Computer Methods and Programs in Biomedicine 198 (2021) 105782

Contents lists available at ScienceDirect

Computer Methods and Programs in Biomedicine

journal homepage: www.elsevier.com/locate/cmpb

Biomedical image classification made easier thanks to transfer and semi-supervised learning

A. Inés*, C. Domínguez, J. Heras, E. Mata, V. Pascual

Department of Mathematics and Computer Science of University of La Rioja, Centro Científico Tecnológico Logroño E-26006, La Rioja, Spain

ARTICLE INFO

Article history:
Received 27 February 2020
Accepted 29 September 2020

Keywords:
AutoML
Image classification
Semi-Supervised learning
Transfer-learning
Benchmark

ABSTRACT

Background and objectives: Deep learning techniques are the state-of-the-art approach to solve image classification problems in biomedicine; however, they require the acquisition and annotation of a considerable volume of images. 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 outside the machine-learning community. In this work, we present an Automated Machine Learning (AutoML) method to deal with these problems.

Methods: Our AutoML method combines transfer learning with a new semi-supervised learning procedure to train models when few annotated images are available. In order to facilitate the dissemination of our method, we have implemented it as an open-source tool called ATLASS. Finally, we have evaluated our method with two benchmarks of biomedical image classification datasets.

Results: Our method has been thoroughly tested both with small datasets and partially annotated biomedical datasets; and, it outperforms, both in terms of speed and accuracy, the existing AutoML tools when working with small datasets; and, might improve the accuracy of models up to a 10% when working with partially annotated datasets.

Conclusions: The work presented in this paper allows the use of deep learning techniques to solve an image classification problem with few resources. Namely, it is possible to train deep models with small, and partially annotated datasets of images. In addition, we have proven that our AutoML method outperforms other AutoML tools both in terms of accuracy and speed when working with small 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 [3]. Just to name a few examples in the biomedical field, deep learning has been applied for classifying lung nodules on computed tomography images [4], skin cancer images [5], or breast cancer histology images [3]; and, in general, deep learning techniques provide an effective and efficient computational tool to deal with the growth in biomedical data produced due to the advances of high-throughput technologies [6]. However, building image classification models using deep learning methods might be challenging for researchers outside the machine-learning community. Namely, deep learning techniques require thousands of annotated images, and acquiring and annotating such an amount of images might be difficult [7] and requires specialised knowledge [8]. Moreover, training a deep learning 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; and organising and processing the data in a proper way [9].

The deep learning community has already tackled these challenges. Techniques like *transfer learning* [10] and *data augmentation* [11] have shown that it is feasible to train deep learning models with a limited amount of data and computational resources [10]. In addition, semi-supervised techniques, like data and model distillation [12], take advantage of unlabelled data for training; and, therefore, reduce the burden of annotating images [13]. Moreover, there are several Automated Machine Learning (AutoML) techniques that automatically select the best model to solve a task without user intervention [14]. 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

* Corresponding author.
E-mail address: adines@unirioja.es (A. Inés).

https://doi.org/10.1016/j.cmpb.2020.105782
0169-2607/© 2020 Elsevier B.V. All rights reserved.

experience working with several deep learning libraries and tools; and their use produce, in many cases, *pipeline jungles* [9]. 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. Furthermore, AutoML tools that apply semi-supervised learning has been only developed for structured data [15], and it does not exist, up to the best of our knowledge, an AutoML tool for constructing image classification models that incorporates semi-supervised learning 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 present an automatic method 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 conduct a thorough analysis for our method and show its performance compared with other AutoML tools for constructing image classification models when working with small datasets — to this aim, we employ a standard benchmark for testing biological image classification algorithms [16].
- We show the benefits of our semi-supervised learning method when dealing with partially annotated datasets of biomedical images — we propose a new benchmark for testing semi-supervised learning algorithms in this context.
- Finally, we present ATLASS, a user-friendly and open-source AutoML tool that allows non-expert users to easily employ our method to construct their own image classification models. ATLASS is the first AutoML tool for constructing image classification models that incorporates semi-supervised learning.

The rest of this paper is organised as follows. In the next section, we provide the necessary background to understand the rest of the paper. Subsequently, we present our method 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 AutoML research [14]. AutoML techniques aim to democratise machine learning by simplifying the construction and usage of models for domain experts with a limited machine learning background. Since there are dozens of alternatives for each step of a machine learning pipeline (for instance, feature extraction, model selection or hyperparameter optimisation), AutoML techniques try to find the best configuration for each particular problem. In the context of image classification, AutoML techniques are mainly focused on Neural Architecture Search (NAS); that is, automatically designing neural deep architectures [17]. Even if this approach has outperformed manually designed architectures [18], the adoption of NAS techniques by non-expert users to construct recognition models is far from trivial; mainly, because these techniques are computationally intensive (for instance, NasNet takes 1800 GPU days [18], and AmoebaNet takes 3150 GPU days [19]) and require huge datasets (usually, datasets like ImageNet [20] or CIFAR [21], that contain hundred of thousands, or even millions, annotated images are employed). In addition, NAS techniques require some prior knowledge about typical properties of architectures to simplify the search [17], and also require experience to configure the tools implementing those methods. Due to these reasons, NAS methods and tools are difficult to adopt in the biomedical context, and it is necessary the development of new AutoML methods and tools that work with a limited amount of annotated images and resources.

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 [22,23]. Among those procedures, *transfer learning* stands out when working with small datasets. *Transfer learning* [10] 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 [24,25]. 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 that are common for most images, whilst the later layers 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 layers 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 employed to reduce the amount of resources required 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 [11] 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 to reduce their generalisation error [26]. In addition, data augmentation can also be employed at test time by feeding several transformations of a sample to a trained model, and ensembling the predictions to obtain the final result. This procedure is known as *test-time augmentation* (TTA) [27]. TTA and ensemble methods are the basis of two semi-supervised learning techniques known as data and model distillation. Semi-supervised learning methods have received growing attention in recent years since they provide a mean of using unlabeled data to improve model performance when large-scale annotated data is not available [28–31]. 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 [32].

Data and model distillation are two forms of self-training [13], a special kind of semi-supervised learning technique. In the case of data distillation [12], 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 [33], 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 [12].

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 architectures 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 AutoML method to construct image classification models by combining transfer learning and a new semi-supervised method based on data and model distillation. Moreover, we implement our method in a user-friendly tool that considerably reduces the burden of using transfer-learning and semi-supervised learning methods.

```
Input:
    $(X,Y)$ set of annotated images, and $\bar{X}$ set of unlabelled images.
    Image transformations $\{T_0,T_1,\dots,T_t\}$ where $T_0$ is the identity.
    Learning algorithms $\{A_1,\dots,A_m\}$.
    threshold $\in [0,1]$.

Algorithm:
$(X_{train},Y_{train}),(X_{test},Y_{test})$ = split$(X,Y)$

do
    for $i$ from 1 to $m$:
        $M_i$ = train($A_i$, $(X_{train},Y_{train})$)
    endfor
    for each image $\bar{x}\in\bar{X}$:
        for $j$ from 0 to $t$:
            for $i$ from 1 to $m$:
                $(\bar{y}_{i,j},\bar{p}_{i,j}) = M_i(T_j(\bar{x}))$
            endfor
        endfor
        $(\bar{y},\bar{p})$ = ensemble($\{(\bar{y}_{i,j},\bar{p}_{i,j}\}_{i\in[1,\dots,m],j\in[0,\dots,t]}$)
        if $\bar{p} >$ threshold then:
            $(X_{train},Y_{train}) = (X_{train},Y_{train})\bigcup\{(\bar{x},\bar{y})\}$
            $\bar{X} = \bar{X}\setminus\{\bar{x}\}$
        endif
    endfor
while($\bar{X}\neq\emptyset$ or stop condition reached)
return bestModel($M_1,\dots,M_m,(X_{test},Y_{test})$)
```

Fig. 1. Pseudocode of the workflow of our method.

3. Methods

In this section, we provide a detailed explanation of our AutoML method, and the tool where we have implemented it. We start by presenting the general workflow of our method.

3.1. Workflow

Given a dataset of annotated images (X, Y), where for $(x, y) \in (X, Y)$ x is an image and y is the associated category of x; and a dataset of unlabelled images $\bar{X}$, our method consists of the following steps.

We start by training m models $\{M_1, \dots, M_m\}$ using (X, Y). Now, for each image $\bar{x} \in \bar{X}$ and using t image transformations $T = \{T_1, \dots, T_t\}$, we generate $t+1$ new images $T_0(\bar{x}), T_1(\bar{x}), \dots, T_t(\bar{x})$, where T_0 is the identity. Subsequently, we apply each model M_i to each $T_j(\bar{x})$ and obtain as a result $(\bar{y}_{i,j}, \bar{p}_{i,j})$ where $\bar{y}_{i,j}$ is the class predicted by M_i for $T_j(\bar{x})$, and $\bar{p}_{i,j}$ is its associated confidence. After that, we ensemble the predictions $\{(\bar{y}_{i,j}, \bar{p}_{i,j})\}_{i\in[1,\dots,m],j\in[0,\dots,t]}$ using the weighted majority voting scheme, where the weights are the confidence score of each prediction, and obtain $(\bar{y}, \bar{p})$. Finally, if $\bar{p}$ is over a fixed threshold, then we add $\{(\bar{x}, \bar{y})\}$ to (X, Y) and remove $\bar{x}$ from $\bar{X}$; and, the process is iterated — the pseudocode for this workflow is provided in Fig. 1. The process 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. The result of this process is a model; in particular, the model with the best performance with respect to an independent labelled test set, in the last iteration. A graphical representation of one of the iterations of our workflow is depicted in Fig. 2.

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 execute it. Namely, starting from the general process previously explained, seven particular cases can be defined.

No Distillation (N.D.): This is the most basic case and occurs when there are not unlabelled images in the original dataset, that is, $\bar{X} = \emptyset$. Then, the process is reduced to train and select the best model model with respect to an independent test set.

Data Distillation without thresholding (D.D.): This case appears when, instead of having a set of base models, we only train one model ($M = \{M_1\}$), 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 [12].

Iterative Data Distillation (I.D.D.): As in the previous case, a single model is trained ($M = \{M_1\}$); but 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; but, we do not perform test-time augmentation of the unlabelled images, that is, ($T = \{T_0\}$), and only ensemble the model predictions of the given image. 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 [33].

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 ($T = \{T_0\}$). 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.

Since our final aim is the development of an AutoML method, a key aspect that remains to be explained is how to train the best possible model. To this aim, we employ transfer learning as explained in the following subsection.

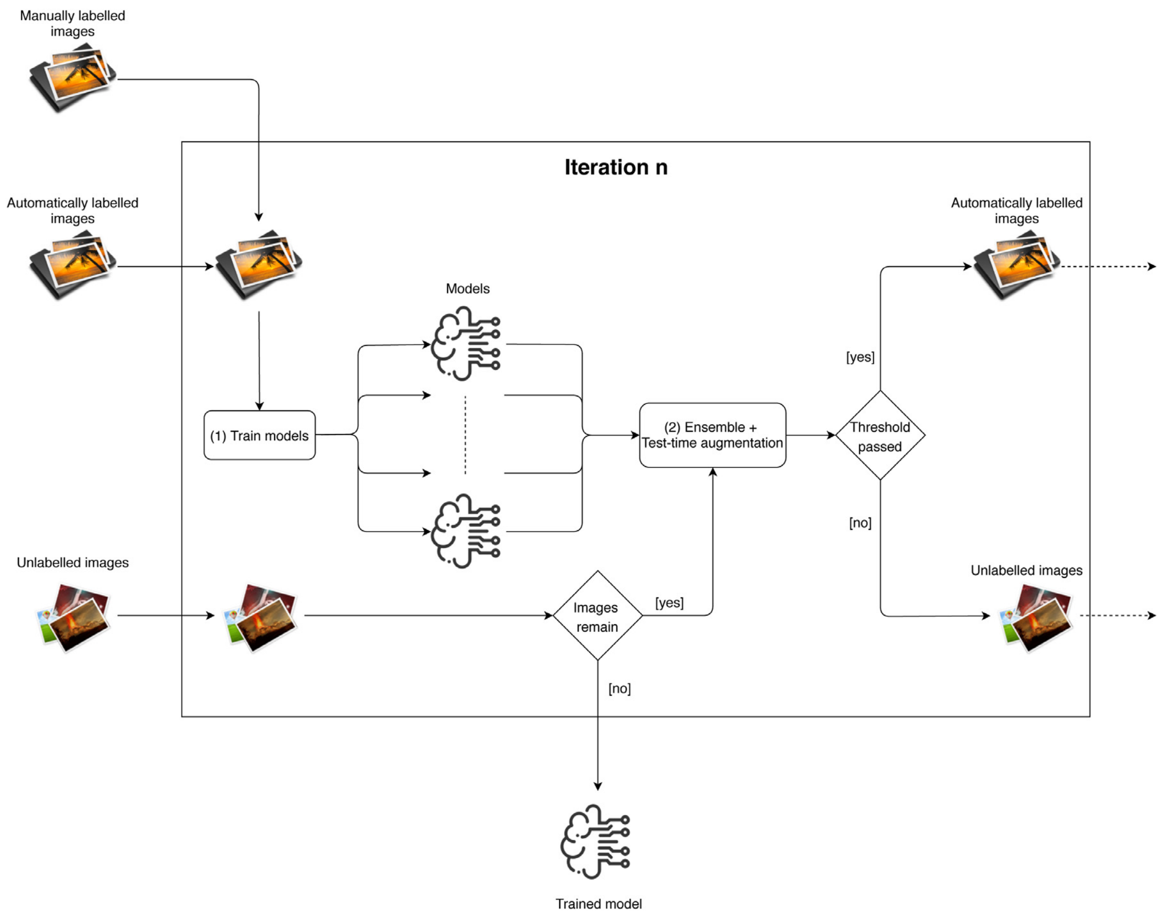


Fig. 2. Workflow of an iteration of our semi-supervised learning method. **Inputs of the iteration**: set of manually labelled images, set of automatically labelled images and set of unlabelled images. **Outputs of the iteration**: set of automatically labelled images and set of unlabelled images. **Output of the process**: Trained model.

3.2. Training the models

In our method, we train each model 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 model (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 to conduct 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 [34]. 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 $1e^{-3}$.

As we will show in Section 4, using this training procedure and our semi-supervised method, we can produce accurate models; however, implementing and using these techniques might be challenging for non-expert users. Therefore, we have developed a tool that facilitates the construction of image classification models using our techniques without writing a single line of code.

3.3. Suite of tools

We have developed an open-source tool, called ATLASS, that implements our method, and guides the user in all the stages of the construction of an image classification model; namely, it helps to annotate the images, train a model, test it, and finally use it. ATLASS is available at https://github.com/adines/ATLASS.

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. Then, the application automatically loads the images of the dataset and shows to the users the corresponding information. 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 and compared using the no distillation procedure. If the dataset contains unlabelled images, the wizard asks for the distillation methods that will be applied. If the users select a data distillation method, they can select the augmentation techniques (the complete list of augmentations is in the project webpage); 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. The output of the wizard is a zip file, containing the organised dataset, and a Jupyter notebook with the necessary instructions to be executed.

Jupyter notebooks are interactive documents that can include executable code together with explanations [35]. 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 [36] 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 [37] or other cloud environments.

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 [38], 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 [39], Icy [40] or ImagePy [41].

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

4. Results

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.

4.1. AutoML for 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 method, since there are not unlabelled images; but, we can search the best base network among several alternatives.

In this study, we have compared our approach with other AutoML tools that can construct image classification models directly from the datasets of images. Specifically, we have compared four AutoML tools (AutoKeras [42], Devol [43], Ludwig [44] and WndCharm [45]) with our no distillation approach using the deep learning architectures: Resnet [46], in particular, ResNet34, ResNet50 and ResNet101; and DenseNet [47] with DenseNet121. The datasets employed for the comparison are the 11 datasets presented in [16], and described in Table 1.

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 Table 2 and Fig. 3. We can notice that our method stands

Table 1
Description of the datasets employed in our experiments. The first 11 datasets were studied using only the No Distillation case, and the rest using the different variants of our semi-supervised learning method.

Dataset	Number of Images	Number of Classes
Binucleate	40	2
C. Elegans	252	4
Cho	340	5
2D-Hela	860	10
Liver aging	850	4
Liver gender (AL)	522	2
Liver gender (CR)	256	2
Lymphoma	375	3
Pollen	630	7
RNAI	200	10
Terminal Bulb aging	970	7
Blindness [48]	3662	5
Chest X Ray [49]	2355	2
Fungi [50]	1204	4
HAM 10,000 [51]	10015	7
ISIC [52]	1500	7
Kvasir [53]	8000	8
Open Sprayer [54]	6697	2
Plants [55]	5500	12
Retinal OCT [49]	84484	4
Tobacco [56]	3492	10

out for all the analysed metrics. We have used Wilcoxon signed-rank tests to compare the accuracy results obtained by our tool with respect to AutoKeras, Devol, Ludwig, and WndCharm AutoML tools. Significant differences with large effect sizes (with z = -2.80, p = 0.005, r = -0.60; z = -2.93, p = 0.003, r = -0.63; z = -2.93, p = 0.003, r = -0.63; and z = -2,67, p = 0.008, r = -0.57; respectively) were obtained in all the cases. Similar results are achieved for other metrics, see Fig. 3 and Appendix A. This is mainly due to the fact that AutoKeras, Devol and Ludwig employ Neural Architecture Search algorithms [17], a family of techniques that require large corpora of data to be trained; and WndCharm employs manually engineered features, that are fixed for all the dataset independently of the concrete problem. In contrast, our method is adapted to each particular problem, and applies transfer learning to reuse the knowledge learned from big datasets. This approach works better than the others when dealing with small datasets since most of the features learned from a big dataset can be reused for the smaller ones; this is especially true for low-level features, like lines or edges, that are common for all kinds of images [57].

In addition, our AutoML method is faster than the other tools (see the last column of Table 2). This is due to the fact that the neural architecture search methods of AutoKeras, Ludwig, and Devol are time-consuming, and WndCharm runs on the CPU instead of the GPU. All the experiments were conducted in the environment provided by Google Colab (2-core Xeon 2.3 GHz, 13 TB Ram, and a Tesla P100 GPU).

4.2. Partially annotated datasets

In the second scenario, we test our semi-supervised method in the biomedical setting. Evaluation of semi-supervised learning methods is done mostly on the MNIST [58], CIFAR [21] or SVHN [59] datasets; however, images in these datasets are usually small (less than 50×50 in resolution) and come from natural settings. Therefore, these datasets are not suitable for testing semi-supervised learning methods in the biomedical context. In this work, we propose a benchmark of 10 partially annotated biomedical datasets, described in Table 1, and evaluate our method using such a benchmark.

For our study, we have split the datasets of the benchmark into two different sets: a training set with the 75% of images and a

Table 2
Comparison of the performance of AutoKeras, Devol, Ludwig, WndCharm, and our AutoML method for automatically constructing models for 11 image classification problems using the accuracy metric. The last column provides the mean time required for training the models with each tool. The best results are highlighted in bold face.

	Binu.	CEle.	Cho	Hela	Liver aging	Liver (AL)	Liver (CR)	Lymp.	Pollen	RNAI	Term.	Mean (S.D.)	Mean Time (min)
AutoKeras	0.63	0.66	0.91	0.78	0.91	0.69	**1.00**	0.73	0.94	0.52	0.49	0.75 (0.17)	30
Devol	0.73	0.42	0.61	0.38	0.33	0.81	0.71	0.56	0.56	0.20	0.33	0.51 (0.19)	16
Ludwig	0.54	0.48	0.64	0.51	0.33	0.65	0.90	0.57	0.58	0	0.50	0.52 (0.22)	31
WndCharm	**1.00**	0.60	0.95	0.85	0.92	**1.00**	0.97	0.79	0.96	0.68	0.50	0.84 (0.17)	53
Ours	**1.00**	**1.00**	**0.97**	**0.99**	**0.98**	**1.00**	**1.00**	**0.95**	**0.97**	**0.77**	**0.73**	0.94 (0.10)	16

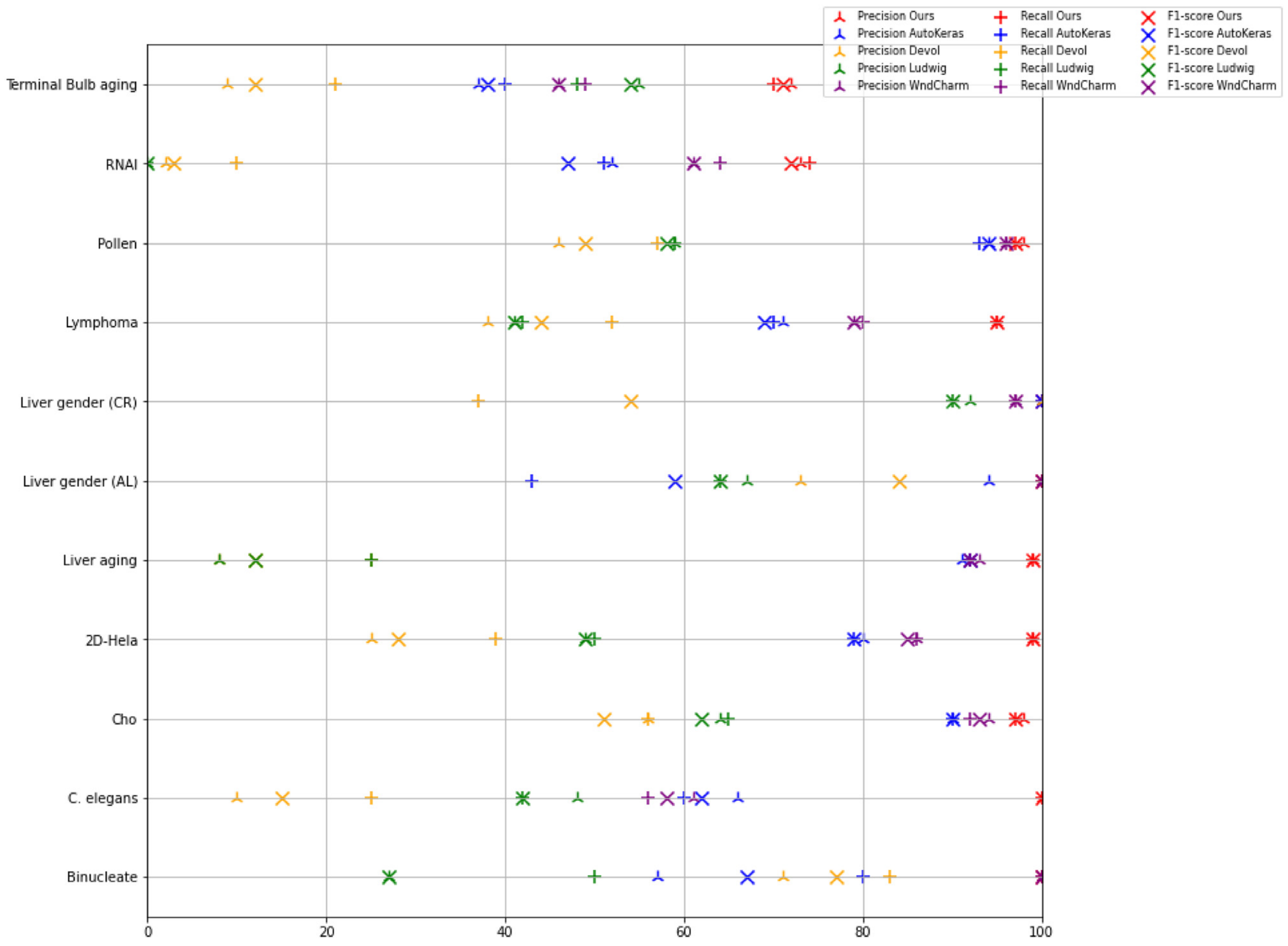


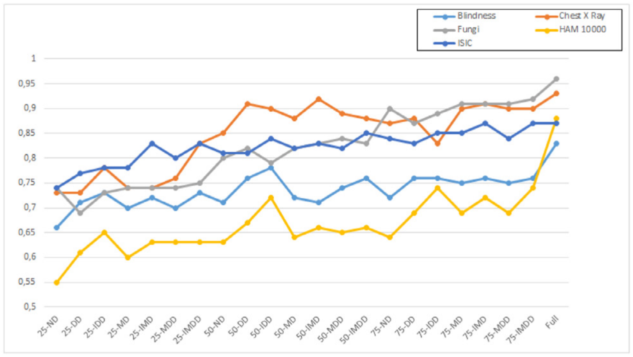
Fig. 3. Comparison of the performance of AutoKeras, Devol, Ludwig, WndCharm, and our AutoML method for automatically constructing models for 11 image classification problems using precision, recall and F1-score.

testing set with the 25% of the images. With this division, we perform an analysis of the seven processes explained in 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 of the training set per class and considering the rest of the training images as unlabelled, we apply the seven processes; and, additionally, we apply the N.D. process to the whole dataset. Furthermore, for the processes that use 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 and Fig. 4. From these experiments, we can draw several conclusions. First of all, we can see that the improvements achieved using our method range from 3% in the worst case up to 10% in the best case. Wilcoxon signed-rank tests found those as significant improvements ($p < 0.05$) with large effect sizes ($r > 0.5$) in all comparisons, except in three of them. We can also notice that the iterative versions of our processes produce better results than their non-iterative counterparts in all the cases but one. In addition, if we compare our semi-supervised processes using 25 and 50 images with respect to the N.D. process using 50 images and 75 images respectively, a 60% of the times our processes obtain better results than the N.D. method. This percentage increases to 70% when we consider only the iterative versions; therefore, even if they are a bit slower, it is preferable to apply one of the iterative process. All these improvements are achieved thanks to the use of additional images that are auto-

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, I.M.D.D.: Iterative Model + Data distillation) in 10 datasets with 25, 50 and 75 annotated images per class. The "Full" column indicates the accuracy of the N.D. method applied to the whole dataset. The best results are highlighted in bold face. Wilcoxon signed-rank tests for comparing the results obtained by the N.D process with respect to the other six processes in each block of 25, 50 and 75 annotated images per class are also included.

	25 per class							50 per class							75 per class							Full
	N.D.	D.D.	I.D.D.	M.D.	I.M.D.	M.D.D.	I.M.D.D.	N.D.	D.D.	I.D.D.	M.D.	I.M.D.	M.D.D.	I.M.D.D.	N.D.	D.D.	I.D.D.	M.D.	I.M.D.	M.D.D.	I.M.D.D.	
Blindness	0.66	0.71	**0.73**	0.70	0.72	0.70	0.73	0.71	0.76	**0.78**	0.72	0.71	0.74	0.76	0.72	0.76	0.76	0.75	**0.76**	0.75	0.76	0.83
Chest X Ray	0.73	0.73	0.78	0.74	0.74	0.76	**0.83**	0.85	0.91	0.90	0.88	**0.92**	0.89	0.88	0.87	0.88	0.83	0.90	**0.91**	0.90	0.90	0.93
Fungi	0.74	0.69	0.73	0.74	0.74	0.74	**0.75**	0.80	0.82	0.79	0.82	0.83	**0.84**	0.83	0.90	0.87	0.89	0.91	0.91	0.91	**0.92**	0.96
HAM10000	0.55	0.61	**0.65**	0.60	0.63	0.63	0.63	0.63	0.67	**0.72**	0.64	0.66	0.65	0.66	0.64	0.69	**0.74**	0.69	0.72	0.69	0.74	0.88
ISIC	0.74	0.77	0.78	0.78	**0.83**	0.80	0.83	0.81	0.81	0.84	0.82	0.83	0.82	**0.85**	0.84	0.83	0.85	0.85	**0.87**	0.84	0.87	0.87
Kvasir	0.79	0.85	0.88	0.88	0.89	0.88	**0.89**	0.84	0.86	0.88	0.88	**0.90**	0.88	0.90	0.87	0.90	0.91	0.90	0.91	0.90	**0.91**	0.93
Open Sprayer	0.84	0.90	0.91	0.83	0.84	0.85	**0.93**	0.87	0.86	0.90	0.90	0.91	0.91	**0.92**	0.90	0.92	0.94	0.94	0.94	0.94	**0.94**	0.97
Plants	0.83	0.86	0.89	0.88	**0.91**	0.88	0.91	0.89	0.91	**0.93**	0.93	0.93	0.92	0.93	0.91	0.92	0.93	0.94	**0.95**	0.93	0.95	0.96
Retinal OCT	0.90	0.90	0.86	0.93	**0.96**	0.93	0.94	0.93	0.95	0.95	0.96	**0.98**	0.97	0.97	0.94	0.97	0.93	0.98	**0.99**	0.98	0.98	0.99
Tobacco	0.66	0.69	0.70	0.74	**0.76**	0.72	0.74	0.72	0.75	0.77	0.77	**0.80**	0.79	0.76	0.78	0.81	**0.84**	0.81	0.81	0.81	0.79	0.86
Mean	0.74	0.77	0.79	0.78	0.80	0.79	0.82	0.81	0.83	0.85	0.83	0.85	0.84	0.85	0.84	0.86	0.86	0.87	0.88	0.87	0.88	0.92
S.D.	0.10	0.10	0.09	0.10	0.10	0.10	0.10	0.09	0.09	0.08	0.10	0.10	0.10	0.10	0.10	0.08	0.07	0.09	0.09	0.9	0.08	0.05
z (Wilcoxon)		-1.91	-2.40	-2.49	-2.53	-2.67	-2.81		-2.57	-2.71	-2.82	-2.67	-2.84	-2.83		-1.96	-1.75	-2.84	-2.84	-2.69	-2.84	
p		0.056	0.016	0.013	0.011	0.008	0.006		0.010	0.007	0.005	0.008	0.004	0.005		0.051	0.080	0.004	0.004	0.007	0.004	
r		-0.43	-0.54	-0.56	-0.57	-0.60	-0.63		-0.57	-0.61	-0.63	-0.60	-0.64	-0.63		-0.44	-0.39	-0.64	-0.64	-0.60	-0.64	



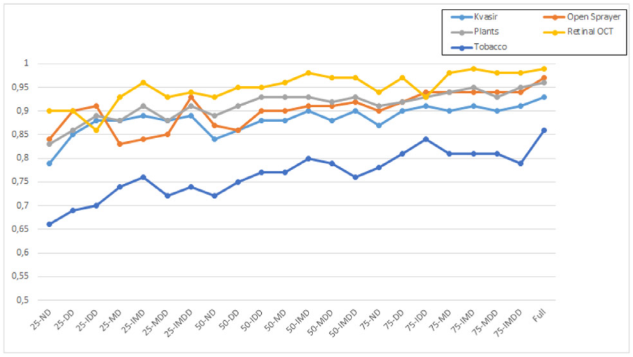


Fig. 4. Comparison of the performance of the seven different processes (N.D.: No distillation, D.D.: Data distillation, I.D.D.: Iterative Data distillation, M.D.: Model distillation, I.M.D.: Iterative Model distillation, M.D.D.: Model + Data distillation, I.M.D.D.: Iterative Model + Data distillation) in 10 datasets with 25, 50 and 75 annotated images per class.

matically annotated by our method; this helps the models to generalise.

Finally, we can observe that as we increase the number of images initially annotated, the results considerably improve. Moreover, in some cases, it is possible to get results close to those obtained using the N.D. process applied to the whole dataset by using just a small part of the dataset (see the Plant, Retinal OCT and ISIC datasets in Table 3).

5. Conclusions and further work

The work presented in this paper allows the use of deep learning techniques to solve an image classification problem with few resources. In particular, we have presented a general AutoML method that combines transfer and semi-supervised learning techniques. This method allows users to train deep models with small, and partially annotated datasets of images. In addition, we have proven that our AutoML method outperforms other AutoML tools both in terms of accuracy and speed when working with small datasets. Furthermore, our semi-supervised learning method improves the accuracy of models up to a 10% when working with partially annotated datasets. Finally, we have developed an open-source tool, that allows users to annotate a dataset, and use it for training a model with our method in an easy way. Altogether, our approach that combines transfer learning and semi-supervised learning simplify the construction of fairly good image classification models in the biomedical context when working with small, or partially annotated datasets.

We have noticed that the results obtained with the data distillation techniques are dependent on the transformations applied to the images; therefore, for further work, 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.

Declaration of Competing Interest

None declared.

Acknowlgedgments

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

Appendix A. Statistical comparison with AutoML tools

Table 4
Comparison of the performance of AutoKeras, Devol, Ludwig, WndCharm, and our AutoML method for automatically constructing models for 11 image classification problems using the precision metric. The best results are highlighted in bold face. Wilcoxon signed-rank tests for comparing the results obtained by our tool with respect to AutoKeras, Devol, Ludwig, and WndCharm AutoML tools are also included.

	Binu.	CEle.	Cho	Hela	Liver aging	Liver (AL)	Liver (CR)	Lymp.	Pollen	RNAI	Term.	Mean (S.D.)	*z* Wilcoxon	*p*	*r*
AutoKeras	0.57	0.66	0.90	0.80	0.91	0.94	**1.00**	0.71	0.94	0.52	0.37	0.76 (0.21)	-2.80	0.005	-0.60
Devol	0.71	0.10	0.56	0.25	0.08	0.73	**1.00**	0.38	0.46	0.02	0.09	0.4 (0.32)	-2.80	0.005	-0.6
Ludwig	0.27	0.48	0.64	0.49	0.08	0.67	0.92	0.41	0.59	0	0.55	0.46 (0.27)	-2.93	0.003	-0.63
wndcharm	**1.00**	0.61	0.94	0.86	0.93	**1.00**	0.97	0.79	0.96	0.61	0.46	0.83 (0.19)	-2.67	0.008	-0.57
Ours	**1.00**	**1.00**	**0.98**	**0.99**	**0.99**	**1.00**	**1.00**	**0.94**	**0.98**	**0.73**	**0.72**	0.94 (0.11)			

Table 5
Comparison of the performance of AutoKeras, Devol, Ludwig, WndCharm, and our AutoML method for automatically constructing models for 11 image classification problems using the recall metric. The best results are highlighted in bold face. Wilcoxon signed-rank tests for comparing the results obtained by our tool with respect to AutoKeras, Devol, Ludwig, and WndCharm AutoML tools are also included.

	Binu.	CEle.	Cho	Hela	Liver aging	Liver (AL)	Liver (CR)	Lymp.	Pollen	RNAI	Term.	Mean (S.D.)	*z* Wilcoxon	*p*	*r*
AutoKeras	0.80	0.60	0.90	0.79	0.92	0.43	**1.00**	0.70	0.93	0.51	0.40	0.73 (0.21)	-2.81	0.005	-0.60
Devol	0.83	0.25	0.56	0.39	0.25	**1.00**	0.37	0.52	0.57	0.10	0.21	0.46 (0.27)	-2.80	0.005	-0.6
Ludwig	0.50	0.42	0.65	0.50	0.25	0.64	0.90	0.42	0.59	0	0.48	0.49 (0.23)	-2.94	0.003	-0.63
wndcharm	**1.00**	0.56	0.92	0.86	0.92	**1.00**	0.97	0.80	0.96	0.64	0.49	0.83 (0.18)	-2.67	0.008	-0.57
Ours	**1.00**	**1.00**	**0.97**	**0.99**	**0.99**	**1.00**	**1.00**	**0.95**	**0.97**	**0.74**	**0.70**	0.94 (0.11)			

Table 6
Comparison of the performance of AutoKeras, Devol, Ludwig, WndCharm, and our AutoML method for automatically constructing models for 11 image classification problems using the F1-score metric. The best results are highlighted in bold face. Wilcoxon signed-rank tests for comparing the results obtained by our tool with respect to AutoKeras, Devol, Ludwig, and WndCharm AutoML tools are also included.

	Binu.	CEle.	Cho	Hela	Liver aging	Liver (AL)	Liver (CR)	Lymp.	Pollen	RNAI	Term.	Mean (S.D.)	*z* Wilcoxon	*p*	*r*
AutoKeras	0.67	0.62	0.90	0.79	0.92	0.59	**1.00**	0.69	0.94	0.47	0.38	0.72 (0.2)	-2.81	0.005	-0.60
Devol	0.77	0.15	0.51	0.28	0.12	0.84	0.54	0.44	0.49	0.03	0.12	0.39 (0.27)	-2.94	0.003	-0.63
Ludwig	0.27	0.42	0.62	0.49	0.12	0.64	0.90	0.41	0.58	0	0.54	0.45 (0.25)	-2.93	0.003	-0.63
wndcharm	**1.00**	0.58	0.93	0.85	0.92	**1.00**	0.97	0.79	0.96	0.61	0.46	0.82 (0.19)	-2.67	0.008	-0.57
Ours	**1.00**	**1.00**	**0.97**	**0.99**	**0.99**	**1.00**	**1.00**	**0.95**	**0.97**	**0.72**	**0.71**	0.94 (0.11)			

References

[1] C. Affonso, A.L.D. Rossi, F.H.A. Vieira, et al., Deep learning for biological image classification, Expert. Syst. Appl. 85 (1) (2017) 114–122.
[2] S. Akçay, M.E. Kundegorski, M. Devereux, 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, in: ICIP'16, 2016, pp. 1057–1061.
[3] T. Araújo, G. Aresta, E. Castro, et al., Classification of breast cancer histology images using convolutional neural networks, PLoS ONE 12 (6) (2017).
[4] K.L. Hua, C.H. Hsu, S.C. Hidayati, W.H. Cheng, Y.J. Chen, Computer-aided classification of lung nodules on computed tomography images via deep learning technique, Onco Targets Ther. 8 (2015) 2015–2022, doi:10.2147/OTT.S80733.
[5] A. Esteva, B. Kuprel, R.A. Novoa, J. Ko, S.M. Swetter, H.M. Blau, S. Thrun, Dermatologist-level classification of skin cancer with deep neural networks, Nature 542 (2017) 115–118, doi:10.1038/nature21056.
[6] C. Cao, F. Liu, H. Tan, et al., Deep learning and its applications in biomedicine, Enomics Proteom. Bioinform. 16 (1) (2018) 17–32.
[7] A. Asperti, 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, in: BIOSTEC 2019, 2, SciTePress, 2018, pp. 199–205.
[8] J. Irvin, P. Rajpurkar, M. Ko, et al., Chexpert: a large chest radiograph dataset with uncertainty labels and expert comparison, in: Thirty-Third AAAI Conference on Artificial Intelligence, in: AAAI'19, 33, 2019, pp. 590–597.
[9] D. Sculley, G. Holt, D. Golovin, et al., Machine learning: the high interest credit card of technical debt, SE4ML: Software Engineering for Machine Learning, NIPS'14 Workshop, 2014.
[10] A.S. Razavian, H. Azizpour, J. Sullivan, et al., CNN features off-the-shelf: An astounding baseline for recognition, in: IEEE Conference on Computer Vision and Pattern Recognition Workshops, in: CVPRW'14, IEEE, 2014, pp. 512–519.
[11] P. Simard, B. Victorri, Y. LeCun, et al., Tangent prop – a formalism for specifying selected invariances in an adaptive network, in: 4th International Conference on Neural Information Processing Systems, in: NIPS'91, 4, 1992, pp. 895–903.
[12] R. Huang, J.A. Noble, A.I.L. Namburete, Omni-supervised learning: Scaling up to large unlabelled medical datasets, in: Medical Image Computing and Computer Assisted Intervention, MICCAI'18, Springer International Publishing, 2018, pp. 572–580.
[13] X. Zhu, A.B. Goldberg, Introduction to Semi-Supervised Learning, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, 2009.
[14] Automated Machine Learning: Methods, Systems, Challenges, F. Hutter, L. Kotthoff, J. Vanschoren (Eds.), Springer, 2018. In press, available at http://automl.org/book.
[15] Y.-F. Li, H. Wang, T. Wei, et al., Towards automated semi-supervised learning, in: AAAI Conference on Artificial Intelligence, in: AAAI-19, 2019, pp. 4237–4244.
[16] L. Shamir, N. Orlov, D.M. Eckley, et al., Iicbu 2008: a proposed benchmark suite for biological image analysis, Med. Biol. Eng. Comput. 46 (9) (2008) 943–947.
[17] T. Elsken, J.H. Metzen, F. Hutter, Neural architecture search: a survey, J. Mach. Learn. Res. 20 (2019) 1–21.
[18] B. Zoph, V. Vasudevan, J. Shlens, et al., Learning transferable architectures for scalable image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, in: CVPR'18, 2018, pp. 1–14.
[19] E. Real, A. Aggarwal, Y. Huang, et al., Regularized evolution for image classifier architecture search, in: Thirty-Third AAAI Conference on Artificial Intelligence, in: AAAI'19, 33, 2019.
[20] J. Deng, et al., ImageNet: a large-scale hierarchical image database, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR09), 2009.
[21] A. Krizhevsky, Learning multiple layers of features from tiny images, 2009. (https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf)
[22] J. Howard, R. Thomas, Practical deep learning for coders, 2019. (https://course.fast.ai/)
[23] A. Rosebrock, Deep Learning for Computer Vision with Python, PyImageSearch, 2018.
[24] S. Christodoulidis, M. Anthimopoulos, L. Ebner, et al., Multisource transfer learning with convolutional neural networks for lung pattern analysis, IEEE J. Biomed. Health Inform. 21 (1) (2017) 76–84.
[25] M. Ghafoorian, A. Mehrtash, T. Kapur, et al., Transfer learning for domain adaptation in MRI: application in brain lesion segmentation, in: Medical Image Computing and Computer-Assisted Intervention, in: MICCAI'17, Springer International Publishing, 2017, pp. 516–524.
[26] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: 25th International Conference on Neural Information Processing Systems, in: NIPS'12, 1, 2012, pp. 1097–1105.

[27] C. Szegedy, V. Vanhoucke, S. Ioffe, et al., Rethinking the inception architecture for computer vision, in: IEEE Conference on Computer Vision and Pattern Recognition, in: CVPR'16, 2016, pp. 2818–2826.
[28] S. Laine, T. Aila, Temporal ensembling for semi-supervised learning, in: 5th International Conference on Learning Representations, ICLR'17, 2017, pp. 1–13.
[29] D. Berthelot, et al., Mixmatch: A Holistic Approach to Semi-supervised Learning, in: Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 5049–5059.
[30] A. Tarvainen, H. Valpola, Mean teachers are better role models: weight-averaged consistency targets improve semi-supervised deep learning results, in: Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 1195–1204.
[31] K. Sohn, et al., Fixmatch: simplifying semi-supervised learning with consistency and confidence, arXiv preprint abs/2001.07685 (2020).
[32] X. Wang, Y. Peng, L. Lu, 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, in: CVPR '17, IEEE Computer Society, 2017.
[33] C. Bucila, R. Caruana, A. Niculescu-Mizil, Model compression: making big, slow models practical, in: 12th International Conference on Knowledge Discovery and Data Mining, in: KDD'06, 2006, pp. 535–541.
[34] L. Smith, Cyclical learning rates for training neural networks, in: IEEE Winter Conference on Applications of Computer Vision, in: WACV'17, 2017, pp. 464–472, doi:10.1109/WACV.2017.58.
[35] T. Kluyver, B. Ragan-Kelley, F. Pérez, et al., Jupyter notebooks a publishing format for reproducible computational workflows, in: 20th International Conference on Electronic Publishing, in: ICEP'16, 2016, pp. 87–90.
[36] G. Bradski, The opencv library, Dr. Dobb's J. Softwar. Tools (2000).
[37] Colaboratory team, Google colaboratory, 2017, (https://colab.research.google.com).
[38] A. Inés, C. Domínguez, J. Heras, et al., Deepclas4bio: connecting bioimaging tools with deep learning frameworks for image classification, Comput. Biol. Med. 108 (2019) 49–56, doi:10.1016/j.compbiomed.2019.03.026.
[39] C.T. Rueden, J. Schindelin, M.C. Hiner, et al., ImageJ2: ImageJ for the next generation of scientific image data, BMC Bioinform. 18 (2017) 529, doi:10.1186/s12859-017-1934-z.
[40] F. de Chaumont, S. Dallongeville, N. Chenouard, et al., Icy: an open bioimage informatics platform for extended reproducible research, Nat. Methods 9 (7) (2012) 690–696, doi:10.1038/nmeth.2075.
[41] W. Anliang, Y. Xiaolong, W. Zhijun, Imagepy: an open-source, python-based and platform-independent software package for bioimage analysis, Bioinformatics 34 (18) (2018) 3238–3240, doi:10.1093/bioinformatics/bty313.
[42] H. Jin, Q. Song, X. Hu, Auto-keras: an efficient neural architecture search system, in: 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, in: KDD'19, ACM, 2019, pp. 1946–1956.
[43] J. Davison, DEvol - deep neural network evolution, 2018. (https://github.com/joeddav/devol)
[44] P. Molino, Y. Dudin, S.S. Miryala, Ludwig: a type-based declarative deep learning toolbox, arXiv preprint (2019). abs/1711.05458
[45] N. Orlov, L. Shamir, T. Macura, et al., Wnd-charm: multi-purpose image classification using compound image transforms, Pattern Recognit. Lett. 29 (11) (2008) 1684–1693.
[46] K. He, X. Zhang, S. Ren, et al., Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, in: CVPR'16, 2016, pp. 770–778.
[47] G. Huang, Z. Liu, L. van der Maaten, et al., Densely connected convolutional networks, in: IEEE Conference on Computer Vision and Pattern Recognition, in: CVPR'17, 2017, pp. 4700–4708.
[48] Kaggle, Aptos 2019 blindness detection, 2019, (83).
[49] D.S. Kermany, M. Goldbaum, W. Cai, et al., Identifying medical diagnoses and treatable diseases by image-based deep learning, Cell 172 (5) (2018) 1122–1131.e9, doi:10.1016/j.cell.2018.02.010.
[50] M. Arredondo-Santoyo, C. Domínguez, J. Heras, et al., Automatic characterisation of dye decolourisation in fungal strains using expert, traditional, and deep features, Soft Comput. 23 (2019) 12799–12812, doi:10.1007/s00500-019-03832-8.
[51] P. Tschandl, C. Rosendahl, H. Kittler, The HAM10000 dataset: a large collection of multi-Source dermatoscopic images of common pigmented skin lesions, Sci. Data 5 (2018), doi:10.1038/sdata.2018.161.
[52] N.C.F. Codella, D. Gutman, M.E. Celebi, et al., Skin lesion analysis toward melanoma detection: a challenge at the international symposium on biomedical imaging (ISBI) 2016, in: Conference on Computer Vision and Pattern Recognition, in: CVPR'17, 2017.
[53] K. Pogorelov, K.R. Randel, C. Griwodz, et al., Kvasir: A multi-class image dataset for computer aided gastrointestinal disease detection, in: 8th ACM on Multimedia Systems Conference, in: MMSys'17, ACM, 2017, pp. 164–169, doi:10.1145/3083187.3083212.
[54] Kaggle, Open sprayer images, 2019. (https://www.kaggle.com/gavinarmstrong/open-sprayer-images)
[55] T.M. Giselsson, R.N. Jorgensen, P.K. Jensen, et al., A public image database for benchmark of plant seedling classification algorithms, arXiv preprint abs/1711.05458 (2017).
[56] J. Kumar, P. Ye, D. Doermann, Structural similarity for document image classification and retrieval, Pattern Recognition Letters 43 (2014) 119–126, doi:10.1016/j.patrec.2013.10.030. ICPR2012 Awarded Papers
[57] L. Schubert, M. Petrov, S. Carter, et al., Openai microscope, 2020, (https://openai.com/blog/microscope/).
[58] Y. LeCun, C. Cortes, The MNIST dataset ofhandwritten digits, 1999.
[59] A.C. Y. Netzer T. Wang, et al., Reading digits in natural images with unsupervised feature learning, NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011.