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

M. García-Domínguez*, C. Domínguez, J. Heras, E. Mata, V. Pascual

*University of La Rioja, Department of Mathematics and Computer Science, Ed. CCT, C. Madre de Dios, 53, E-26004, Logroño, La Rioja, Spain*
*{manuel.garciad,cesar.dominguez,jonathan.heras,eloy.mata,mvico}@unirioja.es*

**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 the FrImCla framework, 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 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.

*Keywords:* Deep learning, Transfer learning, Image classification, AutoML

*Corresponding author
*Email address:* `manuel.garciad@unirioja.es` (M. García-Domínguez)

## 1. 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, 3, 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.

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, 8, 9, 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 2. 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 image classification in a simple way, see Section 3. In order to illustrate the use of FrImCla and its feasibility to construct useful models from small image classification datasets, we study two different biomedical datasets in Section 4; and in addition, we compare FrImCla with other AutoML tools in Section 5. 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.

## 2. 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], Rapid-

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

- We have developed FrImCla, the first, up to the best of our kwnoledge, AutoML tool for image classification based on transfer learning. Moreover, we have designed FrImCla as a modular tool (see Section 3). 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 4).
- 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 5).

## 3. Methods

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; OpenCV [45], to extract traditional computer vision features; Keras [37], to extract deep learning features; and cPickle [46], to serialize objects.

### *The workflow of FrImCla*

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.

### *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.

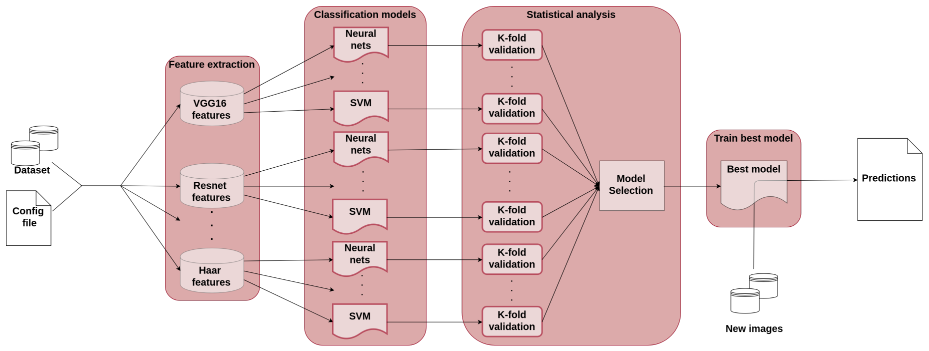
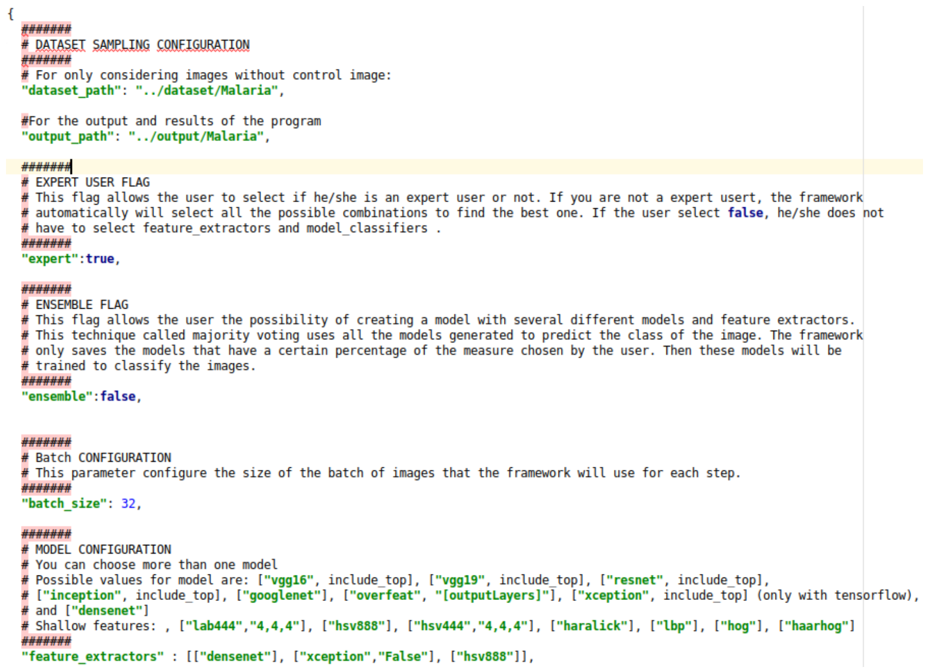


Figure 1: Workflow of the FrImCla framework

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 [47]. 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 five parameters: the dataset of images, the output path, a selection of methods for feature extraction, the supervised learning algorithms and the metric to measure the performance of the generated models (an example is given in Figure 2). The dataset of images is given by the path where the images are stored, the images are expected to be organised in folders that annotate the category of each image. The output path provides the location where the result of FrImCla will be stored. If the user has not knowledge about the feature extractors and supervised learning algorithms to use, FrImCla has an option that uses all the possible combinations. 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 [48]. This is a very useful tool since the user has a detailed explanation of each step that must be executed to use FrImCla (see Figure 3).

## *Feature extraction*

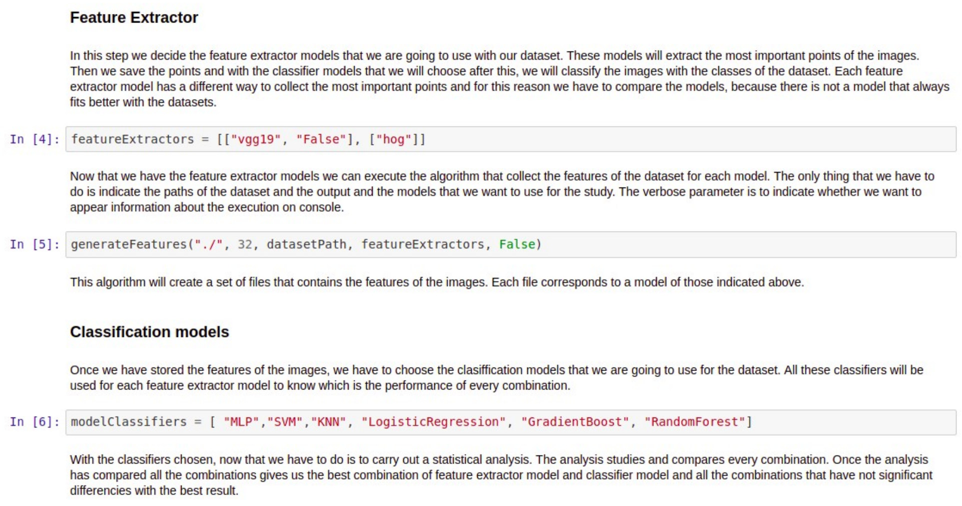
FrImCla offers several feature extractor methods (employing both traditional and transfer-learning techniques) to describe the dataset of images. In the case



```
{
  #######
  # DATASET SAMPLING CONFIGURATION
  #######
  # For only considering images without control image:
  "dataset_path": "../dataset/Malaria",

  #For the output and results of the program
  "output_path": "../output/Malaria",

  #######
  # EXPERT USER FLAG
  # This flag allows the user to select if he/she is an expert user or not. If you are not a expert usert, the framework
  # automatically will select all the possible combinations to find the best one. If the user select false, he/she does not
  # have to select feature_extractors and model_classifiers .
  #######
  "expert":true,

  #######
  # ENSEMBLE FLAG
  # This flag allows the user the possibility of creating a model with several different models and feature extractors.
  # This technique called majority voting uses all the models generated to predict the class of the image. The framework
  # only saves the models that have a certain percentage of the measure chosen by the user. Then these models will be
  # trained to classify the images.
  #######
  "ensemble":false,


  #######
  # Batch CONFIGURATION
  # This parameter configure the size of the batch of images that the framework will use for each step.
  #######
  "batch_size": 32,

  #######
  # MODEL CONFIGURATION
  # You can choose more than one model
  # Possible values for model are: ["vgg16", include_top], ["vgg19", include_top], ["resnet", include_top],
  # ["inception", include_top], ["googlenet"], ["overfeat", "[outputLayers]"], ["xception", include_top] (only with tensorflow),
  # and ["densenet"]
  # Shallow features: , ["lab444","4,4,4"], ["hsv888"], ["hsv444","4,4,4"], ["haralick"], ["lbp"], ["hog"], ["haarhog"]
  #######
  "feature_extractors" : [["densenet"], ["xception","False"], ["hsv888"]],
```

Figure 2: JSON configuration file

of traditional computer vision feature extractors, FrImCla includes LAB and HSV histograms [49], Haralick features [50] and HOG features [51] — 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 [52], as feature extractors; namely, the user can select among VGG16 [53], VGG19 [53], DenseNet [11], ResNet [13], Inception [54], GoogleNet [12], Overfeat [55] and Xception [56]. 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


**Feature Extractor**

In this step we decide the feature extractor models that we are going to use with our dataset. These models will extract the most important points of the images. Then we save the points and with the classifier models that we will choose after this, we will classify the images with the classes of the dataset. Each feature extractor model has a different way to collect the most important points and for this reason we have to compare the models, because there is not a model that always fits better with the datasets.

```
In [4]: featureExtractors = [["vgg19", "False"], ["hog"]]
```

Now that we have the feature extractor models we can execute the algorithm that collect the features of the dataset for each model. The only thing that we have to do is indicate the paths of the dataset and the output and the models that we want to use for the study. The verbose parameter is to indicate whether we want to appear information about the execution on console.

```
In [5]: generateFeatures("./", 32, datasetPath, featureExtractors, False)
```

This algorithm will create a set of files that contains the features of the images. Each file corresponds to a model of those indicated above.

**Classification models**

Once we have stored the features of the images, we have to choose the clasiffication models that we are going to use for the dataset. All these classifiers will be used for each feature extractor model to know which is the performance of every combination.

```
In [6]: modelClassifiers = [ "MLP","SVM","KNN", "LogisticRegression", "GradientBoost", "RandomForest"]
```

With the classifiers chosen, now that we have to do is to carry out a statistical analysis. The analysis studies and compares every combination. Once the analysis has compared all the combinations gives us the best combination of feature extractor model and classifier model and all the combinations that have not significant differencies with the best result.

Figure 3: Fragment of a Jupyter notebook

challenge. The features extracted from images are used to train the classification models.

*Classification models*

Similarly to the case of feature extraction methods, FrImCla users can select among several supervised learning algorithms including SVM [57], KNN [58], Neural networks [59], Gradient Boost [60], Logistic Regression [61] and Random Forest [62] — this list of algorithms can be easily extended to incorporate other methods. In addition, FrImCla includes an ensemble technique called majority voting [63]. 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, this practice gives better results than using just one model for prediction; however, it takes more time because it uses several models to obtain the prediction.

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 AUROC.

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

*Statistical Analysis*

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 Shapiro-Wilk [64] to check normality — with the null hypothesis being that the data follow a normal distribution — and a Levene test [65] 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 [66], 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 d [67] and Eta Squared [68]. 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.

*Output of the framework*

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.

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.

## 4. Results

In order to test the suitability of FrImCla, we consider two dataset of images: the MIAS dataset of images [69] (devoted to study whether an abnormality appears in a mammographic image) and a malaria dataset [70]. 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.

*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 $150 \times 150$. 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 90% for training and 10% 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, Xception, 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 machine learning algorithms: SVM, KNN, Multilayer perceptron (MLP), Gradient Boost (GB), Logistic Regression (LR) and Random Forest (RF).

The results are presented in Tables 1 and 2 and Figure 4. In order to compare all the possible combinations, we analyse the statistical study performed by FrImCla. The analysis has two steps, first to collect which is the best machine learning algorithm for each feature extractor and, then, to know which is the best combination. From the first analysis, see Table 1 and the left side of Figure 4, 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 the right side of Figure 4, 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 ($F = 8.44$; $p = 2.40 \times 10^{-13}$) among the models, with a large size effect $\eta^2 = 0.423197$ (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 92.9%. The result with the test set is a 90.91% of accuracy.

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 [71] and achieved an accuracy of 98% using fine-tuning — a transfer learning technique. In [71], they fine-tuned the networks VGG16, Resnet50 and Inception v3 for 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.

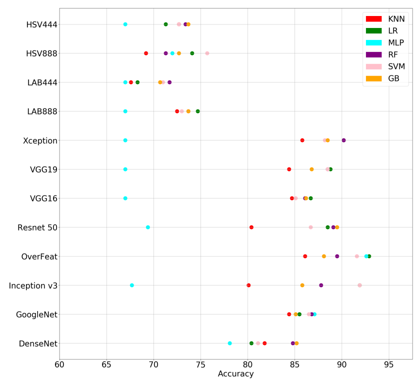
Up to the best of our knowledge, the best model for the MIAS dataset built only from images of this dataset was presented in [72], and also used transfer-learning from natural images, as in our case. In [72], the networks Inception, Xception and MobileNet were fine-tuned achieving an accuracy of 90% in the test set; such an accuracy is similar to ours, but we did not require the use of GPUs.

### *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 150x150. These images can be classified into two classes: parasitized and uninfected. There are 13,779 for each class. For this example,

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 | KNN | LR | MLP | RF | SVM | GB | Test (ANOVA or Friedman) | After post-hoc procedure |
|---|---|---|---|---|---|---|---|---|
| DenseNet | 81.8 (11.1) | 80.4 (7.3) | 78.1 (10.1) | 84.8 (7.9) | 81.1 (5.8) | *85.2 (9.3)* | 0.10 | GB $\simeq$ RF, KNN, SVM, LR, MLP |
| GoogleNet | 84.4 (7.0) | 85.5 (5.4) | *87.1 (6.7)* | 86.8 (6.9) | 86.5 (7.8) | 85.1 (7.6) | 0.06 | MLP$\simeq$ RF, SVM, LR, GB, KNN |
| Inception v3 | 80.1 (8.1) | *91.9 (3.8)* | 67.7 (11.9) | 87.8 (4.1) | 91.9 (5.1) | 85.8 (4.0) | 22.59$^{***}$ | LR $\simeq$ SVM, RF; LR $>$ GB, KNN, MLP |
| Overfeat | 86.1 (8.5) | ***92.9 (6.1)*** | 92.6 (7.2) | 89.5 (5.6) | 91.6 (7.0) | 88.1 (7.0) | 2.04 | LR $\simeq$ MLP, SVM, RF, GB, KNN |
| Resnet 50 | 80.4 (8.1) | 88.5 (5.8) | 69.4 (12.3) | *89.1 (6.5)* | 86.7 (8.4) | 89.5 (4.9) | 9.34$^{***}$ | RF $\simeq$ GB, LR, SVM; RF $>$ KNN, MLP |
| VGG16 | 84.7 (9.4) | *86.7 (10.0)* | 67.0 (10.8) | 86.1 (8.4) | 85.1 (9.5) | 86.2 (7.7) | 3.50$^{**}$ | LR $\simeq$ RF, GB, SVM, KNN; LR$>$ MLP |
| VGG19 | 84.4 (6.1) | *88.8 (7.6)* | 67.0 (10.8) | 88.5 (6.4) | 88.5 (6.4) | 86.8 (6.8) | 8.43$^{***}$ | LR$\simeq$ SVM, RF, GB, KNN; LR $>$ MLP |
| Xception | 85.8 (6.3) | 90.2 (4.5) | 67.0 (10.8) | *90.2 (3.8)* | 88.2 (4.4) | 88.5 (6.2) | 8.49$^{***}$ | LR $\simeq$ RF, GB, SVM, KNN; LR $>$ MLP |
| LAB888 | 72.5 (10.9) | *74.7 (9.8)* | 67.0 (10.8) | 73.7 (9.1) | 73.0 (9.2) | 73.7 (10.9) | 0.25 | LR $\simeq$ GB, RF, SVM, KNN, MLP |
| LAB444 | 67.6 (10.9) | 68.3 (9.1) | 67.0 (10.8) | *71.7 (10.5)* | 71.0 (10.7) | 70.7 (10.1) | 1.18 | RF $\simeq$ SVM, GB, LR, KNN, MLP |
| HSV888 | 69.2 (11.4) | 74.1 (11.4) | 72.0 (10.2) | 71.3 (7.6) | *75.7 (10.7)* | 72.7 (9.7) | 1.43 | SVM $\simeq$ LR, GB, MLP, RF, KNN |
| HSV444 | 72.7 (11.8) | 71.3 (10.9) | 67.0 (10.8) | *73.4 (12.0)* | 72.7 (12.0) | 73.7 (11.1) | 3.97$^{**}$ | RF $\simeq$ GB, KNN, SVM, LR; RF$>$ MLP |



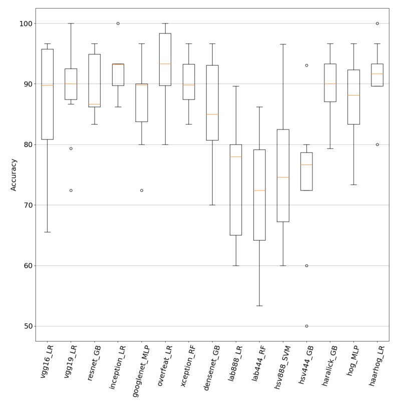


Figure 4: **Left.** Scatter diagram of the MIAS dataset. **Right.** Box diagram of the MIAS dataset

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 $d$ |
|---|---|---|---|---|
| Inception-LR | 0.54 | 0.59 | 1.00 | 0.19 |
| Xception-LR | 1.27 | 0.20 | 0.82 | 0.49 |
| Vgg19-LR | 1.36 | 0.17 | 0.83 | 0.57 |
| Resnet 50-RF | 1.53 | 0.13 | 0.76 | 0.57 |
| Vgg16-LR | 1.73 | 0.08 | 0.58 | 0.71 |
| Googlenet-MLP | 2.07 | 0.04 | 0.31 | 0.86 |
| DenseNet-GB | 2.14 | 0.03 | 0.29 | 0.93 |
| HSV888-SVM | 4.32 | $1.55361 \times 10^{-5}$ | $1.71 \times 10^{-4}$ | 1.87 |
| LAB888-LR | 4.46 | $8.13 \times 10^{-6}$ | $9.75 \times 10^{-5}$ | 2.11 |
| HSV444-RF | 4.63 | $3.72 \times 10^{-6}$ | $4.84 \times 10^{-5}$ | 1.95 |
| LAB444-RF | 4.96 | $7.23 \times 10^{-7}$ | $1.01 \times 10^{-5}$ | 2.34 |

we use only 2,000 images (1,000 images per class) for training and the rest of the dataset for testing.

The results are presented in Tables 3 and 4 and Figure 5 using again all the feature extractors and machine learning algorithms available in FrImCla. 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 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 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 left side of Figure 5, 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. After conducting the statistical analysis using 2000 images, we employed the rest of the dataset for testing, obtaining a 99,8% 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 [70] 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 better accuracy of 99.8% and without using special purpose hardware.

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 |

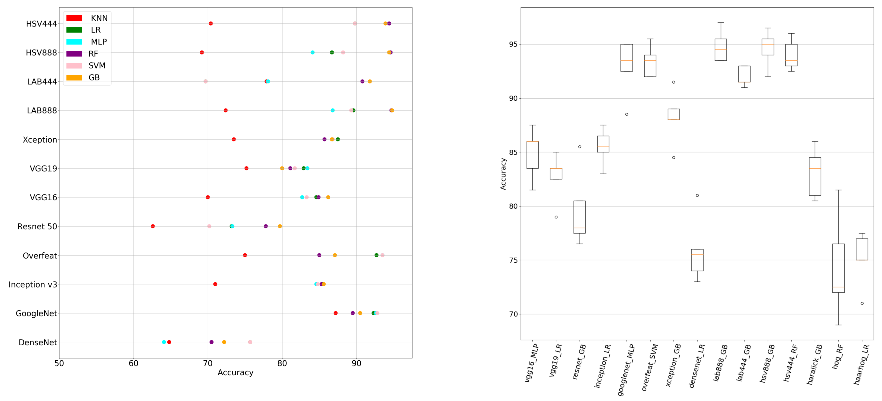


Figure 5: **Left.** Scatter diagram of the Malaria dataset. **Right.** Box diagram of the Malaria dataset

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 | $6.46 \times 10^{-3}$ | 0.06 | 6.34 |
| Resnet 50-GB | 3.09 | $2.01 \times 10^{-3}$ | 0.02 | 5.50 |
| DenseNet-LR | 3.82 | $1.34 \times 10^{-4}$ | $1.60 \times 10^{-3}$ | 7.80 |

## 5. 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 [73]. 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 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 3.

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 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.

Table 5: 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 |

We finish this section by comparing the performance of AutoKeras, Devol, FrimCla, Ludwig, and WND-CHARM by using 6 datasets from different types of image classification problems [73] — the sizes of the training sets, test sets, and the number of classes in each dataset are listed in Table 6. The accuracy achieved by each tool is presented in Table 7. 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 two datasets where other tools achieve a better accuracy, and even in those cases, the achieved 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.

Table 6: Datasets used for comparing AutoML tools

| Dataset | Number of classes | Number of training images | Number of test images |
|---|---|---|---|
| Binucleate | 2 | 30 | 11 |
| Cho | 5 | 245 | 82 |
| Hela | 10 | 508 | 181 |
| Lymphoma | 3 | 280 | 94 |
| Pollen | 7 | 472 | 158 |
| RNAi | 10 | 150 | 50 |

Table 7: Comparison of the performance of AutoKeras, Devol, FrImCla, Ludwig, and WND-CHARM for constructing in six image classification problems. The best result is highlighted in bold face

| | Binucleate | Cho | Hela | Lymphoma | Pollen | RNAi |
|---|---|---|---|---|---|---|
| FrImCla | **1.00** | **0.98** | 0.82 | 0.86 | **0.96** | **0.69** |
| AutoKeras | 0.55 | 0.96 | 0.47 | **0.89** | 0.81 | 0.24 |
| Devol | 0.54 | 0.75 | 0.68 | 0.55 | 0.89 | 0.28 |
| Ludwig | 0.54 | 0.64 | 0.51 | 0.57 | 0.58 | 0 |
| WND-CHARM | **1.00** | 0.95 | **0.88** | 0.79 | **0.96** | 0.66 |

## 6. Conclusions

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.

## Compliance with Ethical Standards

Funding: This work was partially supported by Ministerio de Economía, Industria y Competitividad [MTM2017-88804-P], and Agencia de Desarrollo Económico de La Rioja [ADER-2017-I-IDD-00018] and FPI grant from Community of La Rioja 2018.

Conflict of Interest: All the authors declare that they have no conflict of interest.

Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

## References

[1] S. Akçay, M. E. Kundegorski, M. Devereux, T. P. Breckon, Transfer learning using convolutional neural networks for object classification within x-ray baggage security imagery, in: 2016 IEEE International Conference on Image Processing (ICIP), 2016, pp. 1057–1061.

[2] A. Asperti, C. Mastronardo, The effectiveness of data augmentation for detection of gastrointestinal diseases from endoscopical images, in: Proceedings of the 11th International Joint Conference on Biomedical Engineering Systems and Technologies - Volume 2: KALSIMIS,, INSTICC, SciTePress, 2018, pp. 199–205. `doi:10.5220/0006730901990205`.

[3] E. Valle, et al., Data Depth and Design: Learning reliable models for Melanoma screening (11 2017).
URL `https://arxiv.org/abs/1711.00441`

[4] A. Galdran, et al., Data-Driven Color Augmentation Techniques for Deep Skin Image Analysis, CoRR abs/1703.03702.
URL `https://arxiv.org/abs/1703.03702`

[5] M. Mohri, A. Rostamizadeh, A. Talwalkar, Foundations of Machine learning, MIT press, 2012.

[6] A. S. Razavian, et al., CNN features off-the-shelf: An astounding baseline for recognition, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW'14), IEEE Computer Society, IEEE, Columbus, Ohio, USA, 2014, pp. 512–519.

[7] S. J. Pan, Q. Yang, A survey on transfer learning, IEEE Transactions on Knowledge and Data Engineering 22 (10) (2010) 1345–1359.

[8] S. Christodoulidis, et al., Multisource Transfer Learning With Convolutional Neural Networks for Lung Pattern Analysis, IEEE Journal of Biomedical and Health Informatics 21 (1) (2017) 76–84.

[9] M. Ghafoorian, et al., Transfer learning for domain adaptation in MRI: Application in brain lesion segmentation, in: Medical Image Computing and Computer-Assisted Intervention - MICCAI 2017, Springer International Publishing, Cham, 2017, pp. 516–524.

[10] A. Menegola, M. Fornaciali, R. Pires, F. V. Bittencourt, S. Avila, E. Valle, Knowledge transfer for melanoma screening with deep learning, in: 2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017), 2017, pp. 297–300. doi:10.1109/ISBI.2017.7950523.

[11] G. Huang, Z. Liu, L. van der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17), 2017, pp. 2261–2269.

[12] C. Szegedy, et al., Going deeper with convolutions, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15), IEEE Computer Society, IEEE, 2015, pp. 1–9.

[13] K. He, et al., Deep Residual Learning for Image Recognition, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16), IEEE Computer Society, IEEE, 2016, pp. 770–778.

[14] S. Garcia, et al., Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, Information Sciences 180 (2010) 2044–2064.

[15] D. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, CRC Press, 2011.

[16] D. Sculley, et al., Hidden technical debt in machine learning systems, in: Advances in Neural Information Processing Systems, 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, E. Viegas, AutoML challenge 2015: Design and first results, in: Proc. of AutoML 2015@ICML, 2015. URL `https://drive.google.com/file/d/0BzRGLkqgrI-qWkpzcGw4bFpBMUk/view`

[18] F. Hutter, L. Kotthoff, J. Vanschoren (Eds.), Automated Machine Learning: Methods, Systems, Challenges, Springer, 2018, in press, available at http://automl.org/book.

[19] J. Bergstra, Y. Bengio, Random Search for Hyper-Parameter Optimization, Journal of Machine Learning Research 13 (2012) 281–305.

[20] J. Snoek, et al., Practical Bayesian Optimization of Machine Learning Algorithms, Advances in Neural Information Processing Systems 25 (2012) 2951–2959.

[21] D. H. Wolpert, W. G. Macready, et al., No free lunch theorems for optimization, IEEE transactions on evolutionary computation 1 (1) (1997) 67–82.

[22] F. Hutter, H. H. Hoos, K. Leyton-Brown, Sequential Model-Based Optimization for General Algorithm Configuration, in: Proceedings of the 2011 International Conference on Learning and Intelligent Optimization, Vol. 6683 of Lecture Notes in Computer Science, Springer, 2011, pp. 507–523.

[23] F. Hutter, H. H. Hoos, K. 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.

[24] M. Reif, F. Shafait, M. Goldstein, T. Breuel, A. Dengel, Automatic classifier selection for non-experts, Pattern Analysis and Applications 17 (1) (2012) 83–96.

[25] M. Feurer, et al., Photoelectric Color-Difference Meter, in: Advances in Neural Information Processing Systems, Vol. 28, 2015, pp. 2962–2970.

[26] J. M. Kanter, K. Veeramachaneni, Deep Feature Synthesis: Towards Automating Data Science Endeavors, in: Proceedings of the International Conference on Data Science and Advance Analytics, IEEE, 2015.

[27] R. Olson, et al., Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 – April 1, 2016, Proceedings, Part I, Springer International Publishing, 2016, Ch. Automating Biomedical Data Science Through Tree-Based Pipeline Optimization, pp. 123–137. `doi:10.1007/978-3-319-31204-0_9`.
URL `http://dx.doi.org/10.1007/978-3-319-31204-0_9`

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

[29] M. Kuhn, Building Predictive Models in R Using the caret Package, Journal of Statistical Software 28 (5) (2008) 1–26.

[30] C. Steinruecken, et al., The Automatic Statistician, in: F. Hutter, L. Kotthoff, J. Vanschoren (Eds.), Automated Machine Learning: Methods, Systems, Challenges, Series on Challenges in Machine Learning, Springer, 2019. `doi:10.1007/978-3-030-05318-5_9`.

[31] T. Elsken, J. H. Metzen, F. Hutter, Neural architecture search: A survey, Journal of Machine Learning Research 20 (2019) 1–21.

[32] Y. Jaafra, et al., Reinforcement learning for neural architecture search: A review, Image and Vision Computing 89 (2019) 57–66.

[33] J. Liang, E. Meyerson, R. Mikkulainen, Evolutionary architecture search for deep multitask networks, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'18), GECCO'18, ACM, 2018, pp. 466–473.

[34] B. Zoph, et al., Learning Transferable Architectures for Scalable Image Recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18), CVPR'18, 2018, pp. 1–14.

[35] E. Real, et al., Regularized Evolution for Image Classifier Architecture Search, in: Proceedings of the the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI'19), Vol. 33 of AAAI'19, 2019.

[36] H. Jin, Q. Song, X. Hu, Auto-keras: An efficient neural architecture search system, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, 2019, pp. 1946–1956.

[37] F. Chollet, Keras (2017).
URL `https://keras.io/`

[38] J. Davison, DEvol - deep neural network evolution (2018).
URL `https://github.com/joeddav/devol`

[39] P. Molino, Y. Dudin, S. S. Miryala, Ludwig: a type-based declarative deep learning toolbox, CoRR abs/1909.07930.
URL `https://arxiv.org/abs/1909.07930`

[40] Google, TensorFlow: An open-source software library for Machine Intelligence (2018).
URL `https://www.tensorflow.org`

[41] T. Kraska, A. Talwalkar, J. Duchi, MLbase: A Distributed Machine-learning System, in: Conference on Innovative Data Systems Research, 2013.

[42] Google, Google cloud automl (2018).
URL https://cloud.google.com/automl/

[43] P. Domingos, A few useful things to know about machine learning, Communications of the ACM 55 (10) (2012) 78–87.

[44] D. Cournapeau, Scikit-learn (2018).
URL http://scikit-learn.org/stable/

[45] G. Bradski, The OpenCV Library, Dr. Dobb's Journal of Software Tools.

[46] Python software foundation, Cpickle library (2018).
URL https://docs.python.org/2/library/pickle.html

[47] P. P. Authority, Pip (2011).
URL https://pypi.org/project/pip/

[48] D. Avila, et al., Project jupyter (2018).
URL https://jupyter.org/

[49] R. S. Hunter, Photoelectric Color-Difference Meter, Journal of the Optical Society of America 38 (7) (1948) 661.

[50] R. M. Haralick, K. Shanmugam, I. Dinstein, Textural features for image classification, IEEE Transactions on Systems, Man and Cybernetics SMC-3 (6) (1973) 610–621.

[51] N. Dalal, B. Triggs, Histograms of Oriented Gradients for Human Detection, in: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01, CVPR '05, IEEE Computer Society, 2005, pp. 886–893.

[52] O. Russakovsky, et al., ImageNet Large Scale Visual Recognition Challenge, International Journal of Computer Vision 115 (3) (2015) 211–252.

[53] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, CoRR abs/1409.1556.
URL http://arxiv.org/abs/1409.1556

[54] C. Szegedy, et al., Rethinking the inception architecture for computer vision, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2818–2826. `doi:10.1109/CVPR.2016.308`.

[55] P. Sermanet, et al., OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks, in: International Conference on Learning Representations (ICLR2014), CBLS, April 2014, 2014.

[56] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1800–1807. `doi:10.1109/CVPR.2017.195`.

[57] C. Cortes, V. Vapnik, Support-Vector Networks, Machine Learning 20 (3) (1995) 273–297.

[58] T. Cover, P. Hart, Nearest Neighbor Pattern Classification, IEEE Trans. Inf. Theor. 13 (1) (2006) 21–27.

[59] C. M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.

[60] J. H. Friedman, Greedy function approximation: A gradient boosting machine., Ann. Statist. 29 (5) (2001) 1189–1232. `doi:10.1214/aos/1013203451`.

[61] P. McCullagh, J. A. Nelder, Generalized Linear Models, Chapman & Hall, 1989.

[62] L. Breiman, Random Forests, Machine Learning 45 (1) (2001) 5–32.

[63] C. Zhang, Y. Ma, Ensemble Machine Learning: Methods and Applications, Springer Publishing Company, Incorporated, 2012.

[64] S. S. Shapiron, M. B. Wilk, An analysis for variance test for normality (complete samples), Information Sciences 180 (1965) 2044–2064.

[65] H. Levene, Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling, Stanford University Press, 1960, Ch. Robust tests for equality of variances, pp. 278–292.

[66] O. S. Holm, A simple sequentially rejective multiple test procedure, Scandinavian Journal of Statistics 6 (1979) 65–70.

[67] J. Cohen, Statistical Power Analysis for the Behavioral Sciences, Academic Press, 1969.

[68] J. Cohen, Eta-squared and partial eta-squared in fixed factor anova designs, Educational and Psychological Measurement 33 (1973) 107–112.

[69] J. Suckling, J. Parker, D. Dance, S. Astley, I. Hutt, C. Boggis, I. Ricketts, E. Stamatakis, N. Cerneaz, S. e. a. Kok, Mammographic image analysis society (mias) database v1.21 (2018).
URL `https://www.repository.cam.ac.uk/handle/1810/250394`

[70] S. Rajaraman, S. K. Antani, M. Poostchi, K. Silamut, M. A. Hossain, R. J. Maude, S. Jaeger, G. R. Thoma, Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images, PeerJ.
URL `https://lhncbc.nlm.nih.gov/system/files/pub9752.pdf`

[71] H. Chougrad, H. Zouaki, O. Alheyane, Deep convolutional neural networks for breast cancer screening, Computer Methods and Programs in Biomedicine 157 (2018) 19 – 30. `doi:https://doi.org/10.1016/j.cmpb.2018.01.011`.
URL `http://www.sciencedirect.com/science/article/pii/S0169260717301451`

[72] L. Xavier, A. Larhmam, M. Saïd, I. Nedjar, Assessing breast cancer screening using recent deep convolutional neural networks, International Journal of Computer Assisted Radiology and Surgery.

[73] N. Orlov, et al., Wnd-charm: Multi-purpose image classification using compound image transforms, Pattern Recognition Letters 29 (11) (2008) 1684–1693.