AutoML using Transfer and Semi-Supervised Learning

Adrián Inés and **César Domínguez** and **Jónathan Heras** and **Eloy Mata** and **Vico Pascual**[1]

Abstract. Deep learning techniques are the state-of-the-art approach to solve image classification problems; however, they are very data and computational demanding — that is, they require a lot of annotated images and also special-purpouse hardware like GPUs or TPUs. In addition, using deep learning libraries and tuning the hyperparameters of the networks trained with them might be challenging for several users. These drawbacks prevent the adoption of these techniques in several contexts outside the machine-learning community. To deal with these problems, we present an AutoML tool that uses transfer learning to train models with small datasets, and a new semi-supervised learning procedure to train models when few annotated images are available. Our AutoML tool also faces the usability problem thanks to a graphical interface that guides the user from the annotation of images, up to the construction and use of the classification models. The benefits of our methods have been thoroughly analysed with several small datasets, and partially annotated datasets.

1 INTRODUCTION

Deep learning techniques, and namely convolutional neural networks, have become the state-of-the-art approach to solve image classification problems in a wide variety of fields such as biology [1], security [2], or medicine [4]. However, building image classification models using deep learning methods might be challenging for non-expert users due to several reasons. First of all, deep learning techniques require thousands of images, and acquiring such an amount of images in some context, like biomedicine, might be difficult [5]. In addition, even if it is possible to obtain the necessary images, it still remains the task of manually annotating them, a tedious and time-consuming process that might require specialised knowledge [27]. Furthermore, deep learning methods are not only data demanding but also computationally demanding, and require special-purpose hardware like GPUs or TPUs [29]. Finally, training a deep model poses several technical issues like installing and using deep learning libraries, choosing several hyperparameters, such as the architecture of the network or the optimisation algorithm, or organising and processing the data in a proper way [39].

The deep learning community has already tackled these challenges. Techniques like *transfer learning* [35] and *data augmentation* [40] have shown that it is feasible to train deep learning models with a limited amount of data and computational resources [35]. In addition, semi-supervised techniques, like data and model distillation [24], take advantage of unlabelled data for training; and, therefore, reduce the burden of annotating images [44]. Moreover, there are several Automated Machine Learning (AutoML) techniques that automatically select the best model to solve a task without user intervention [25]. In spite of these advances, the construction of image classification models using deep learning methods is far from trivial. Techniques like transfer learning, data augmentation, or data and model distillation require a considerable experience working with several deep learning libraries and tools; and their use produce in many cases *pipeline jungles* [39]. Neither AutoML tools are the panacea, since they are usually computationally intensive and also require some coding experience to use and configure the libraries that implement the AutoML methods.

In this work, we seek to overcome the aforementioned problems by providing an AutoML pipeline that combines transfer learning with data and model distillation in a fully automated way, and only using limited resources. In particular, the contributions of this work are the following:

- We explore different alternatives to combine transfer learning, and data and model distillation with limited resources; namely, using small annotated datasets, and using the free GPU setting provided by Google Colaboratory [12]. As a by-product, we present an automatic pipeline for training classification models that combines transfer learning, and a new semi-supervised learning method based on the notions of data and model distillation.
- We present a set of open-source tools — namely, a standalone application and a suite of Jupyter notebooks [6] — that allows non-expert users to easily employ our method to construct their own classification models. This set of tools is available at `https://github.com/adines/AnnotationTool`.
- Finally, we conduct a thorough analysis for our methods and show their performance compared with other AutoML tools for constructing image classification models when working with small datasets. Moreover, we show the benefits of our semi-supervised method when partially annotated datasets are employed.

The rest of this paper is organised as follows. In the next section, we provide the necessary background to fully understand the rest of the paper. Subsequently, we present our methods and tools in Section 3, and evaluate them in Section 4. The paper ends with a section of conclusions and further work.

2 BACKGROUND

This work can be framed in the context of Automated Machine Learning (AutoML) research [25]. *AutoML* techniques aim to democratise machine learning by simplifying the construction and usage of models for domain experts that have a limited machine learning background. Since there are dozens of alternatives for each one of the steps of a machine learning pipeline (for instance, feature extraction, model selection or hyperparameter optimisation), AutoML techniques try to find the best combination for each particular problem. In the context of image classification, AutoML techniques are mainly focused on Neural Architecture Search (NAS); that is,

[1] Universidad de La Rioja, Spain, email: {adrian.ines, cesar.dominguez, jonathan.heras, eloy.mata, vico.pascual}@unirioja.es

automatically designing neural deep architectures [16]. Even if this approach has outperformed manually designed architectures [45], the adoption of NAS techniques by non-expert users to construct recognition models is far from trivial; mainly, because these techniques are data and computationally intensive (for instance, NasNet takes 1800 GPU days [45], and AmoebaNet takes 3150 GPU days [36]), require some prior knowledge about typical properties of architectures to simplify the search [16], and also some experience to properly configure the tools implementing those methods.

In spite of not being part of the AutoML toolbox, there are some well-established deep learning techniques that generally produce accurate classification models, are applicable in a wide variety of contexts, and do not require so many resources as AutoML tools [22, 37]. Among those procedures, *transfer learning* stands out when working with small datasets. *Transfer learning* [35] is a set of techniques that reuse a model trained in a source task in a new target task — usually the data available in the target task are much smaller than in the source task [11, 17]. Transfer learning techniques are based on the idea that convolutional neural networks are designed to learn a hierarchy of features; in particular, the initial layers of the network focus on generic features, whilst the later ones focus on specific features for the task at hand. *Fine-tuning* is a transfer learning technique that freezes the initial layers trained on a source task, while retraining the final ones for the specific problem. In this way, the generic features learned in the source task are re-used, and the specific features for the target task are learned. This approach has been successfully used to reduce the amount of resources to train state-of-the-art models. When working with a limited amount of images, transfer learning is usually combined with data augmentation.

Data augmentation [40] is a technique that generates new training samples from the original dataset by applying transformations that do not alter the class of the data. This method has proven to be effective to improve the accuracy of models, and reduce their generalisation error [30]. In addition, data augmentation can also be employed at test time by feeding to the model several transformations of a sample, and ensembling the predictions for the generated samples to obtain the final result, this procedure is known as test-time augmentation (TTA) [42]. TTA and ensembles are the basis of two semi-supervised learning techniques known as data and model distillation. These techniques might be helpful when we have access to a large corpus of images, but it is difficult, or time-consuming, to annotate them — for instance, in the biomedical context [43].

Data and model distillation are two forms of self-training [44]. Self-training techniques train a model on its own predictions, but this does not usually provide any benefit; hence, data and model distillation employ a different approach. In the case of data distillation [24], given a model trained on manually labelled data, this technique applies such a model to multiple transformations of unlabelled data, ensembles the multiple predictions, and, finally, retrains the model on the union of manually labelled data and automatically labelled data. In the case of model distillation [10], several models are employed to obtain predictions of unlabelled data; subsequently, those predictions are ensembled, and used to train a new model. Both techniques can also be combined as shown in [24].

Even if the aforementioned techniques have proven to be useful in several contexts; it might be challenging to use them since they can be applied in several ways (for instance, there are several base networks that can be employed for transfer learning, or transformation techniques for data augmentation). In addition, there is not a single library that provides all these techniques, and connecting different libraries might be difficult for non-expert users, and might produce pipeline jungles in the case of expert users. In this paper, we tackle these problems by providing an automatic pipeline to construct image classification models by combining transfer learning and a new semi-supervised method based on data and model distillation.

3 METHODS

In this section, we present a general pipeline to train image classification models, see Section 3.1, and a set of tools that allows non-expert users to apply our pipeline, see Section 3.2.

3.1 Workflow

Our workflow to train an image classification model is an iterative process that takes as input a partially annotated dataset; that is, a set of annotated images, and another set of unlabelled images. Each iteration of our process can be divided into two steps, as we can see in Figure 1. In Step (1), we use the labelled images to train a set of base models; and, in Step (2), we use this set of base models to automatically label the unlabelled images when a condition based on the confidence score is fulfilled. As a result of Step (2), a set of automatically labelled images, and a new set of unlabelled images are obtained. These two sets and the initial set of manually labelled images are the input of the next iteration. The process ends when there are not unlabelled images or after a fixed number of iterations; and the output of the process is the best model with respect to an independent test set. We have to remark that, in the first iteration, we start from a set of manually labelled images and a set of unlabelled images; hence, the set of automatically labelled images is empty. In the rest of this section, we detail how Steps (1) and (2) are conducted, and how this method can be particularised.

In Step (1), we train a set of base models applying fine-tuning from the Imagenet challenge and following the two-stage procedure presented in [22] — in this way, we take advantage of the numerous descriptors learned from the Imagenet dataset. In the first stage of the training process, we replace the last layers of the models (that is, the layers that give us the classification of the images), with new layers adapted to the number of classes of each particular dataset. Then, we train these new layers with the data of each particular dataset for two epochs. Since training only the last layers of a model may not be enough to obtain a good performance in the new dataset, it is necessary a second stage. In the second stage, we unfreeze the whole model and retrain all the layers of the model with the new data for eight epochs. When training all layers of the model, we should be careful with the learning rate used. The lower layers of the model have the most basic descriptors (colours, borders, shapes, ...), which are common for all images, and as we go up of our model, the descriptors become more specific to the used data. Thus, the idea is to modify the lowest descriptors minimally, and make a greater modification in the upper layers of the model. This is translated into using a low learning rate in the initial layers and a higher learning rate in the following layers. Then, to train our models, we use a learning rate slice $(\frac{lr}{100}, lr)$, that starts at a low learning rate $(\frac{lr}{100})$ for the lower layers and increases as we move through the layers until we reach lr in the higher layers. In addition, we look for this specific lr that improves the performance of our model; i.e., we select lr that decreases the loss to the minimum possible value using the approach presented in [41]. In particular, we select the learning rate with the lowest loss value, and, in case that this learning rate is too small ($lr < 1e^{-5}$), we change its value to $1e^{-3}$.

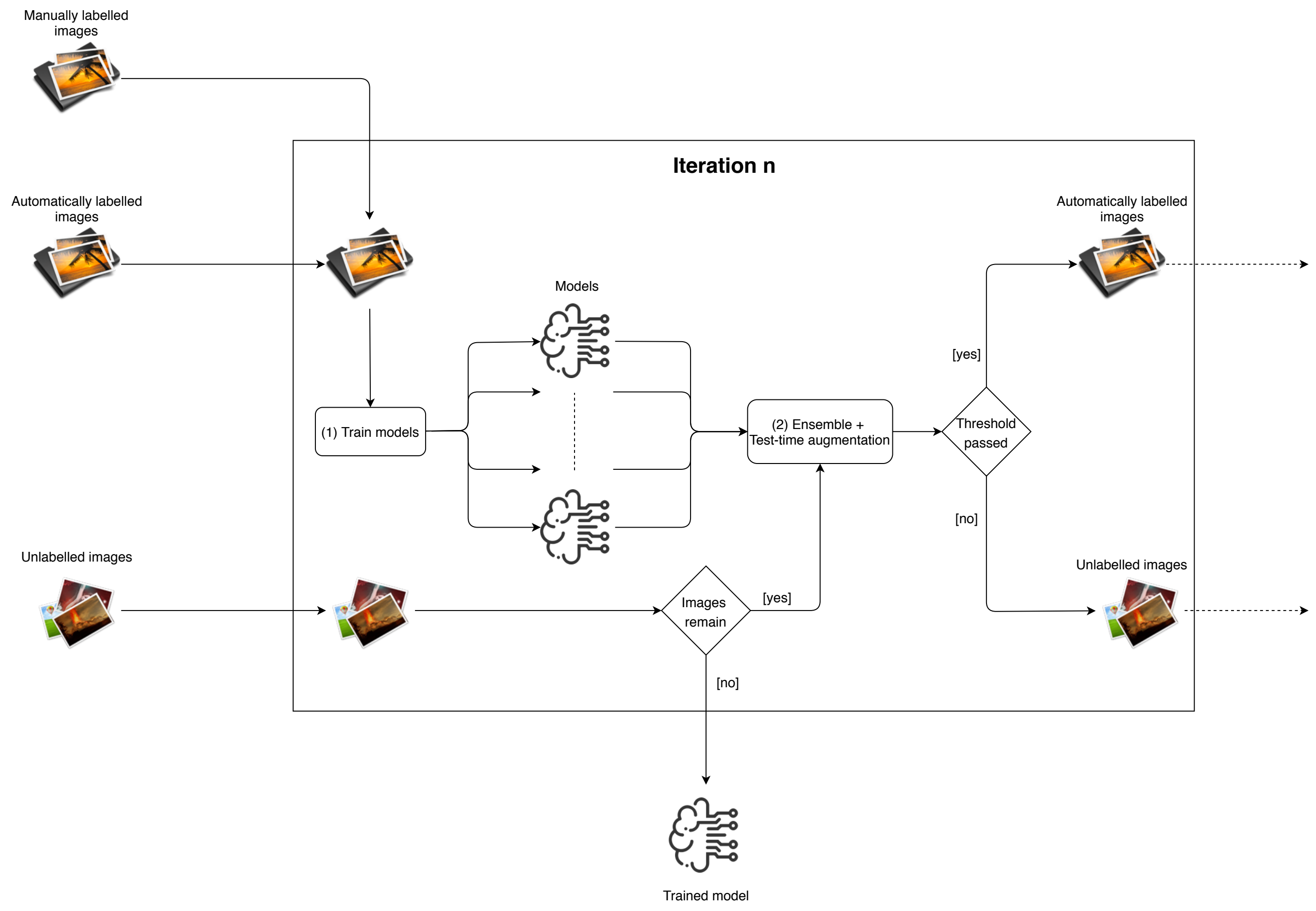


Figure 1. Workflow of an iteration of our semi-supervised learning process. **Inputs of the iteration**: set of manually labelled images, set of automatically labelled images and set of unlabelled images. **Outputs of the iteration**: set of automatically labelled images and set of unlabelled images. **Output of the process**: Trained model.

Once we have trained the set of base models, we classify the unlabelled images in Step (2) using those models. Namely, we use two techniques to obtain the final prediction: test-time augmentation and the ensemble of models. These techniques improve the accuracy of the predictions on the unlabelled images, and are applied as follows. First of all, test-time augmentation is applied to the set of unlabelled images, i.e., a set of transformations is applied to each image and as a result we obtain new images. Then, these new images are classified by each base model. In this way, for each image of the unlabelled dataset, we have $t+1$ images (where t is the number of applied transformations) and $m \times (t+1)$ predictions (where m is the number of different base models). To classify the original image, we make an ensemble of these predictions taking the most common class of the $m \times (t+1)$ predictions. To compute the confidence score of the ensembled prediction, the average of the confidence score of each of the $m \times (t+1)$ predictions is calculated. This confidence score is employed to decide whether the image is classified. Namely, if the confidence score for the image does not pass a fixed threshold, it is left unlabelled — that is, our automatic method only annotates those images that are classified with a high confidence score. Otherwise, the image is annotated with the predicted class. As a result of this step, two new sets, a set of unlabelled images and a set of automatically labelled images, are obtained.

This two-step process is iterated and ends when there are not unlabelled images — this condition can be replaced by others, such as a maximum number of iterations, or a condition on model improvement. Finally, the result of this process is a trained model; in particular, the model with the best performance in the last iteration.

It is worth nothing that the aforementioned process might be time consuming if there are lots of unlabelled images, several models, or iterations. To deal with this issue, our method can be particularised in several ways, reducing the time needed to finish the process. Namely, starting from the general process previously explained, seven particular cases can be obtained.

No distillation (N.D.): This is the most basic case and occurs when there are not unlabelled images in the original dataset, that is, when we start from a totally annotated dataset. Then, the process is reduced to Step (1), and the best base model with respect to an independent test set is selected.

Data distillation without thresholding (D.D.): This case appears when, instead of having a set of base models, we only have one model, and we do not set a threshold; that is, the threshold value is 0. In this case, the process does not iterate, since all the unlabelled images are annotated in the first iteration. This is the data distillation method presented in [24].

Iterative data distillation (I.D.D.): This case is very similar to the previous one. Again, instead of a set of models, a single model is trained. In addition, we establish a threshold to be passed, which leads us to have, probably, several iterations.

Model distillation without thresholding (M.D.): This case starts by training a set of models. In this case, we do not perform test-time augmentation of the unlabelled images, but only the ensemble of the model predictions is employed to label them. In addition, the value of the threshold is set to 0, that is, we do not have an iterative process. This is the model distillation procedure presented in [10].

Iterative model distillation (I.M.D.): In this case, we also start by training a set of models, and as in the previous case, we do not perform test-time augmentation. The main difference with the previous case is that we establish a threshold, which leads us to have an iterative process.

Model + Data distillation without thresholding (M.D.D): This case is very similar to the general workflow. We use a set of models and test-time augmentation to annotate the unlabelled images. The only difference is that the threshold is set to 0, and, therefore, we do not have an iterative process.

Iterative Model + Data distillation (I.M.D.D): This is the general workflow explained previously.

As we will show in Section 4, the general workflow and its particular cases can produce accurate models; however, implementing and using these techniques might be challenging for non-expert users. Therefore, we have developed a set of tools that facilitates the construction of image classification models using our techniques without writing a single line of code.

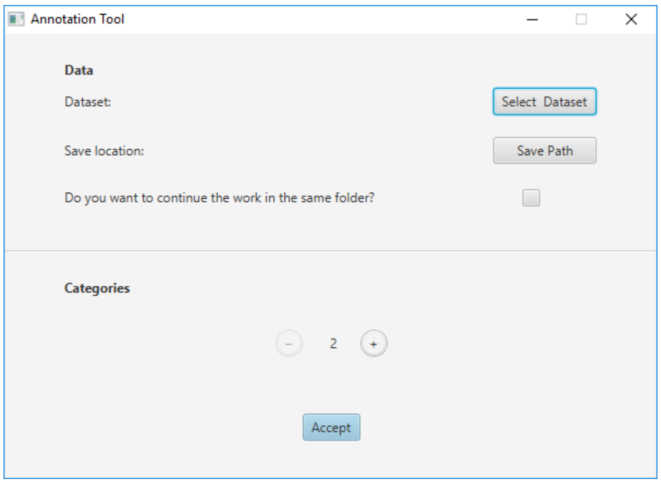
3.2 Set of tools

We have developed a suite of open-source tools that implements the workflow presented previously, and guides the user in all the stages of the construction of an image classification model using our workflow; namely, it helps to annotate the images, train a model, test it, and finally use it.

The first step for building an image classification model consists in annotating a set of images. To this aim, we have developed a graphical user interface (GUI) implemented in Java that provides all the necessary features to annotate a dataset of images. This includes the functionality to visualise and organise the images by categories, and manage the categories (that is, add, remove, and edit categories).

The workflow of this application is straightforward. When the GUI starts, the application asks the users to select the folder containing the dataset of images, the location where the annotated dataset will be saved and the number of categories of the dataset, see Figure 2. Then, the application automatically loads the images of the dataset and shows to the users the corresponding information, see Figure 3. From this window, the user can conduct the annotation process. Once the dataset has been annotated, or at least partially annotated, the training process can start.

The GUI includes a wizard that allows the users to configure the training process. First of all, the wizard detects whether there are unlabelled images in the dataset. If the dataset has been fully annotated, the users can select the networks that will be trained using the no distillation procedure. If the dataset contains unlabelled images, the


Annotation Tool

Data

Dataset:

Save location:

Do you want to continue the work in the same folder?

Categories

Figure 2. Initial screen of the developed GUI. From this interface, users can select the dataset, the location where they want to save the result, and the number of categories of the dataset.

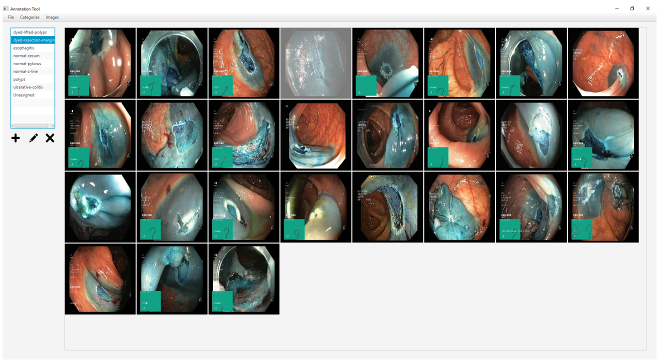


Figure 3. Example of use of the developed GUI where users can visualise the images organised by categories. Also, users can add new categories, edit or delete categories, and annotate images selecting them and choosing the particular category.

wizard asks what are the distillation methods that will be applied, see Figure 4. If the users select a data distillation method, they can select the augmentation techniques; if a model distillation method is chosen, the users can decide the base networks to be trained; and, if an iterative process is selected, the users must fix a threshold value, see Figure 5. The output of the wizard is a zip file, containing the organised dataset, and a Jupyter notebook with the necessary instructions to be executed.

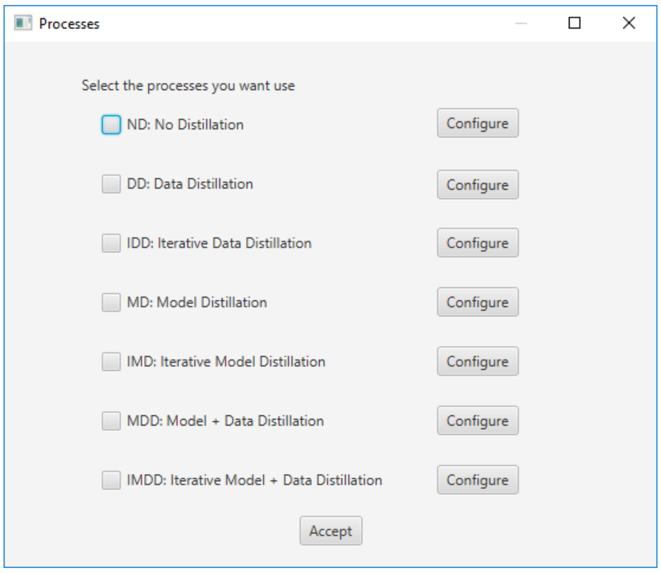

Processes
Select the processes you want use
ND: No Distillation
DD: Data Distillation
IDD: Iterative Data Distillation
MD: Model Distillation
IMD: Iterative Model Distillation
MDD: Model + Data Distillation
IMDD: Iterative Model + Data Distillation
Configure
Accept

Figure 4. Wizard window of our tool where users can select the seven different processes and configure them.

Jupyter notebooks are interactive documents that can include executable code together with explanations [6]. The Jupyter notebook generated by our wizard employs the FastAI library [22] to implement the workflow presented in Section 3.1 — other libraries like OpenCV [9] are also employed. The Jupyter notebook generated by the wizard can be run in the users' local computer provided that they have a GPU; and, since this might not be the case, the notebook can also be uploaded to Google Colaboratory [12], a free Jupyter Notebook environment that does not requires any configuration, runs completely in the cloud, and provides free access to GPUs.

Finally, the trained models can be used in different ways. One way to use these models is based on the same method used to train them: using a Jupyter notebook, either in Google Colaboratory or locally. Another way would be using the DeepClas4Bio API [26], an extensible API that facilitates the use of deep learning models. This API allows users to include their models easily in the API and use them. In addition, thanks to DeepClas4Bio, we can use our models from different image processing programs like ImageJ [38], Icy [14] or ImagePy [3].

As we have indicated previously, our tools are highly configurable; hence, some users may feel lost when using our assistant. To deal with this problem, we have conducted a study with two objectives: evaluate our methods and select the default techniques to use.

4 EVALUATION

In this section, we conduct a thorough analysis for our methods in two different scenarios. In particular, we test our methods in small annotated datasets, see Section 4.1; and in partially annotated datasets, see Section 4.2.

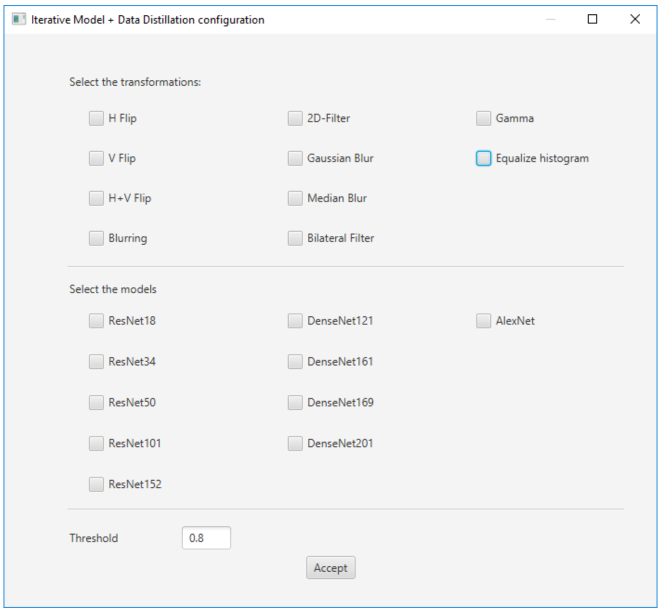


Figure 5. Wizard window of our tool where the users can configure one of the seven processes. In particular, we can see the I.M.D.D. configuration screen where users can select the different transformations and models to use, and also establish the threshold. Similar screens have been developed for the other processes.

4.1 Small annotated datasets

In the first scenario, we analyse the performance of our methods with small, but fully annotated, datasets. In these experiments, we can only employ the no distillation case of our approach, since there are not unlabelled images; but, we can employ different base networks.

In this study, we have compared our approach with AutoML tools that can construct image classification models directly from the datasets of images. Specifically, we have compared five AutoML tools (AutoKeras [28], Devol [13], Frimcla [31], Ludwing [32] and WND-CHARM [33]) with our no distillation approach using the deep learning architectures: Resnet [20], in particular, ResNet34, ResNet50 and ResNet101; and DenseNet [23] with DenseNet121. The datasets employed for the comparison are: Binucleate [33], Cho [7], 2-D Hela [8], Lymphoma [33], Pollen [15] and RNAI [33]. A description including the number of classes and size of these datasets is provided in Table 1.

	Number of classes	Number of images
Binucleate	2	40
Cho	5	340
2D-Hela	10	860
Lymphoma	3	375
Pollen	7	630
RNAI	10	200

Table 1. Description of the small datasets used in the comparison.

Each dataset was split using a 75% for training and a 25% for testing. The result of the comparison of the AutoML tools can be seen in the Table 2. We can notice that there are two tools that stand out: FrImCla, and our no distillation method. This is mainly due to the fact that AutoKeras, Devol and Ludwing employ Neural Architecture Search algorithms [16], a family of techniques that require large corpora of data to be trained; and WND-Charm does not employ deep learning but manually engineered features. On the contrary, FrImCla and our no distillation method employ transfer learning, and this approach works better when dealing with small datasets of images. The main difference between FrImCla and our no distillation method is that FrImCla employs several pre-trained networks as feature extractors and conduct a thorough statistical analysis to select the best model. On the contrary, the no distillation method employs fine-tuning of only one network; and, therefore, it is faster than FrImCla.

If we focus on the different networks employed in the no distillation method, we can notice the No Free Lunch theorem [21]: there is not a single architecture that is the best in all the datasets; but, ResNet50 and ResNet101 seem to excel the others. However, ResNet50 is trained faster ($\sim$ 6 min) than ResNet101 ($\sim$ 12 min). Hence, this is the by-default option provided by our tool; in addition, users can also select the others to search for the best model.

	Binucleate	Cho	Hela	Lymphoma	Pollen	RNAI
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
FrImCla	**1.00**	**0.98**	0.82	0.86	**0.96**	0.69
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
N.D. ResNet34	0.9	0.91	**0.98**	0.8	**0.96**	0.54
N.D. ResNet50	**1.00**	0.91	**0.98**	**0.91**	0.94	0.64
N.D. ResNet101	**1.00**	0.95	0.97	0.8	**0.96**	**0.74**
N.D. DenseNet121	**1.00**	0.92	0.96	0.93	**0.96**	0.62

Table 2. Comparison of the performance of AutoKeras, Devol, FrImCla, Ludwig, WND-CHARM, and our no distillation (N.D.) method using the ResNet34, ResNet50, ResNet101 and DenseNet121 networks for automatically constructing models for six image classification problems. The best results are highlighted in bold face.

4.2 Partially annotated datasets

In the second scenario, we test the processes explained in Section 3.1 with partially annotated datasets. In particular, we have analysed the impact of our method in three different datasets: the plant seedling, the Kvasir, and the ISIC datasets. The Plant Seedlings dataset [18] is a biological dataset that contains approximately 5500 unique plants belonging to 12 species at several growth stages. This dataset comprises annotated RGB images with a physical resolution of roughly 10 pixels per mm. The Kvasir dataset [34] is a medical dataset that contains gastrointestinal diseases images. In particular, this dataset consists of 8000 images with different resolution from 720×576 up to 1920×1072 pixels organised in 8 classes, i.e., approximately 1000 images for each class. The ISIC dataset [19] is a medical dataset that contains melanoma images. In particular, this dataset consists of 1500 images with different resolution organised in 7 classes, i.e., approximately 200 images for each class. The ISIC dataset is a dataset created by the International Skin Imaging Collaboration to facilitate the application of digital skin imaging to help reduce melanoma mortality.

For our study, we have split the datasets into two different sets: a training set with the 75% of images and a testing set with the 25% of the images. With this division, we perform an analysis of the seven processes explained in the Section 3.1: No Distillation (N.D.), Data Distillation (D.D.), Iterative Data Distillation (I.D.D), Model Distillation (M.D.), Iterative Model Distillation (I.M.D.), Model + Data Distillation (M.D.D.) and Iterative Model + Data Distillation (I.M.D.D). For each process, we carry out three different experiments, starting from 25, 50 and 75 annotated images per class and leaving the rest of the training images unlabelled. In addition, for the processes that use the test-time augmentation we have selected five augmentation techniques, namely, horizontal flip, vertical flip, horizontal and vertical flip, blurring, and gamma correction. In the model distillation processes, we have used the architectures ResNet34, ResNet50, ResNet101 and DenseNet121. Finally, for the iterative processes, we have established a threshold value of 0.8.

The result of these experiments can be seen in Table 3. In addition, for each dataset, we have trained a model using the complete training set, and we have obtained the following accuracy values: Plant Seedlings dataset, 0.9616; Kvasir dataset, 0.9340; and ISIC dataset, 0.8724. From these experiments, we can draw several conclusions. First, if we start from a dataset with 25 annotated images per class, we can obtain an improvement of up to a 10% by applying the distillation process. In addition, we can observe that as we increase the number of images initially annotated, the results considerably improve. Also, the processes that work better are the Iterative Model distillation and the Iterative Model + Data distillation. Finally, we can note that the results obtained in the three datasets starting on 75 images annotated per class are very close to the results obtained when training a model with the whole dataset.

	N.D.	D.D.	I.D.D.	M.D.	I.M.D.	M.D.D.	I.M.D.D.
25 per class	0.8348	0.8688	0.8906	0.8833	**0.9167**	0.8819	0.9130
50 per class	0.8905	0.9174	**0.9355**	0.9341	0.9341	0.9261	0.9304
75 per class	0.9137	0.9282	0.9399	0.9413	**0.9514**	0.9399	0.9500
25 per class	0.7975	0.8500	0.8850	0.8800	0.8925	0.8845	**0.8970**
50 per class	0.8430	0.8665	0.8855	0.8880	**0.9025**	0.8880	0.9010
75 per class	0.8730	0.9040	0.9165	0.9050	0.9155	0.9070	**0.9170**
25 per class	0.7473	0.7765	0.7819	0.7845	**0.8377**	0.8005	0.8337
50 per class	0.8111	0.8111	0.8404	0.8218	0.8377	0.8297	**0.8510**
75 per class	0.8404	0.8324	0.8590	0.8537	**0.8776**	0.8430	0.8723

Table 3. Comparison of the performance of the seven different processes (N.D.: No distillation, D.D.: Data distillation, I.D.D.: Iterative Data distillation, M.D.: Model distillation, I.M.D.: Iterative Model distillation, M.D.D.: Model + Data distillation, and, I.M.D.D.: Iterative Model + Data distillation) in the three datasets with 25, 50 and 75 annotated images per class. The first three rows correspond with the results for the Plant Seedlings dataset, the next three rows provide the results for the Kvasir dataset, and the last three rows for the ISIC dataset. The best results are highlighted in bold face.

5 CONCLUSIONS AND FURTHER WORK

The work presented in this paper allows anyone to use deep learning techniques to solve an image classification problem with few resources. In particular, we have presented a general semi-supervised learning process, that combines transfer learning techniques, and data and model distillation techniques. This process allows users to train deep models with small, and partially annotated datasets; that is, datasets with with few annotated images. Also, we have checked that the results obtained with our method when working with partially annotated datasets are very close to train a model with a fully annotated dataset. In addition, we have developed a suite of open-source tools, that allows users to annotate a dataset, and use it for training a model with our method in an easy way.

For further work, we have noticed that the results obtained with the data distillation techniques are dependent on the transformations applied to the images. Therefore, we want to select those transformations automatically depending on the problem we are working with. Moreover, in this project, semi-supervised learning techniques and transfer learning techniques have been applied to solve image classification problems; in the future, we want to transfer these techniques and these processes to other computer vision problems like semantic segmentation. Finally, the processes we have developed use tools in the cloud to train models with all the security issues that this entails; then, we want to improve the security of our deep learning models using encrypted techniques.

ACKNOWLEDGEMENTS

This work was partially supported by Ministerio de Economía y Competitividad [MTM2017-88804-P], and Agencia de Desarrollo Económico de La Rioja [2017-I-IDD-00018], and FPU Grant 16/06903 of the Spanish MEC.

REFERENCES

[1] C. Affonso et al., 'Deep learning for biological image classification', *Expert Systems with Applications*, **85**(1), 114–122, (2017).

[2] S. Akay et al., 'Transfer learning using convolutional neural networks for object classification within x-ray baggage security imagery', in *2016 IEEE International Conference on Image Processing*, ICIP'16, pp. 1057–1061, (2016).

[3] W. Anliang, Y. Xiaolong, and W. Zhijun, 'ImagePy: an open-source, Python-based and platform-independent software package for bioimage analysis', *Bioinformatics*, **34**(18), 3238–3240, (2018).

[4] T. Araújo et al., 'Classification of breast cancer histology images using convolutional neural networks', *PLoS ONE*, **12**(6), (2017).

[5] A. Asperti and C. Mastronardo, 'The effectiveness of data augmentation for detection of gastrointestinal diseases from endoscopical images', in *11th International Joint Conference on Biomedical Engineering Systems and Technologies*, volume 2 of *BIOSTEC 2019*, pp. 199–205. SciTePress, (2018).

[6] D. Avila et al. Project jupyter. `https://jupyter.org/`, 2018.

[7] M. V. Boland, M. K. Markey, and R. F. Murphy, 'Automated recognition of patterns characteristic of subcellular structures in fluorescence microscopy images', *Cytometry: The Journal of the International Society for Analytical Cytology*, **33**(3), 366–375, (1998).

[8] M. V. Boland and R. F. Murphy, 'A neural network classifier capable of recognizing the patterns of all major subcellular structures in fluorescence microscope images of hela cells', *Bioinformatics*, **17**(12), 1213–1223, (2001).

[9] G. Bradski, 'The OpenCV Library', *Dr. Dobb's Journal of Software Tools*, (2000).

[10] C. Bucila, R. Caruana, and A. Niculescu-Mizil, 'Model compression: making big, slow models practical', in *12th International Conference on Knowledge Discovery and Data Mining*, KDD'06, pp. 535–541, (2006).

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

[12] Colaboratory team. Google colaboratory. `https://colab.research.google.com`, 2017.

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

[14] F. de Chaumont et al., 'Icy: an open bioimage informatics platform for extended reproducible research', *Nature Methods*, **9**(7), 690–696, (2012).

[15] A. Duller et al., 'A pollen image database for evaluation of automated identification systems', *Quaternary Newsletter*, 4–9, (1999).

[16] T. Elsken, J. H. Metzen, and F. Hutter, 'Neural architecture search: A survey', *Journal of Machine Learning Research*, **20**, 1–21, (2019).

[17] 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'17, pp. 516–524. Springer International Publishing, (2017).

[18] T. M. Giselsson et al., 'A Public Image Database for Benchmark of Plant Seedling Classification Algorithms', *arXiv preprint*, **abs/1711.05458**, (2017).

[19] D. Gutman et al., 'Skin Lesion Analysis toward Melanoma Detection: A Challenge at the International Symposium on Biomedical Imaging (ISBI) 2016, hosted by the International Skin Imaging Collaboration (ISIC)', in *Conference on Computer Vision and Pattern Recognition*, CVPR'17, (2017).

[20] K. He et al., 'Deep residual learning for image recognition', in *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR'16, pp. 770–778, (2016).

[21] Y.C. Ho and D.L. Pepyne, 'Simple explanation of the no-free-lunch theorem and its implications', *Journal of Optimization Theory and Applications*, **115**(3), 549–570, (Dec 2002).

[22] J. Howard et al. Practical deep learning for coders. `https://course.fast.ai/`, 2019.

[23] G. Huang et al., 'Densely connected convolutional networks', in *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR'17, pp. 4700–4708, (2017).

[24] R. Huang et al., 'Omni-supervised learning: Scaling up to large unlabelled medical datasets', in *Medical Image Computing and Computer Assisted Intervention*, MICCAI'18, pp. 572–580. Springer International Publishing, (2018).

[25] *Automated Machine Learning: Methods, Systems, Challenges*, eds., F. Hutter, L. Kotthoff, and J. Vanschoren, Springer, 2018. In press, available at http://automl.org/book.

[26] A. Inés et al., 'Deepclas4bio: Connecting bioimaging tools with deep learning frameworks for image classification', *Computers in Biology and Medicine*, **108**, 49 – 56, (2019).

[27] J. Irvin et al., 'Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison', in *Thirty-Third AAAI Conference on Artificial Intelligence*, volume 33 of *AAAI'19*, pp. 590–597, (2019).

[28] H. Jin, Q. Song, and X. Hu, 'Auto-keras: An efficient neural architecture search system', in *25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD'19, pp. 1946–1956. ACM, (2019).

[29] N. P. Jouppi et al., 'In-datacenter performance analysis of a tensor processing unit', in *44th Annual International Symposium on Computer Architecture*, ISCA'17, pp. 1–12, (2017).

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, 'ImageNet Classification with Deep Convolutional Neural Networks', in *25th International Conference on Neural Information Processing Systems*, volume 1 of *NIPS'12*, pp. 1097–1105, (2012).

[31] M. García-Domínguez et al. FrImCla: A Framework for Image classification using traditional and Transfer Learning Techniques. `https://github.com/ManuGar/FrImCla`, 2019.

[32] P. Molino, Y. Dudin, and S. S. Miryala, 'Ludwig: a type-based declarative deep learning toolbox', *arXiv preprint*, **abs/1711.05458**, (2019).

[33] N. Orlov et al., 'Wnd-charm: Multi-purpose image classification using compound image transforms', *Pattern recognition letters*, **29**(11), 1684–1693, (2008).

[34] K. Pogorelov et al., 'Kvasir: A multi-class image dataset for computer aided gastrointestinal disease detection', in *8th ACM on Multimedia Systems Conference*, MMSys'17, pp. 164–169. ACM, (2017).

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

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

[37] A. Rosebrock, *Deep Learning for Computer Vision with Python*, PyImageSearch, 2018.

[38] C. T. Rueden et al., 'ImageJ2: ImageJ for the next generation of scientific image data', *BMC Bioinformatics*, **18**, 529, (2017).

[39] D. Sculley et al., 'Machine learning: The high interest credit card of technical debt', in *SE4ML: Software Engineering for Machine Learning*, NIPS'14 Workshop, (2014).

[40] P. Simard et al., 'Tangent prop – a formalism for specifying selected invariances in an adaptive network', in *4th International Conference on Neural Information Processing Systems*, volume 4 of *NIPS'91*, pp. 895–903, (1992).

[41] L. Smith, 'Cyclical learning rates for training neural networks', in *2017 IEEE Winter Conference on Applications of Computer Vision*, WACV'17, pp. 464–472, (2017).

[42] C. Szegedy et al., ‘Rethinking the Inception Architecture for Computer Vision’, in *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR’16, pp. 2818–2826, (2016).

[43] X. Wang et al., ‘ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases’, in *2017 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, CVPR ’17. IEEE Computer Society, (2017).

[44] X. Zhu and A. B. Goldberg, *Introduction to Semi-Supervised Learning*, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, 2009.

[45] B. Zoph et al., ‘Learning Transferable Architectures for Scalable Image Recognition’, in *IEEE Conference on Computer Vision and Pattern Recognition*, CVPR’18, pp. 1–14, (2018).