, Jan. 2006.
[60] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Oxford Univ. Press, 1995.
[61] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001.
[62] P. McCullagh and J. A. Nelder, *Generalized Linear Models*. London, U.K.: Chapman & Hall, 1989.
[63] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[64] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, 2006.
[65] M.-L. Zhang, Y.-K. Li, X.-Y. Liu, and X. Geng, "Binary relevance for multi-label learning: An overview," *Frontiers Comput. Sci.*, vol. 12, no. 2, pp. 191–202, Apr. 2018.
[66] J. Read, "Classifier chains for multi-label classification," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Berlin, Germany: Springer, 2009, pp. 254–269.
[67] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognit.*, vol. 40, no. 7, pp. 2038–2048, Jul. 2007.
[68] W.-J. Chen, Y.-H. Shao, C.-N. Li, and N.-Y. Deng, "MLTSVM: A novel twin support vector machine to multi-label learning," *Pattern Recognit.*, vol. 52, pp. 61–74, Apr. 2016.
[69] K. Bandara, C. Bergmeir, and S. Smyl, "Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach," *Expert Syst. Appl.*, vol. 140, Feb. 2020, Art. no. 112896.
[70] C. Fernandez-Lozano, A. Carballal, P. Machado, A. Santos, and J. Romero, "Visual complexity modelling based on image features fusion of multiple kernels," *PeerJ*, vol. 7, p. e7075, Jul. 2019.
[71] S. H. Ghafarian and H. S. Yazdi, "Identifying crisis-related informative tweets using learning on distributions," *Inf. Process. Manage.*, vol. 57, no. 2, Mar. 2020, Art. no. 102145.
[72] S. S. Shapiron and M. B. Wilk, "An analysis for variance test for normality (complete samples)," *Inf. Sci.*, vol. 180, nos. 3–4, pp. 2044–2064, 1965.
[73] H. Levene, "Robust tests for equality of variances," in *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling*. Palo Alto, CA, USA: Stanford Univ. Press, 1960, pp. 278–292.
[74] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandin. J. Statist.*, vol. 6, no. 2, pp. 65–70, 1979.
[75] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*. New York, NY, USA: Academic, 1969.
[76] J. Cohen, "Eta-squared and partial Eta-squared in fixed factor anova designs," *Educ. Psychol. Meas.*, vol. 33, no. 1, pp. 107–112, Apr. 1973.
[77] C. Zhang and Y. Ma, *Ensemble Machine Learning: Methods and Applications*. New York, NY, USA: Springer, 2012.
[78] J. Suckling, J. Parker, D. Dance, S. Astley, I. Hutt, C. Boggis, I. Ricketts, E. Stamatakis, N. Cerneaz, and S. E. A. Kok. (2018). *Mammographic Image Analysis Society (MIAS) Database v1.21*. [Online]. Available: https://www.repository.cam.ac.uk/handle/1810/250394
[79] S. Rajaraman, S. K. Antani, M. Poostchi, K. Silamut, M. A. Hossain, R. J. Maude, S. Jaeger, and G. R. Thoma, "Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images," *PeerJ*, vol. 6, p. e4568, Apr. 2018.
[80] H. Chougrad, H. Zouaki, and O. Alheyane, "Deep convolutional neural networks for breast cancer screening," *Comput. Methods Programs Biomed.*, vol. 157, pp. 19–30, Apr. 2018.
[81] L. Xavier, A. Larhmam, M. Saïd, and I. Nedjar, "Assessing breast cancer screening using recent deep convolutional neural networks," *Int. J. Comput. Assist. Radiol. Surg.*, vol. 13, pp. S100–S102, Jun. 2018.
[82] Kaggle. (2019). *Aptos 2019 Blindness Detection*. [Online]. Available: https://www.kaggle.com/c/aptos2019-blindness-detection
[83] D. S. Kermany, M. Goldbaum, W. Cai, C. C. S. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, and J. Dong, "Identifying medical diagnoses and treatable diseases by image-based deep learning," *Cell*, vol. 172, no. 5, pp. 1122–1131.e9, Feb. 2018.
[84] M. Arredondo-Santoyo, C. Domínguez, J. Heras, E. Mata, V. Pascual, M. S. Vázquez-Garcidueñas, and G. Vázquez-Marrufo, "Automatic characterisation of dye decolourisation in fungal strains using expert, traditional, and deep features," *Soft Comput.*, vol. 23, no. 23, pp. 12799–12812, Dec. 2019.
[85] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, vol. 1288, Aug. 2013, pp. 1–8.
[86] P. Tschandl, C. Rosendahl, and H. Kittler, "The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," *Sci. Data*, vol. 5, Mar. 2018, Art. no. 180161.
[87] Kaggle. (2019). *Open Sprayer Images*. [Online]. Available: https://www.kaggle.com/gavinarmstrong/open-sprayer-images
[88] G. Cheng, J. Han, and X. Lu, "Remote sensing image scene classification: Benchmark and state of the art," *Proc. IEEE*, vol. 105, no. 10, pp. 1865–1883, Oct. 2017.
[89] J. Kumar, P. Ye, and D. Doermann, "Structural similarity for document image classification and retrieval," *Pattern Recognit. Lett.*, vol. 43, pp. 119–126, Jul. 2014.
[90] Y. Yang and S. Newsam, "Bag-of-visual-words and spatial extensions for land-use classification," in *Proc. 18th SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst. (GIS)*, 2010, pp. 270–279.
[91] Z.-H. Zhou and M.-L. Zhang, "Multi-instance multi-label learning with application to scene classification," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*. Vancouver, BC, Canada: MIT Press, 2007, pp. 1609–1616.
[92] B. Chaudhuri, B. Demir, S. Chaudhuri, and L. Bruzzone, "Multilabel remote sensing image retrieval using a semisupervised graph-theoretic method," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 2, pp. 1144–1158, Feb. 2018.
[93] Kaggle. (2020). *Apparel Images Dataset*. [Online]. Available: https://www.kaggle.com/trolukovich/apparel-images-dataset
[94] N. Orlov, L. Shamir, T. Macura, J. Johnston, D. M. Eckley, and I. G. Goldberg, "WND-CHARM: Multi-purpose image classification using compound image transforms," *Pattern Recognit. Lett.*, vol. 29, no. 11, pp. 1684–1693, Aug. 2008.
[95] L. Shamir, N. Orlov, D. Mark Eckley, T. J. Macura, and I. G. Goldberg, "IICBU 2008: A proposed benchmark suite for biological image analysis," *Med. Biol. Eng. Comput.*, vol. 46, no. 9, pp. 943–947, Sep. 2008.

MANUEL GARCÍA-DOMÍNGUEZ received the B.S. degree in computer science, and the M.S. degree in computer technology from the University of La Rioja, Logroño, Spain, in 2016 and 2017, respectively, where he is currently pursuing the Ph.D. degree in computer science. His research interests include machine learning and image analysis.

CÉSAR DOMÍNGUEZ received the B.S. degree in mathematics from the University of Zaragoza, Zaragoza, Spain, in 1996, and the Ph.D. degree in mathematics from the University of La Rioja, Logroño, Spain, in 2003. He is currently an Associate Professor with the University of La Rioja. His research interests include formal methods, e-learning, machine learning, deep learning, and image analysis.

JÓNATHAN HERAS received the B.S. degree in mathematics, the B.S. degree in computer science, and the Ph.D. degree in computer science from the University of La Rioja, Logroño, Spain, in 2007, 2008, and 2011, respectively. He is currently an Associate Professor with the University of La Rioja. His research interests include machine learning, deep learning, and computer vision.

ELOY MATA received the B.S. degree in mathematics from the University of Zaragoza, Zaragoza, Spain, in 1987, and the Ph.D. degree in mathematics from the University of La Rioja, Logroño, Spain, in 2009. He is currently an Associate Professor with the University of La Rioja. His research interests include web services, machine learning, and high performance computing.

VICO PASCUAL received the B.S. degree in mathematics from the University of Zaragoza, Zaragoza, Spain, in 1994, and the Ph.D. degree in mathematics from the University of La Rioja, Logroño, Spain, in 2002. She is currently an Associate Professor with the University of La Rioja. Her research interests include machine learning, deep learning, and computer vision.