Universidad Nacional de Educación a Distancia

Master Universitario en I.A. Avanzada: Fundamentos, Métodos y Aplicaciones

Escuela Técnica Superior de Ingeniería Informática

Improving classification of pollen grain images of the POLEN23E dataset deep learning

Author:
Victor Sevillano Plaza

Supervisor:
Dr. José Luis Aznarte Mellado

June 13, 2019

To my mother, who always supported me.

Abstract

In palynology, the visual classification of pollen grains from different species is a hard task which is usually tackled by human operators using microscopes. Its complete automatization would save a high quantity of resources and provide valuable improvements especially for allergy-related information systems, but also for other application fields as paleoclimate reconstruction, quality control of honey-based products, collection of evidences in criminal investigations or fabric dating and tracking. This paper presents three state-of-the-art deep learning classification methods applied to the recently published POLEN23E image dataset. The three methods make use of convolutional neural networks: the first one is strictly based on the idea of transfer learning, the second one is based on feature extraction and the third one represents a hybrid approach, combining transfer learning and feature extraction. The results from the three methods are indeed very good, reaching over 99% correct classification rates in the training set of images and over 96% in images not previously seen by the models where other authors reported around 70%.

Acknowledgments

I would like to acknowledge everyone who played a role in my academic accomplishments. First, my parents, who supported me with love and understanding. Without you, I could never have reached this current level of success. Secondly, my advisor, who has provided patient advice and guidance throughout the research process. Thank you all for your unwavering support.

Preface

This work began as a voluntary assessment within the Data Mining subject. Although, initially the goal was to establish a line of research on the possible classification of different kinds of pollen seeds, the results obtained invited us to go deeper into the technique. Finally, the paper was published in two of the most relevant scientific journals, PLOS One and PMC, and it is found in Google scholar search. Recently this technique has been applied to a much more extensive dataset, consisting of 46 different pollen classes from New Zealand belonging to the company Veritaxa Ltd. The results have been satisfactory with a measure of accuracy close to 99% and FScore above 98%. In this dataset, pairs of seeds traditionally labeled as indistinguishable have been included, proving the model robustness, giving rise to its possible commercialization and application to different areas of work.

Contents

1 Introduction

The surge in the prevalence of allergies, with 30 percent of adults and 40 percent of children having at least one allergy according to the ACAAI [1], implies that measuring the concentrations of airborne pollen grains becomes a necessity for public health institutions and patients. The determination of such concentrations is usually performed through volumetric spore traps which capture pollen grains, which in turn have to be visually identified and counted. The morphological similarity between pollen grains from different vegetal species complicates the identification process, which is usually performed by the visual inspection of optical microscope images by an experienced human operator.

During last decades, different computational intelligence or machine learning techniques have been developed to detect objects of interest in images and to identify categories of such objects (see [2] for a comprehensive review). Models built with different approaches and data sets, with different number of classes also. This work reflects the interest in finding a global classification method capable of adapting to different data sets.

These techniques try to teach computers to do what humans do naturally, i.e., to learn from experience, and use algorithms capable to learn directly from the images without previous knowledge of the field under study. A common approach is to extract discriminant features that represent particular characteristics of the objects of interest. These extracted features are then used to develop models capable to learn and identify patterns from the image data.

In the standard automatic classification approaches, the design and selection of these features is a time-consuming manual process which requires a deep mathematical knowledge of the information that can be obtained from images, as extraction of features involves pre-processing of the images by different operations to discriminate each of them. Recently, so-called *deep learning* algorithms have been developed to automatically learn these features from images without human intervention.

These algorithms are especially suited for image processing and are being applied to solve problems such as facial recognition, motion detection, advanced driver assistance technologies such as autonomous driving, track detection, pedestrian detection and automatic parking, among many others. Given the similarities in the shape of different pollen types it is an interesting scientific challenge to study the use of deep learning techniques to develop automatic systems capable to distinguish between many different species of pollen.

In fact, over the last few years, many studies have been presented with the aim to ease the classification of pollen grains or to develop automatic classification methods. A few of the most recent make use of some forms of deep learning, and this is especially the case of the recent contribution presented in [3]. This contribution is threefold: the authors present a new, accessible and annotated pollen image dataset, they then study the performance of human operators faced to the pollen grain identification task and finally they apply machine learning to solve this same problem. The results show no great improvement of the automatic techniques over the human operators.

However, in this paper, we present three automatic approaches based on deep learning that significantly increase the percentage of correct classifications on the same image dataset. The three methods rely on the use of a convolutional neural network (CNN) to automatically extract the discriminant features of the images. In the first method, a simple classifier is used to classify the images directly from the features extracted by the CNN. The second method applies a technique known as *transfer learning* and makes use of a pre-trained deep neural network. Finally, the third method constitutes a hybrid solution of the two previous.

The structure of the paper is as follows: the following section presents a brief description of the state of the art in automatic pollen classification. Consecutively, we present the image dataset and the deep learning methods, as well as the results obtained. These results are discussed, and some conclusions are drawn in closing.

2 State of the Art

As mentioned earlier, many authors have faced the task of pollen classification. Most of these authors propose approaches based on the analytical extraction of features from the images. Amongst them, we can distinguish at least three.

The first of these approaches is based on developing features using morphological methods in which visual features, such as shape, symmetry or size are measured. In this line of work, [4] works with grain perimeter, roundness and area, then using a Fisher linear discriminant to classify up to 12 different types of pollen collected on Henderson Island, Polynesia. They reported a proportion of correctly classified pollen that depends on the subset of variables used, with the best set of variables obtaining an overall classification rate of about 95%. [5] uses changes in grain contour to train a hidden Markov model as classifier to classify 17 genders and species from 11 different families of tropical honeybees plants reporting a mean of 98.77% of success. In [6], researchers use contour-inner pollen segmentation to classify a pollen collection dataset with 15 different types and 120 observations per type. A vector support machine was used as classifier, achieving a mean of 93.8% of accuracy.

A second approach is based on texture-based methods which make use of the characteristics of the grain surface as discriminant feature. For example,[7] used measurements of gray level co-occurrence matrices, neighborhood gray level dependence statistics and entropy to classify 5 different types of pollen from three locations, reporting a 76% of success. In [8], wavelet coefficients are used as a representation of the spatial frequency and to calculate a gray level co-occurrence matrix which is in turn used to classify 7 types of pollen reporting a F-score of 0.79. Researchers in [9] apply these same techniques to classify different pollen species responsible for respiratory allergies. The resulting system is evaluated for the discrimination of species of the Urticaceae family, which are quite similar. The performance reported is about 89% of correct pollen grains. Finally, in [10] an application of segmentation methods based on texture techniques are used to classify 5 different classes of Brazilian pollen, in which texture, shape and color features were extracted from each image, obtaining over 98% of success.

The third approach is constituted by hybrid methods which make a combined use of different features. For example, [11] combines morphological features, such as area and perimeter, with Fourier descriptors and color features to train a multilayer neural network as a classifier to identify 17 pollen species, reporting a mean of 96.49% of success. [12] also classifies five pollen types by using shape (area, perimeter, diameter) and texture features (mean, standard deviation, and the entropy of gray level histograms). The approach classifies fraudulent microscopic pollen grain objects with a reported 92.3% of success.

However, there is another way of tackling the problem of pollen image classification: instead of analytically extracting features from the images, one can rely on an automated system to do that job. An example of such alternative is presented, for example, in [13], where a model learns not only the features but also the classifier itself from training data under a deep learning framework. To further enhance the classification ability, the proposal makes use of transfer learning to leverage knowledge from networks that have been pre-trained on large datasets of images, to train a dataset of 30 pollen types, achieving a 94% correct classification rate. In [3], which presents and makes use of the same dataset used in this paper, three different alternatives based on automatic feature extractors are explored. The three feature extractors are the "bag of visual words" (BOW), the "color, shape and texture" (CST) and a combination of BOW and CST (CST+BOW). They test several machine learning classifiers, including two variations of support vector machines, a decision tree based classifier and the k-nearest neighbor approach. According to these authors, the highest correct classification rate, 64%, is achieved using CST+BOW and support vector machines. Another example is [14], which uses a neural network approach to perform pollen classification for the reconstruction of remote paleo environments, reporting over 90% of the grains being correctly identified. BOW-related techniques have also been applied in [15], with a 70% reported correct classification rate over 9 pollen types, and in [16], with a reported 97.2% of accuracy over just 1 pollen variety.

When comparing different automatic classification methods, it is important to bear in mind that different experiments have different degrees of difficulty. Of course, the similarity of the grains from different species is the main challenge, but the quality of the images, for example, is crucial as well to develop good classifiers. Also, the number of tagged images is essential, as more images for each type will allow to train much more accurate models. Finally, it is very different to classify just one pollen type with a binary classifier than developing a multiclass classifier able to recognize tens of pollen types. Thus, in the above comparison of the results reported by different authors we need to consider these issues. For example, many of the aforementioned studies have been made with sets of data with a very limited number of images or pollen types. For example, [7], [8], [9], [10] , [12], [15], and [16] work with sets of data containing less than 10 pollen types. The results presented in [3], [4], [5], [6], [11], and [13] work with larger sets of data. Particularly [13] presents results over 94% for a set of images containing 30 different pollen types. Of course, as the sets contain more classes, the classification process becomes more complicated, presenting lower performances as in [3]. Attempts to classify datasets with large classes are a major challenge since they are intended to provide a more generalized method, applicable to a wider range of solutions.

The solution presented in this paper aims to provide a model capable of adapting to different data sets that contain a significant number of classes, even with similar features.

3 Materials and Methods

3.1 Image pollen dataset

In [3] the authors presented an image dataset, called POLEN23E, which consists of photos of 23 pollen types present in the Brazilian savannah: *Anadenanthera colubrina*, *Arecaceae*, *Arrabidaea*, *Cecropia pachystachya*, *Chromolaena laevigata*, *Combretum discolor*, *Croton urucurana*, *Dipteryx alata*, *Eucalyptus*, *Faramea*, *Hyptis*, *Mabea fistulifera*, *Matayba guianensis*, *Mimosa somnians*, *Myrcia*, *Protium heptaphyllum*, *Qualea multiflora*, *Schinus terebinthifolius*, *Senegalia plumosa*, *Serjania laruotteana*, *Syagrus*, *Tridax procumbens* and *Urochloa decumbens*. Fig. 1 contains a sample of each species.

The POLLEN23E dataset is publicly available and, according to its description in [3], contains 35 sample images for each pollen type. These were taken with a digital microscope at different angles, compiling a total of 805 images. However, for the type *Anadenanthera colubrina* only 20 images were found in the original source, so in our experiment 15 synthetic images were generated through rotating and scaling the original images of this type.

To assess the robustness of the models against over-fitting, the accuracy on each conducted experiment was measured using a 10-fold cross-validation process.

3.2 Convolutional neural networks for image classification

The work presented in this paper is based on the automatic extraction of discriminant features from images by deep learning convolutional neural networks (CNN).

This type of network is a variation of the well-known multilayer perceptron, however, because its application is performed in two-dimensional matrices, they

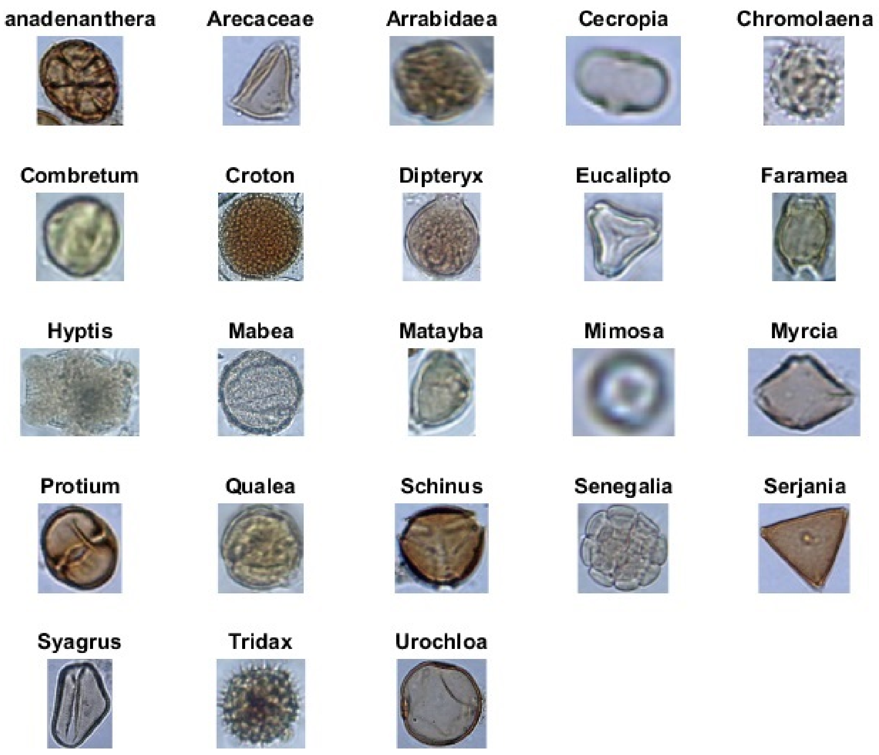


Figure 1: Sample images for each pollen type.

are effective for artificial vision tasks, like image classification and segmentation, among other. The fundamentals of CNN are based on the Neocognitron, introduced by Kunihiko Fukushima in 1980 [17]. This model was later improved by Yann LeCun in 1998 [18], introducing a learning method capable to train the network through backpropagation. In the year 2012, they were refined by [19] and implemented in a GPU, thus obtaining impressive results.

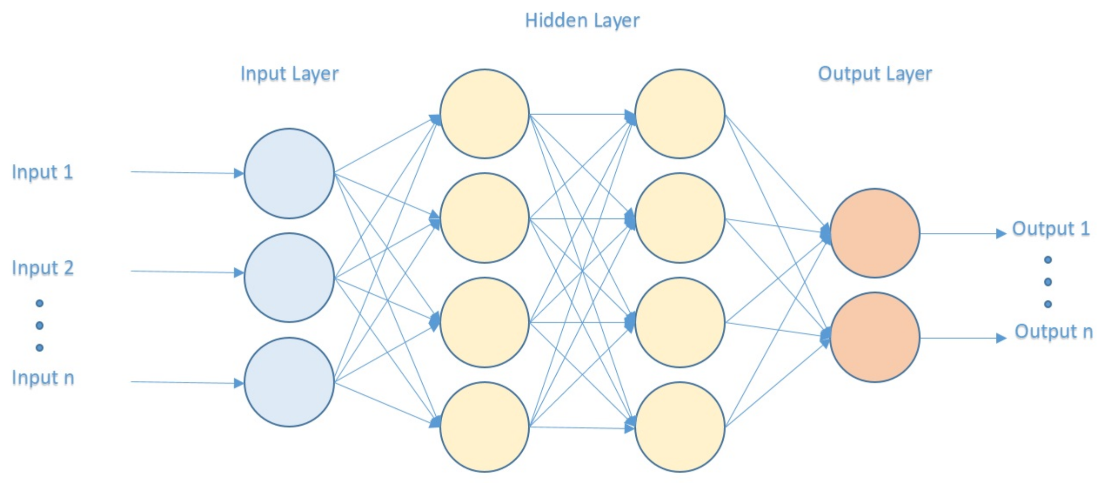


Figure 2: Neural network architecture.

The term "deep" usually refers to the number of hidden layers in the neural network. Traditional neural networks (as the multilayer perceptron shown in Fig. 2) contain only two or three hidden layers, while deep networks can have hundreds of them. In order to train these networks, extensive tagged data sets are required. CNN are amongst the most popular types of deep neural networks.

A CNN is especially suited for processing 2D data such as images as it makes use of 2D hidden ("convolutional") layers to convolve the features with the input data. The main strength of CNN is that it eliminates the need for manual feature extraction by automatically extracting the more discriminant features of a set of images. Of course, this automated feature extraction makes CNN very useful for artificial vision tasks such as object classification. In this paper, only a succinct description of CNN is included. For more information, refer for example to [20].

Fig. 3 shows a scheme of a CNN with many layers. Filters are applied to each training image with different resolutions, and the output of each convolved image

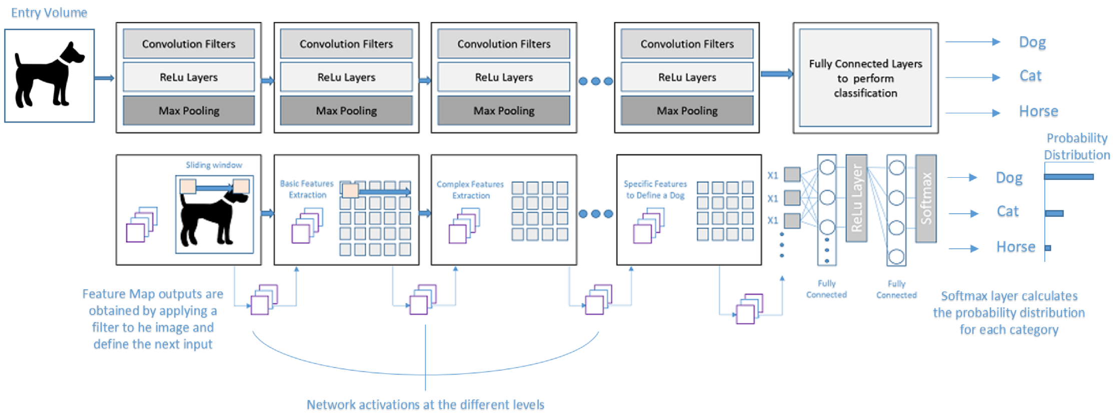

Entry Volume
Convolution Filters
ReLu Layers
Max Pooling
Fully Connected Layers to perform classification
Dog
Cat
Horse
Probability Distribution
Sliding window
Basic Features Extraction
Complex Features Extraction
Specific Features to Define a Dog
X1
ReLu Layer
Softmax
Fully Connected
Fully Connected
Feature Map outputs are obtained by applying a filter to he image and define the next input
Softmax layer calculates the probability distribution for each category
Network activations at the different levels

Figure 3: Convolutional neural network architecture and operation (based on a similar figure by [21]).

is used as input for the next layer. As shown in the figure, a CNN with a single convolutional layer consists of four fundamental components: the convolutional layer, a ReLu layer, a pooling layer and a fully connected layer.
These networks have an associated terminology and a set of concepts that is unique to them, and that the differences of other types of neural network architectures. In the following subsections we define the most important ones.

3.2.1 Input / output volumes

Convolutional neural networks are generally applied to images. Each image is an array of pixel values. The range of values that can be encoded in each pixel depends on its size in bits. Normally we have 8-bit or 1-byte pixels. Therefore, the possible range of values that a single pixel can represent is [0, 255]. However, with the colored images, particularly the RGB-based images (red, green and blue), the presence of separate color channels (3 in the case of RGB images) introduces an additional dimensional field of depth to the data. In this way, for a given RGB image of size, say 255x255 (width x height) pixels, we will have 3 matrices associated to each image, one for each of the color channels. Therefore, the whole image constitutes a three-dimensional structure called input volume (255x255x3). In Fig. 4 we can see a representation of this structure.

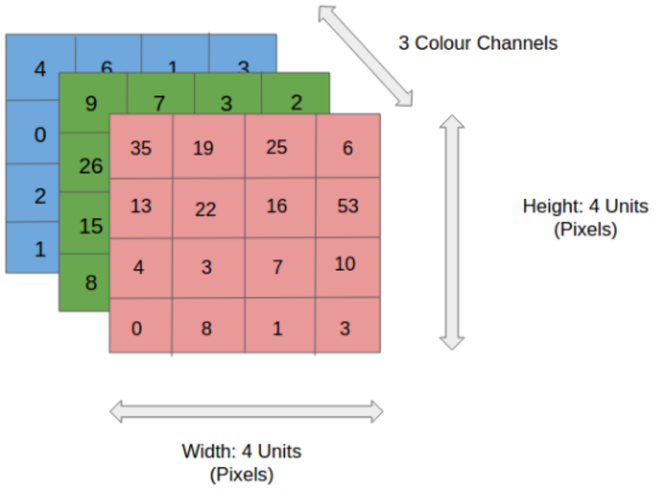


Figure 4: Entry volume structure.

3.2.2 Features

As its literal meaning implies, a feature is an observation and constitutes a differentiating and useful pattern obtained from the input data. These patterns help to perform the desired analysis of the image. The convolutional network learns the features from the input images. Typically, they emerge repeatedly from the data to gain importance. For example, when performing face detection, the fact that each human face has a pair of eyes will be treated as a feature by the system, which will be detected and learned by the different layers. In the generic classification of objects, the contours of the edges behave as features.

3.2.3 Filters

A filter (or kernel) is an integral component of the layered architecture. Generally, it refers to an operator applied to the entire image in such a way that it transforms the encoded information into the pixels. In practice, however, a kernel is a smaller array compared to the dimensions of the input image. The kernels are convolved with the input volume to obtain the so-called activation maps. Activation maps indicate activated regions, that is, regions where specific features of the kernel have been detected in the input. The actual values of the kernel matrix change with each training set learning iteration, indicating that the network is learning to identify which regions are important for extracting features from the data. The exact procedure to perform the convolution of a kernel (say, size 16 x 16) with the input volume (a RGB image of 256 x 256 x 3 in our case) involves taking portions or patches from the input image of a size equal to that of the 16 x 16 kernel), and perform the convolution process (or calculate the product) between the values of the portion and those of the kernel or kernel matrix.

The convolution value obtained by adding the resulting terms of the product results in a single entry in the activation matrix. The selection of the portion then slides (to the right or down when the limit of the matrix is reached) at a certain amount called step value or stride, and the process is repeated until the entire input image is processed. The process is performed for all color channels. For normalization purposes, we divide the calculated value of the activation matrix by the sum of values in the kernel matrix. The process is demonstrated in Fig. 5, using an example consisting of an input image of 4 x 4 pixels of 3 channels and a kernel matrix (kernel or filter) of 3 x 3.

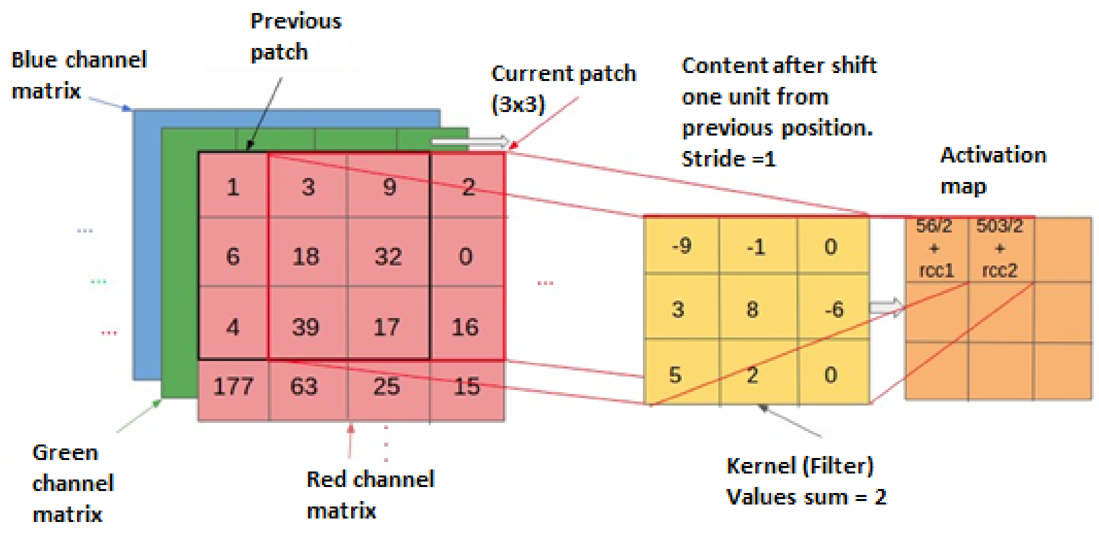


Figure 5: The convolution value is calculated by taking the product of the corresponding values in the kernel and the channel matrices. The current patch is indicated by the red outline in bold in the volume of the input image.

Therefore, in Fig. 5, the entry in the activation matrix is calculated as:

$$AM[1][2] = RCMC + GCMG + BCMC =$$

$$((3,9,2).(-9,-1,0) + (18,32,0).(3,8,-6) + (39,17,16).(5,2,0))/2 + rcc2$$

meaning:
RCMC the contribution of the matrix of the red channel, GCMC the contribution of the matrix of the green channel, BCMC the contribution of the matrix of the blue channel and the rcc2 represents the contribution of the rest of the channels.

3.2.4 Receptive field

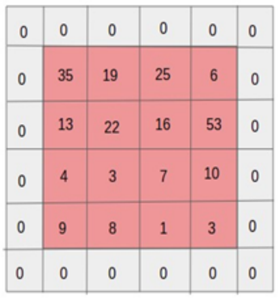
It is not practical to connect all neurons with all possible regions of the input volume. This would lead to too many weights to train and would produce too high computational complexity. Therefore, instead of connecting each neuron to all possible pixels, we specify a two-dimensional region called receptive field (for example, size 5x5 units) that extends to the entire depth of the input (5x5x3 for an input channel of three colors), within which the encompassed pixels are fully connected to the input layer of the neural network. It is on these small regions that the cross sections of the network layer (each of which consists of several neurons called depth columns) operate and produce the activation map that we previously defined.

3.2.5 Padding

Padding refers to the process of symmetrically adding zeros to the input matrix. It is a commonly used modification that allows the size of the entrance to fit our requirements. It is mainly used in the design of the layers of the convolution network when the dimensions of the input volume need to be conserved in the output volume. In Fig. 6 we can see an example of this technique.

3.2.6 Hyperparameters

In convolution neural networks the properties belonging to the structure of layers and neurons, such as the spatial arrangement and the values of the receptive field, are called hyperparameters. The hyperparameters specify only the layers. The main hyperparameters of a convolution neural network are the receptive field (R), the zero fill (P), the dimensions of the input volume (width x height x depth, or W x H x D) and step length (S).



0	0	0	0	0	0
0	35	19	25	6	0
0	13	22	16	53	0
0	4	3	7	10	0
0	9	8	1	3	0
0	0	0	0	0	0

Figure 6: A 4 x 4 matrix filled with zeros becomes a 6 x 6 matrix.

3.3 Convolutional neural networks architecture

The architecture of a typical convolution neural network is composed of multiple layers where each one performs a specific function to transform its input into a useful representation. There are three main types of layers that are commonly observed in complex neural network architectures. Next, we will describe each one of them.

3.3.1 Convolutional layer

It forms the basis of the neural network and performs the convolution operation on the input image. It consists of a three-dimensional array of neurons as a stack of two-dimensional layers of neurons, one for each channel. The output of each convolutional neuron is calculated as

$$Y_j = g\left(b_j + \sum_i K_{ij} \otimes Y_i\right), \tag{1}$$

where the output Y_j of a neuron j is a matrix which is calculated by the linear combination of the outputs Y_i of the neurons in the previous layer, each of them operated by the convolution core K_{ij} corresponding to that connection. This quantity is added to a term called influence, b_j, and then passed through a non-linear activation function $g(\cdot)$.

The convolution operator has the effect of filtering the input image with a previously trained kernel. This transforms the data in such a way that certain features (determined by the kernel shape) become more dominant in the output image as these have a higher numerical value assigned to the pixels representing them.

These kernels have specific image processing skills. An example is edge detection, that can be performed with kernels that highlight the gradient in a particular direction. However, kernels trained by a CNN are, generally, more complex for extracting abstract and nontrivial features.

Each neuron is connected to a certain region of the input volume called the receptive field (explained in the previous section). For example, for an input image of dimensions 28x28x3, if the receptive field is 5 x 5, then each neuron in the convolutional layer is connected to a region of 5x5x3 in the input volume. The region always includes all the depth of the entrance, that is, all the channel matrices. Therefore, each neuron will have 75 weighted entries. For a particular value of R (receptive field), we have a cross section of neurons entirely dedicated to the inputs of this region. This cross section is called the depth column and extends to the entire depth of the convolution layer.

To optimize the convolution layer, we can use a shared weight model that reduces the number of unique weights to train and consequently the matrix calculations that are made per layer. In this model, each depth cut, or a single two-dimensional layer of neurons, in the convolutional architecture share the same weights. One aspect to consider in this model is that the shared use of parameters does not work well with images that cover a spatially centered structure (such as face images), and in applications where we want the different features of the image to be detected in locations spatially different from the layer.

The network works in the same way as a feed-forward network, the weights in the convolutional layers are trained and updated in each learning iteration using an extended back-propagation algorithm to be able to be applied to three-dimensional structures of neurons. In Fig. 7 we can see a representation of the receptive field with shared weights.

3.3.2 ReLu Layer

The term ReLu refers to the rectifier linear unit, the more frequently used activation function for the neuron outputs in CNN. Mathematically, it is described as $\max(0, x)$.

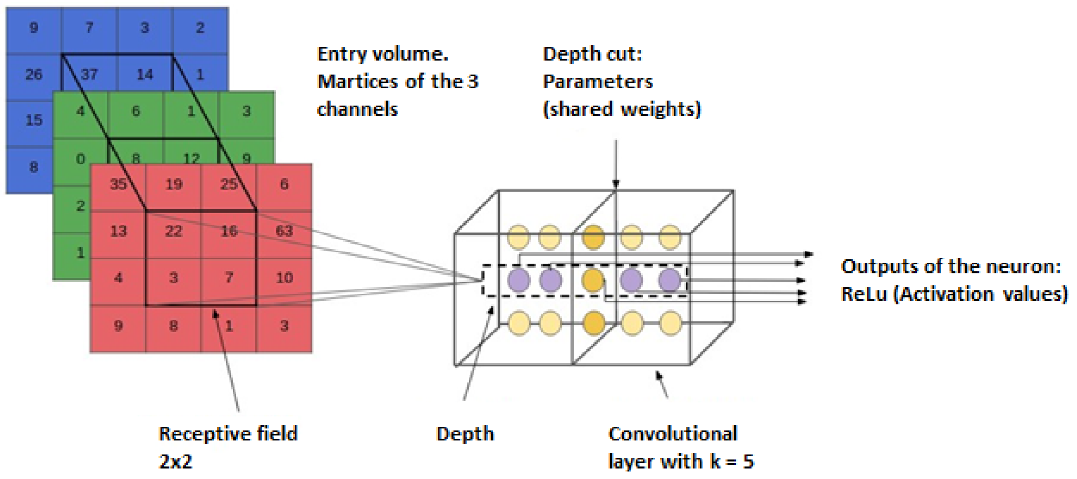

Entry volume. Martices of the 3 channels
Depth cut: Parameters (shared weights)
Outputs of the neuron: ReLu (Activation values)
Receptive field 2x2
Depth
Convolutional layer with k = 5

Figure 7: Receptive field concept.

Unfortunately, the ReLu function is not differentiable at the origin, making it difficult to be used with the backpropagation algorithm. Instead, a smoothed version called *softplus* function is frequently used. This function is described as

$$f(x) = \ln(1 + e^x). \tag{2}$$

Fig. 8 shows both ReLu and softplus functions.

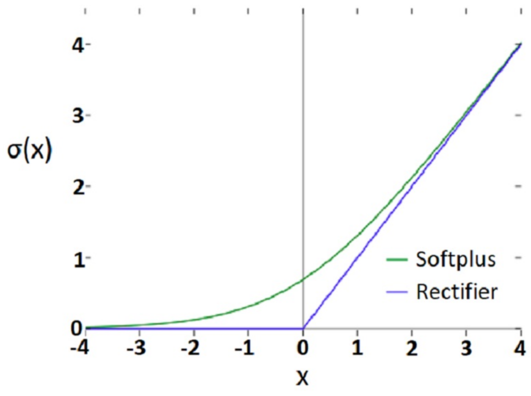


Figure 8: Rectifier linear unit and softplus functions.

3.3.3 Pooling Layer

Neural networks are known to have some tolerance to small perturbations in the input data. For example, if two almost identical images (different only by a few pixels) are fed through a neural network, the result should be essentially the same. In the case of CNN, this is obtained, in part, by the reduction of sampling. By reducing the resolution, the same features will correspond to a larger activation field in the input image. Operations such as max-pooling are very effective in summarizing features about a region. The max-pooling operation finds the maximum value between a sample window and passes this value as a feature summary over that area. As a result, the size of the data is reduced by a factor equal to the size of the sample window on which it is operated. Fig. 9 shows max-pooling operation.

The pooling layer is usually placed after the convolutional layer. Its main porpoise lies in reducing the spatial dimensions (width x height) of the input volume for the next convolutional layer, while guaranteeing a spatial invariance of the features obtained by the filters. It does not affect the depth dimension of the volume. The operation performed by this layer is also called down sampling, since the reduction in size also leads to the loss of information. However, this loss is beneficial for the network for two reasons, first, the decrease in size leads to a lower computational overload for the next layers of the network, and second, reduces the network overfitting.

Like the convolution operation performed previously, the pooling layer adopts a sliding window or a specific region that moves through the input, transforming the values into representative values. The transformation is normally carried out taking the maximum value of the observable values in the window (called max pooling) or taking the mean of the values (mean pooling). The operation is performed for each depth cut. Like the convolution operation performed previously, the pooling layer adopts a sliding window or a specific region that moves through the input, transforming the values into representative values. The transformation is normally carried out taking the maximum value of the observable values in the window (called max pooling) or taking the mean of the values (mean pooling). No new parameters are introduced into the matrix by performing this operation. The operation can be considered as the application of a function on the input values, having portions of fixed size and, in turn, this size being a modifiable parameter.

Pooling is optional in convolution neural networks, and many architectures do not implement it.

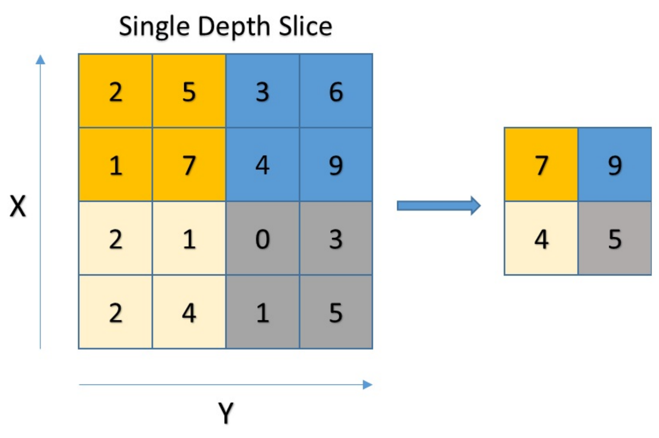


Figure 9: Max-pooling operation.

3.3.4 Fully Connected Layer

The last layer is fully connected to the output of the previous layer, in the sense that every neuron of one layer is connected to all the neurons in the next one. Fully connected layers are typically used in the last stages of the networks to perform the high-level feature detection and are no different from regular multilayer perceptron hidden layers.

3.4 Deep learning convolutional neural networks

Many real-world problems are being addressed efficiently using convolutional neural networks. More complex problems, such as objects recognition, require the use of deep neural networks, deep learning, with millions of parameters to obtain last generation results. In the next section we will describe the basic concepts of these networks that are the objective of our study in its application to the image recognition problem that we raised at the beginning of this work.

Recognizing objects in images is a complex problem that involves multiple conflicting objectives. Deep convolutional neural networks, trained in large data sets, achieve satisfactory results and are currently the most advanced approach to this task. However, the resources needed to train such deep networks present a great disadvantage. Therefore, we will approach our problem using a previously trained network, in this case AlexNet, from which we will later describe its characteristics.

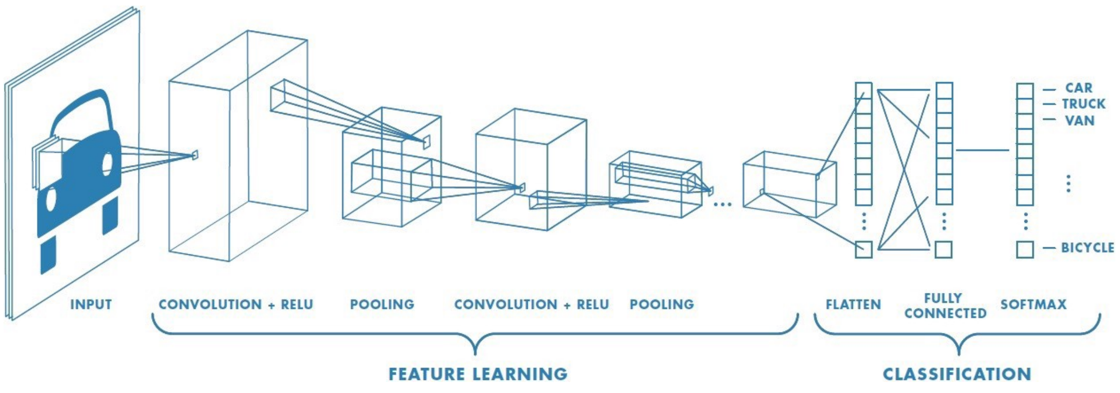


Figure 10: Layer structure of a deep learning neural network.

The deep learning algorithms try to learn multiple levels of representation of increasing complexity of abstraction. Deep learning uses neural networks to learn useful representations of features directly from the data. Neural networks combine multiple layers of nonlinear processing, using simple elements that work in parallel and inspired by biological systems. Deep learning models can achieve state-of-the-art precision in the classification of objects, sometimes exceeding the level of human performance.

The training of these networks is carried out by data sets of a large number of labeled examples and with neural network architectures that contain a large number of layers, including, in most cases, convolutional layers, as we have previously described. As mentioned previously, the training of these models is computationally intensive and can usually be accelerated through the use of high performance GPUs.

Fig. 10 shows how convolutional neural networks combine layers that automatically learn features of many images in order to classify new images. As we can see, this structure corresponds to what we previously defined as a convolutional neuronal network.

3.4.1 Alexnet

AlexNet is the name of a convolutional neuronal network, designed by Alex Krizhevsky, that runs on GPUs implemented in CUDA. This network consists of 11 layers distributed as shown in Fig. 11.

The network has a very similar architecture as LeNet by Yann LeCun, but deeper, with more filters per layer, and with stacked convolutional layers. It consists of $11x11$, $5x5$, $3x3$, convolutions, max pooling, dropout, data augmentation, ReLU activations, SGD with momentum. It attaches ReLU activations after every convolutional and fully connected layer.

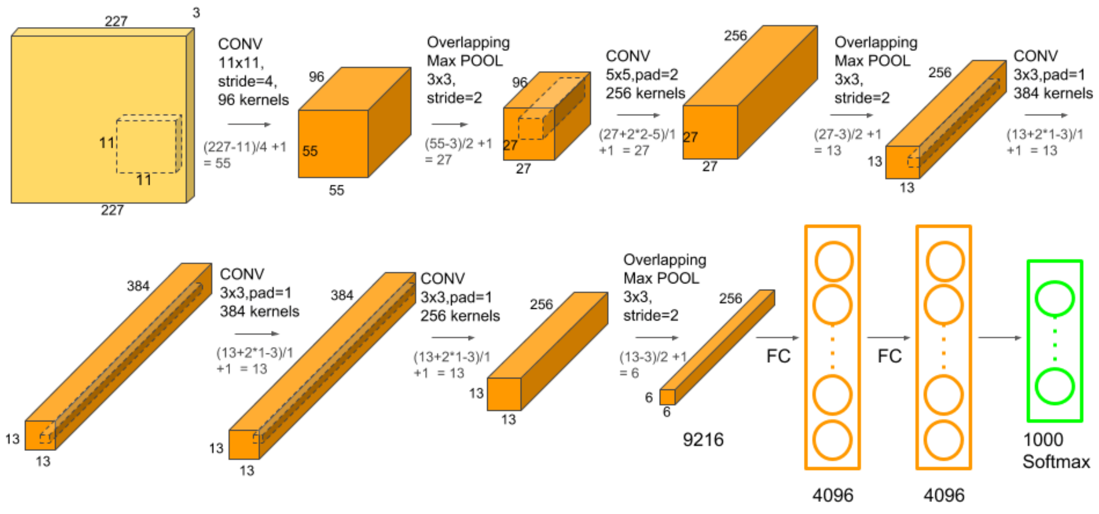


Figure 11: Layer structure of Alexnet.

The layers of the network are configured in the following way:

- Layer 0: Entry Volume.
  - Size: 227x227x3.

- Layer 1: Convolutional layer with 96 filters, size 11x11, stride 4, padding 0.
  - Size: 55x55x96.
  - Output size: ((227 11)/4) + 1 = 55.
  - Depth 96, because of the 96 filters.

- Layer 2: Max pooling with 3x3 filter, stride 2.
  - Output size: ((55 3)/2) + 1 = 27.
  - Depth 96, because of the 96 filters.

- Layer 3: Convolutional layer with 256 filters, size 5x5, stride 1, padding 2.
  - Size: 27x27x256.
  - The original size is recovered by padding. (5 1/2) = 2.
  - Depth 256, because of the 256 filters.

- Layer 4: Max pooling with 3x3 filter, stride 2.
  - Size: 13x13x256.
  - Output size: ((27 3/2) + 1) = 13.
  - Size: The same as the previous one (256) since the pooling is done independently in each layer.

- Layer 5: Convolutional layer with 384 filters, size 3x3, stride 1, padding 1.
  - Size: 13x13x384.
  - The original size is recovered by padding. (3 1/2) = 1.
  - Depth: 385, because of the 385 filters.

- Layer 6: Convolutional layer with 384 filters, size 3x3, stride 1, padding 1.
  - Size: 13x13x384.
  - The original size is recovered by padding. (3 1/2) = 1.
  - Depth: 384, because of the 384 filters.

- Layer 7: Convolutional layer with 256 filters, size 3x3, stride 1, padding 1.
  - Size: 13x13x256.
  - The original size is recovered by padding. (3 1/2) = 1.
  - Depth: 256, because of the 256 filters.

- Layer 8: Max pooling layer with 3x3 filter, stride 2.
  - Size: 6x6x256.
  - Output size: ((13 3/2) + 1) = 6.
  - Depth: The same as the previous one (256) since the pooling is done independently in each layer.

- Layer 9: Fully connected layer with 4096 neurons.
  - In this layer, each of the 6x6x256 = 9216 pixels are fed to each of the 4096 neurons and the weights determined by back propagation.
- Layer 10: Fully connected layer with 4096 neurons.
  - As in the previous layer, each of the 6x6x256 = 9216 pixels are fed to each of the 4096 neurons and the weights determined by back propagation.
- Layer 11: Fully connected layer with 1000 neurons.
  - This is the last layer and it has 1000 neurons because the IMAGENET dataset (database for which AlexNet was initially trained) has 1000 classes to predict.

3.4.2 Features detection

Convolutional neural networks use features to classify images. The network learns these features itself during the training process. What the network learns during training is sometimes unclear.

The convolutional layers of a network output multiple 2-D arrays. Each array (or channel) corresponds to a filter applied to the layer input. The channels output by fully connected layers correspond to high-level combinations of the features learned by earlier layers.

There are five 2-D convolutional layers in the AlexNet network. The convolutional layers towards the beginning of the network have a small receptive field size and learn small, low-level features. The layers towards the end of the network have larger receptive field sizes and learn larger features.

Each layer of the convolutional neural network produces a response, or activation, to the input image. Only a few layers of the convolutional neuronal network are suitable for the extraction of image features. To represent this, we are going to visualize the network filters weights in the convolutional layers and in the fully connected layers. This will help us to have a first idea about why the features we extract using the convolutional neural network work so well for image recognition tasks.

The features on the first convolutional layer, which is the second layer in the network, mostly contain edges and colors, which indicates that the filters at layer 1 are edge detectors and color filters. Fig. 12 shows the first 56 features learned by the first convolutional layer. We can see how the first layer of the network has trained

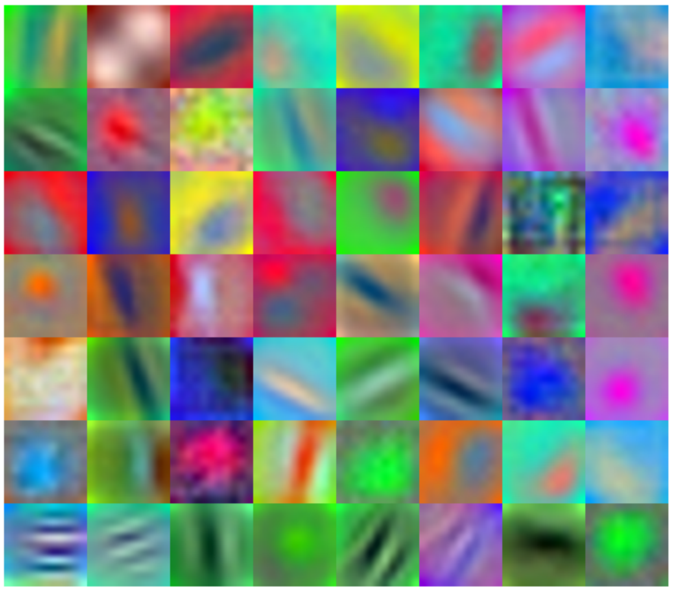


Figure 12: First convolutional layer features.

the filters to learn the basic features that we mentioned earlier. These primitive features are processed by deeper layers of the network, combining the first features to create new features of the top-level image. These higher-level features are more suitable for identification tasks since they combine all the primitive features in a better representation of the image.

Similarly, we can obtain the features for the second convolutional layer. These features are created using the features from the first convolutional layer. The second convolutional layer corresponds to layer 6. Fig. 13 shows the first 30 features learned by this layer. For each of the remaining convolutional layers, we visualize the first 30 features learned. Convolutional layers 3 (Fig. 14), 4 (Fig. 15) and 5 (Fig. 16) correspond to layers 10, 12 and 14 respectively. Notice that the layers which are deeper into the network yield more detailed filters.

There are three fully connected layers in the AlexNet model. The fully connected layers are towards the end of the network and learn high-level combinations

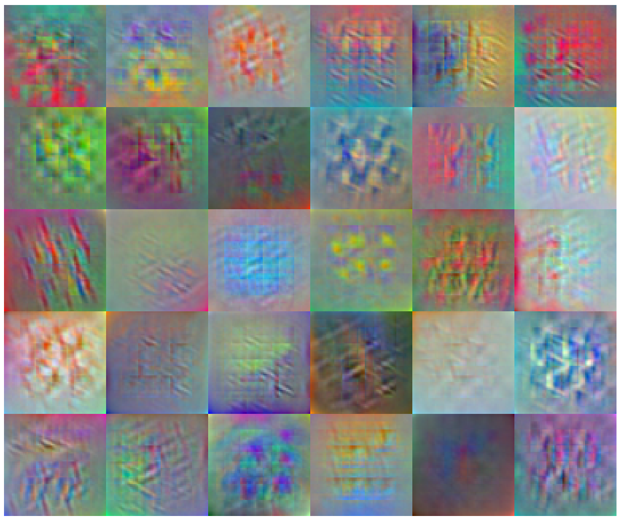


Figure 13: Second convolutional layer features.

of the features learned by the earlier layers.

Fig. 17, Fig. 18 and Fig. 19 shows the fully connected layers features. Fig. 19 shows the last fully connected layer features for classes anadenanthera, chromolaena, eucalipto, mabea, senegalia, urochloa. This layer shows the highest level features of the convolutional network.

3.4.3 Activations visualization

In this section we will feed an image to a convolutional neural network, Alexnet, and display the activations of different layers of the network. We will examine the activations and discover which features the network learns by comparing areas of activation with the original image, find out that channels in earlier layers learn simple features like color and edges, while channels in the deeper layers learn more complex features. Identifying features in this way can help us to understand what the network has learned. To illustrate an example of the activations we will use an image belonging to the Arecaceae class. Fig. 20 shows the image. Fig. 21 displays the 96 images on an 8-by-12 grid, one for each channel in the first convolutional

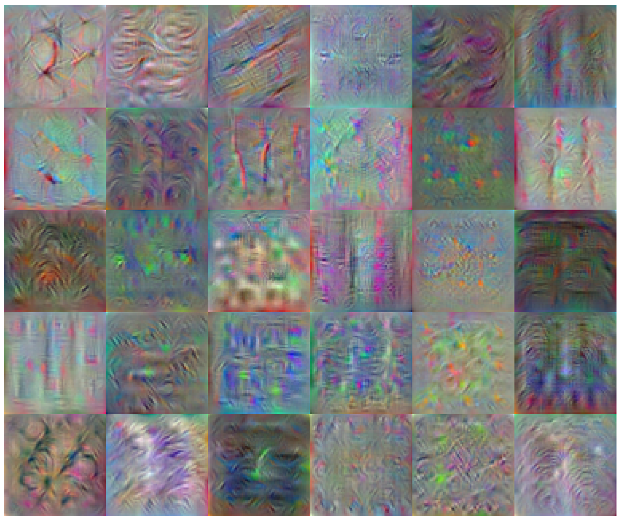


Figure 14: Third convolutional layer features.

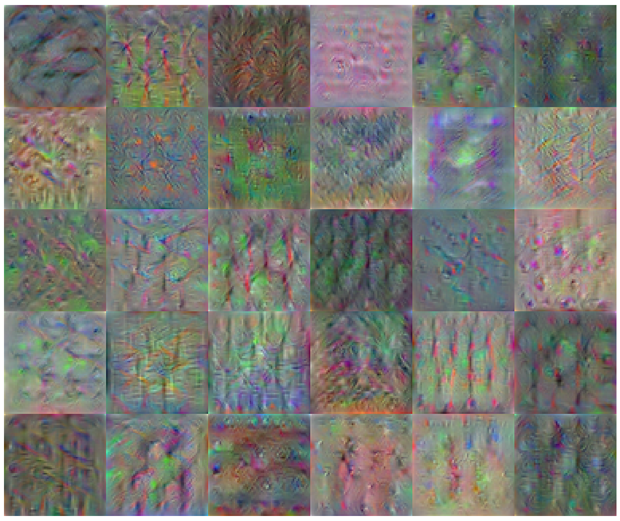


Figure 15: Forth convolutional layer features.

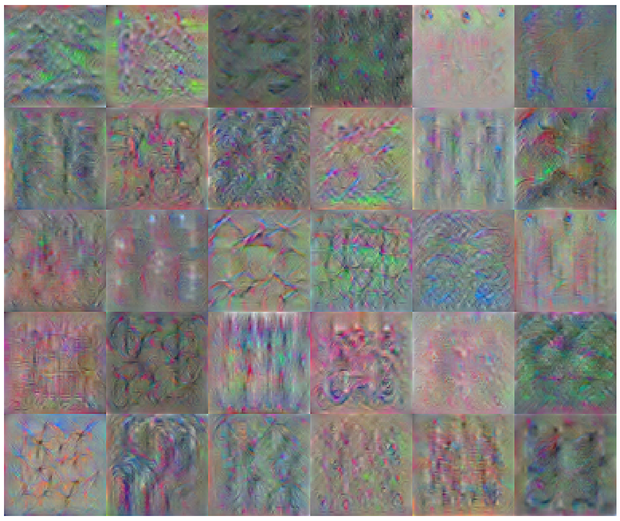


Figure 16: Fifth convolutional layer features.

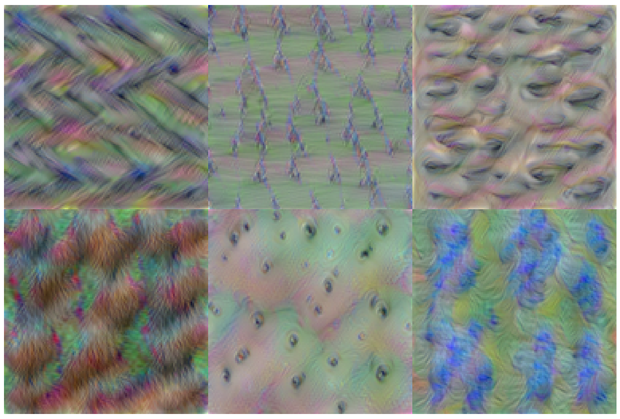


Figure 17: First fully connected layer features.

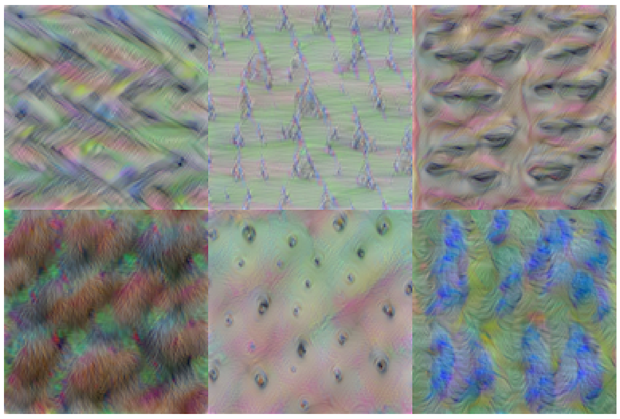


Figure 18: Second fully connected layer features.

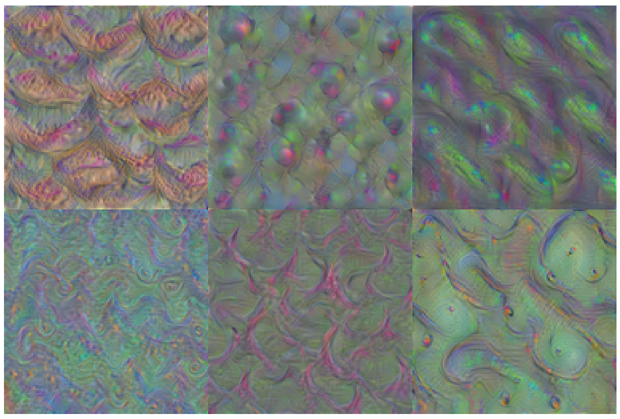


Figure 19: Third fully connected layer features.

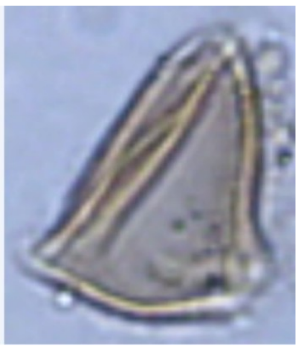


Figure 20: Arecaceae class sample image.

layer. All activations are scaled so that the minimum activation is 0 and the maximum is 1. Each tile in the grid of activations is the output of a channel in the first convolutional layer. White pixels represent strong positive activations and black pixels represent strong negative activations. A channel that is mostly gray does not activate as strongly on the input image. The position of a pixel in the activation of a channel corresponds to the same position in the original image. A white pixel at some location in a channel indicates that the channel is strongly activated at that position. To illustrate this concept, we are going to resize the activations in channel 85 to have the same size as the original image and display the activations. in Fig. 22, we observe easily how this channel is detecting the edges for the processed image. We can compare to the original image and notice that this channel activates on edges. This channel represents at the same time the channel with the highest activation in the first convolutional layer for the processed image.

Most convolutional neural networks learn to detect features like color and edges in their first convolutional layer. In deeper convolutional layers, the network learns to detect more complicated features. Later layers build up their features by combining features of earlier layers. Fig. 23 shows the fifth convolutional layer in the same way as we did with the first convolutional layer. There are too many images to investigate in detail, so we will focus on some of the more interesting ones. In Fig. 24, we display the strongest activation in the fifth convolutional layer. In this case, the maximum activation channel is not as interesting for detailed features

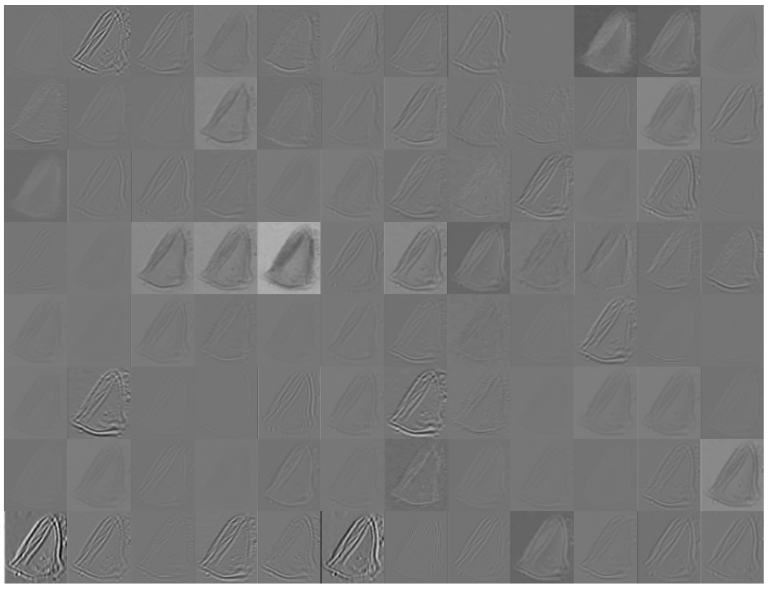


Figure 21: Activations of first convolutional layer.

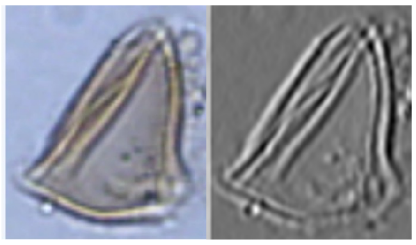


Figure 22: Activations of channel 85 at the first convolutional Layer.

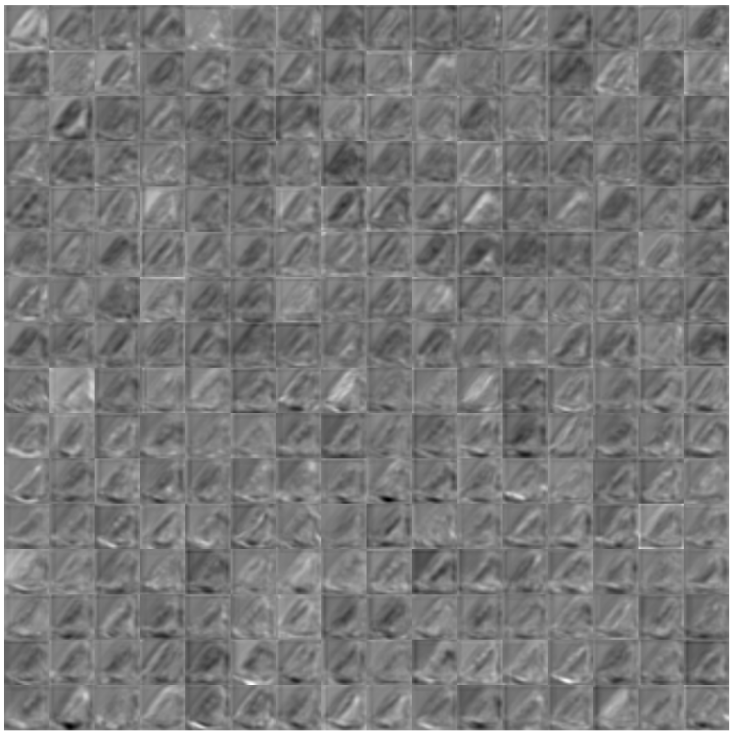


Figure 23: Activations of fifth convolutional layer.

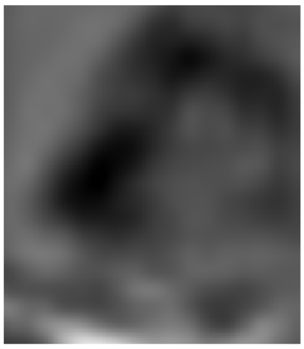


Figure 24: Strongest activation of fifth convolutional layer.

as some others and shows strong negative (dark) as well as positive (light) activations. In the grid of all channels, there are channels that might be activating on edges, like channels 1 and 30 showed on Fig. 25. Many of the channels contain

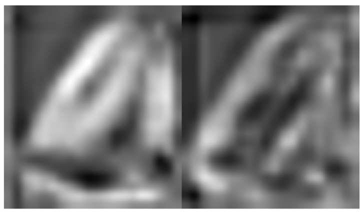


Figure 25: Better activations of fifth convolutional layer.

areas of activation that are both light and dark. These are positive and negative activations, respectively. However, only the positive activations are used because of the rectified linear unit (ReLU) that follows the fifth convolutional layer. To investigate only positive activations, we can repeat the analysis to visualize the activations of the fifth ReLu layer. Compared to the activations of the fifth convo-

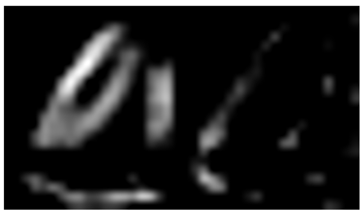


Figure 26: Activation of fifth ReLu layer.

lutional layer, the activations of the fifth ReLu layer clearly pinpoint areas of the image that have strong border features. The activations in the different convolutional and ReLu layers help us to understand how the network learns to identify the features that differentiate the classes.

3.5 Linear discriminant classifier

When using a CNN as a feature extractor, as described above, it is important to devise an automatic learning classifier that takes the extracted features and discriminates the input images according to them. In this document we will use a linear discriminant classifier (LDC) for this task.

Linear discriminant analysis is normally used as a technique for dimensionality reduction in machine learning applications. The objective is to project a data set in a space of smaller dimensions with a better capacity for class separation. In summary, the objective of the linear discriminant classifier is to project a space of features (a set of n-dimensional samples) into a smaller subspace k (where k n - 1) while maintaining the discriminatory information of the class. In other words, a linear discriminant classifier tries, explicitly, to model the difference between the data classes. In general, the reduction of dimensionality not only helps to reduce computational costs for a given classification task, but it can also be useful to avoid overfitting by minimizing error in parameter estimation.

The linear discriminant analysis is closely related to the analysis of variance (ANOVA) and the regression analysis and is a popular classification algorithm because it is fast, accurate and easy to interpret. It is also especially suitable for large datasets and assumes that different classes generate data based on different Gaussian distributions. To train the classifier, the adjustment function estimates the parameters of a Gaussian distribution for each class and then creates linear boundaries between the classes. It is a quick prediction, uses little memory and is easy to interpret, although it has a comparatively low flexibility.

3.6 Data augmentation

Due to the low number of images available in the different classes, we have created an image data augmenter that preprocesses images before training. This augmenter rotates images by random angles in the range [0, 360] degrees. Rotating the images produces new samples that improve the classification. This is a very common technique that allows to increase the size of the training sets, to facilitate features detection.

4 Experimental design

To avoid overfitting and to test the robustness and generalization ability of the proposed methods, a 10-fold cross-validation scheme has been applied. For each fold, the original dataset has been divided into two subsets of images: a training set and a test set, containing 90% and 10% of the observations respectively, reserving the last one to test the models after the training process. Since there are 35 images per pollen type, in each fold the training set contains 32 images and the test set is composed of 3 images.

From the previously described techniques, we implemented and trained the following combinations. Setup A: Feature extraction and linear discriminant classifier (FE+LD). This approach is based on the idea of taking advantage of a pre-trained convolutional neural network architecture (in our case, AlexNet) to perform feature extraction, and then use a LD classifier upon these features.

The AlexNet network builds a hierarchical representation of the images. Deeper layers hold sets of higher level extracted features, constructed from lower level extracted features from previous layers. Thus, the first layer of the network consists of convolutional filters which learn the basic features. These primitive features are then processed by deeper layers of the network, combining the first features to create new ones on a higher level. Selecting a particular layer to extract features is a design consideration, but it is common to select the layer immediately before the classification layer.

We have adopted this convention in Setup A, selecting the seventh convolutional layer of AlexNet to obtain the representative features.

Fig. 27 shows how the afore mentioned cross-validation scheme has been applied in this setup. For each of the 10 folds, the dataset is initially divided into two subsets, training and test. Each training set is used to perform the feature extraction process. The resulting set of extracted features is used to train the LD classifier. Once the classifier has been trained with these extracted features, it is used to classify the images of the test set reserved at the beginning of the procedure.

Setup B: Transfer learning (TL). In this setup, the pre-trained CNN AlexNet has been adjusted to learn the features of the POLLEN23E images dataset. The last three layers of the AlexNet are originally configured for 1000 classes, and they must be fine-tuned for the new classification problem. For this purpose, all layers are extracted, except the last three, from the pre-trained network and transferred to learn the new classification task by replacing these last three layers with a new fully connected layer, a softmax layer, and a classification output layer, all adjusted

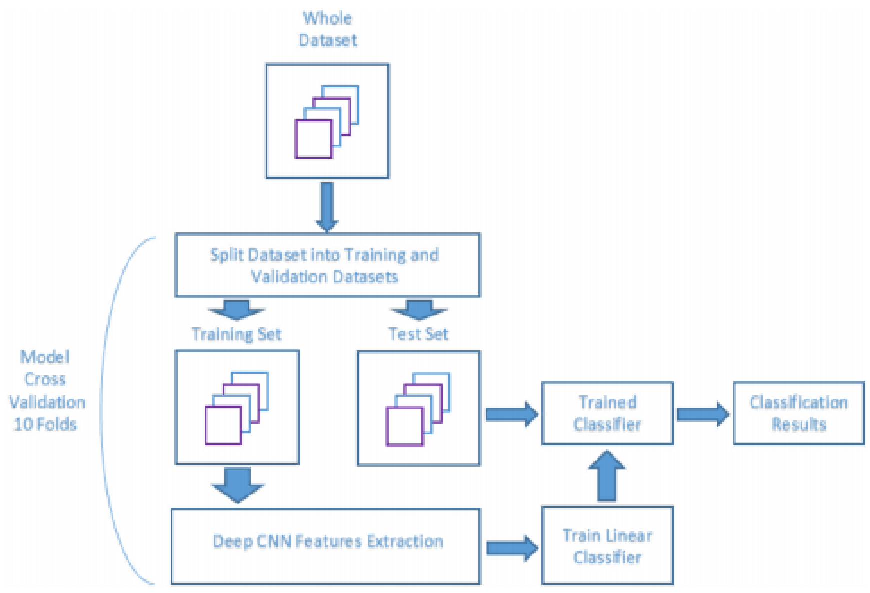

Whole Dataset
Split Dataset into Training and Validation Datasets
Training Set
Test Set
Model Cross Validation 10 Folds
Trained Classifier
Classification Results
Deep CNN Features Extraction
Train Linear Classifier

Figure 27: Cross-validation schematic for setups A and C.

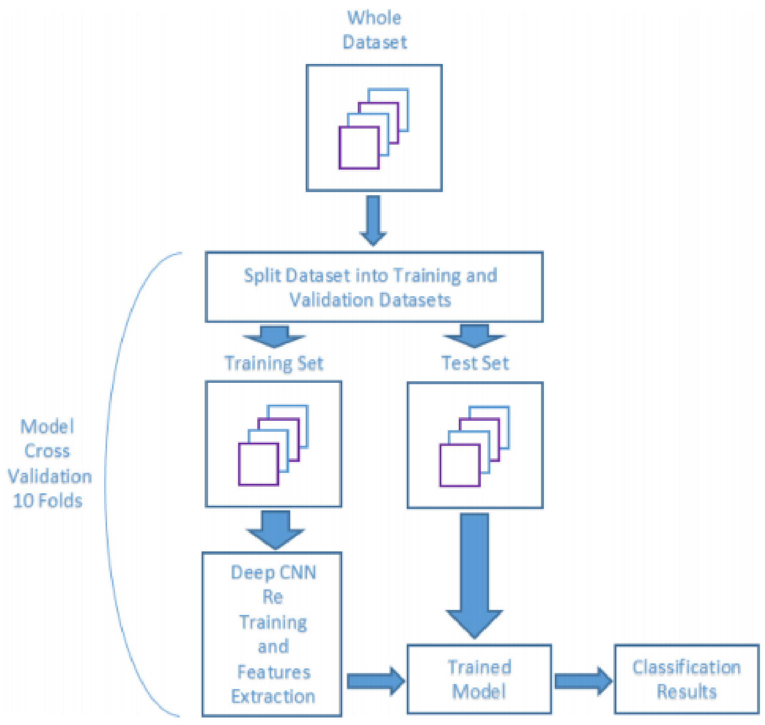

Whole Dataset
Split Dataset into Training and Validation Datasets
Training Set
Test Set
Model Cross Validation 10 Folds
Deep CNN Re Training and Features Extraction
Trained Model
Classification Results

Figure 28: Cross-validation schematic fo setups B.

to 23 classes. Then the network is retrained to learn the new task.

The idea is to keep the features pre-learnt in the convolutional layers of the AlexNet and force the learning just in the fully connected ones. Hence, we set a low learning rate (0.0001) in the transferred layers to fine-tune them and a higher learning rate (0.2) in the fully connected layers to allow learning in the newly created layers. This combination of learning rate settings results in fast learning only in the new layers and slower learning in the early ones. Fig. 28 shows the schematic description of the cross-validation process applied in this setup to retrain the CNN and extract the features from the training dataset, and then apply the retrained network to the test set.

Setup C: Transfer learning, feature extraction and linear discriminant (TL+FE+LD). This setup is a hybrid approach aimed to preserve the advantages of the two previous setups. As stated above, through transfer learning we retrain the last three layers of the network leaving the previous layers practically unchanged by setting a very low learning rate for them. However, during the training process, it is observed that even a low learning ratio for these previous layers of the network is enough to modify the weights of the different convolutional filters, allowing them to incorporate features of the new provided images, and thus increasing the classification performance. Then, once applied transfer learning, it is possible to extract the features from the fine-tuned CNN and subsequently apply a classifier. In this approach, the learnt features can be extracted from the layer immediately before the classification layer, exactly as we did in setup A. However, we can also extract the features from the fully connected layer or even the softmax layer recently retrained.

For a dataset like POLEN23E, with a reduced amount of observations, it is common to select the first option, that is, extracting the features from the layer immediately before the classification layer. In this case, the convolutional filters from the earlier layers will have slightly changed their weights adding new features but keeping most of the original network features. We keep the same learning rates applied in setup B, using a low learning rate (0.0001) in the transferred layers and a higher learning rate (0.2) in the fully connected layer.

For sets of images with a higher amount of observations, extracting features from the new convolutional layers can be a good alternative to increase the performance.

Once the network has been re-trained and the features extracted, a linear discriminant classifier is built to classify the images. As it was the case with setup A and B, a 10-fold cross-validation procedure has been applied. The schematic of this procedure for setup C is the same as for setup A, shown in Fig. 27.

5 Results

This section presents the results obtained in the experiments. As mentioned above, these experiments correspond to the application of three different setups based on different combinations of techniques. In order to compare the results of the three setups amongst them and with previous works, we use the following commonly used performance measures for classification:

$$\text{CCR} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

$$\text{precision} = \frac{TP}{TP + FP} \tag{4}$$

$$\text{recall} = \frac{TP}{TP + FN} \tag{5}$$

$$\text{F-score} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{6}$$

In all these, TP refers to true positives, TN to true negatives, FP to false positives, and FN to false negatives.

In machine learning, it is important to compare models using data previously unseen during the training process. These data constitute an example close to reality, since they are completely unknown to the models and allow to measure their behavior against new observations. Table 1 presents the results obtained by the three setups using the validation datasets. This table shows averages of the cross-validation results for the correct classification rate (CCR), precision, recall, and F-score. In brackets under the values, the standard deviation for each measurement is also shown.

From the table it is clear that Setup C obtains the best results in all indices. A very high CCR value, 97.2273 %, is obtained with a low standard deviation of 0.9000%. This is accompanied by high precision and recall values which verify its good performance against false positives and false negatives, also showing a high

Table 1: Average validation results of the cross-validation procedure for each setup. Results from [3] are computed from their confusion matrices.

	CCR (%)	precision	recall	F-score
CST+BOW (from [3])	68.5714	0.3988	0.8203	0.5366
Human (from [3])	63.5710	0.3030	0.6279	0.4087
Setup A (FE+LD)	96.6247 (± 1.1107)	0.9366 (± 0.0210)	0.9955 (± 0.0016)	0.9592 (± 0.0137)
Setup B (TL)	96.1529 (± 1.2089)	0.9275 (± 0.0229)	0.9949 (± 0.0020)	0.9541 (± 0.0151)
Setup C (TL+FE+LD)	97.2273 (± 0.9000)	0.9477 (± 0.0170)	0.9964 (± 0.0014)	0.9669 (± 0.0115)

F-score value (0.9669) with a low standard deviation (0.0115). The results for CCR are displayed in Fig. 29.

In this figure, we observe that setup C is better both in terms of median CCR and in the dispersion of this magnitude through the different cross-validation folds. The bottom "whisker" for this setup is located above 80%, meaning this is the minimum CCR we can expect for any class. The median is clearly above the value of setup B and slightly better than the value obtained by setup A. However, the latter bottom "whisker" places the minimum near 0.6 and very far from the first quartile, which suggests discarding this model as a solution. Setup C is also superior to setup B in terms of the first and third quartiles.

Fig. 30, Fig. 31 and Fig. 32 show the confusion matrices for the three setups. These matrices have been constructed by accumulating the individual cross-validation results in the validation dataset. Again, it can be easily observed that setup C shows better performance than the other two.

In order to compare the computational complexity of each setup, Table 2 presents the prediction speed (measured in observations processed per second) and the total training time for each setup. We observe that the three models offer reasonable training times, highlighting the setup C prediction speed, with approximately 170 predictions per second.

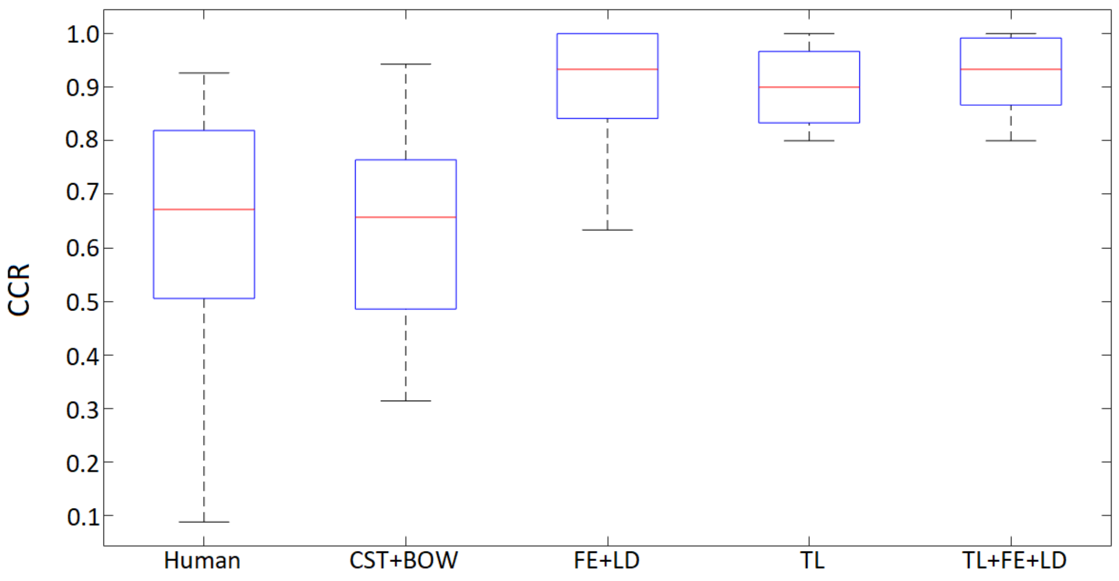
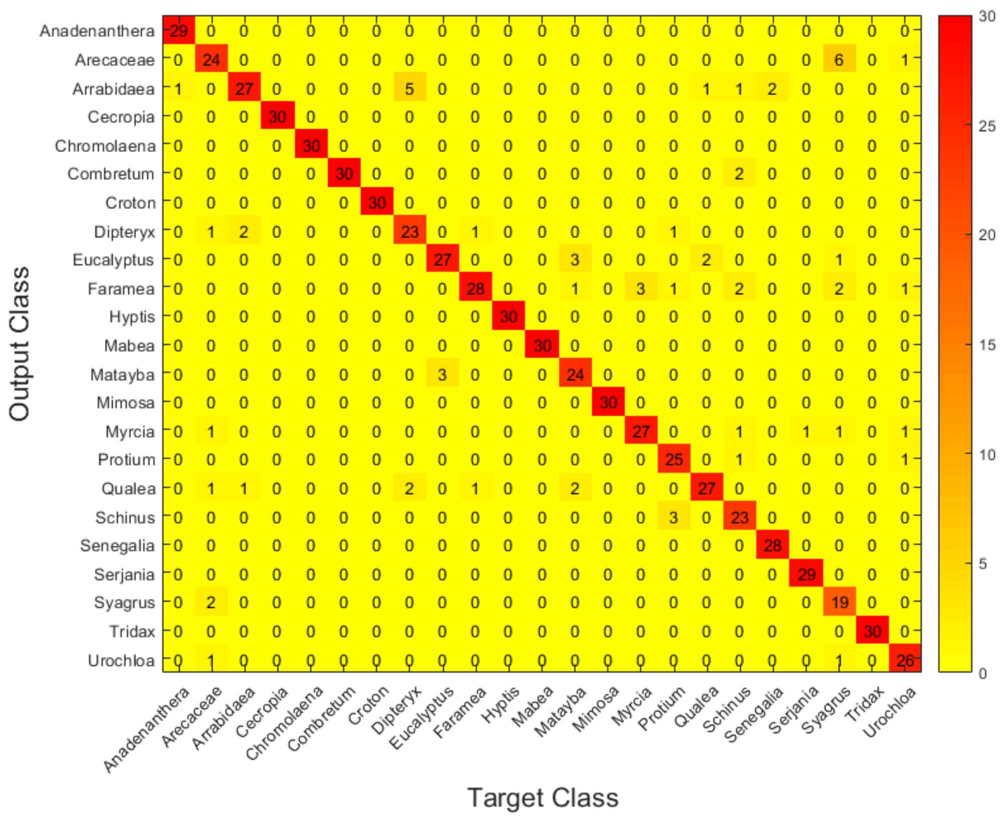


Figure 29: Correct classification rate for human operators and the best model of [3] and the three proposed deep learning setups.

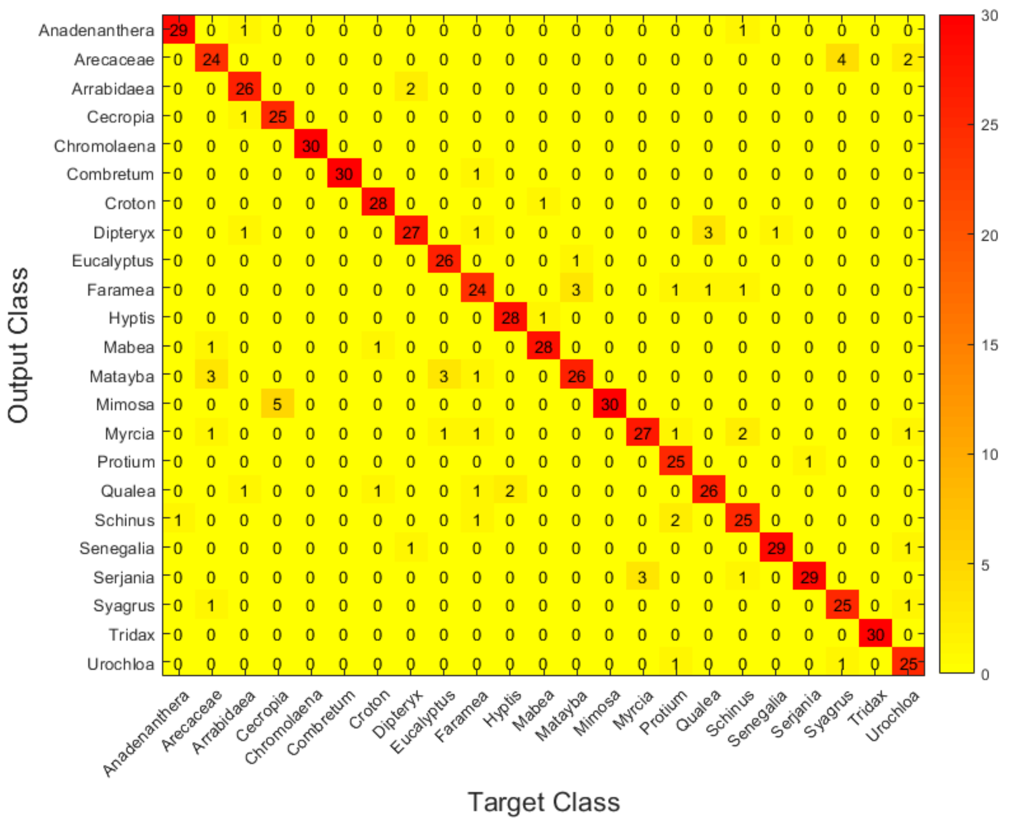
Table 2: Measures for computational complexity.

	Prediction speed	Training time
Setup A (FE+LD)	∼140 obs/sec	8.69 min
Setup B (TL)	∼155 obs/sec	16.49 min
Setup C (TL+FE+LD)	∼170 obs/sec	16.61 min



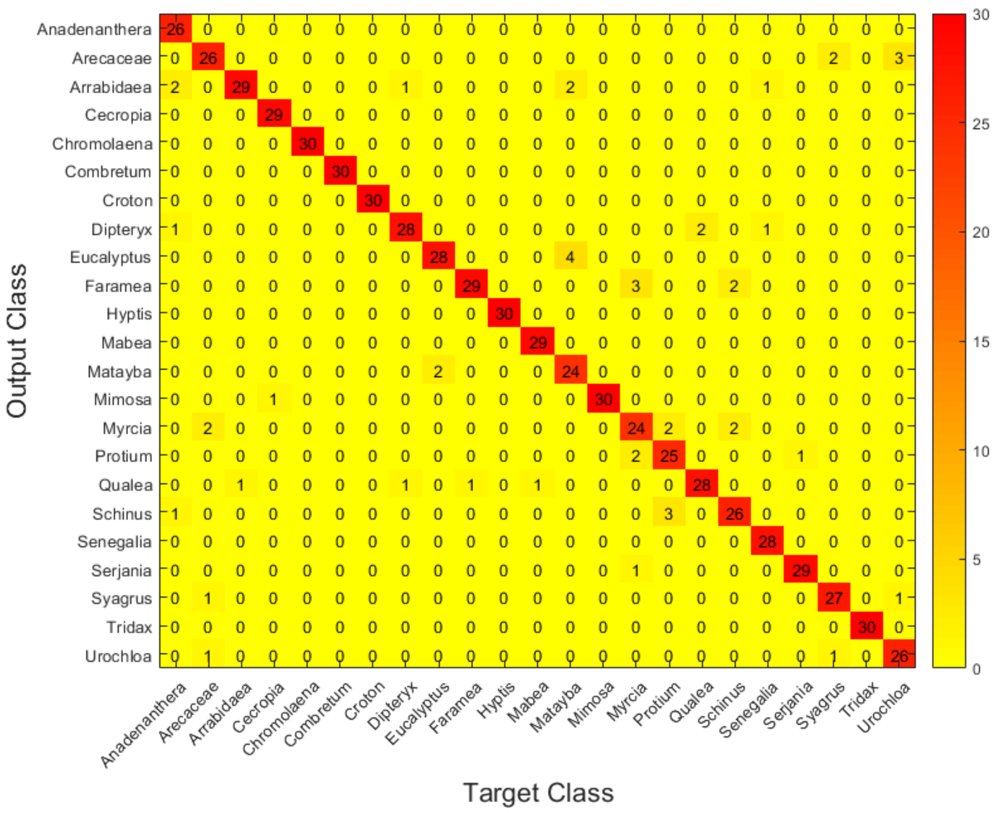
	Anadenanthera	Arecaceae	Arrabidaea	Cecropia	Chromolaena	Combretum	Croton	Dipteryx	Eucalyptus	Faramea	Hyptis	Mabea	Matayba	Mimosa	Myrcia	Protium	Qualea	Schinus	Senegalia	Serjania	Syagrus	Tridax	Urochloa
Anadenanthera	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Arecaceae	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	1
Arrabidaea	1	0	27	0	0	0	0	5	0	0	0	0	0	0	0	0	1	1	2	0	0	0	0
Cecropia	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chromolaena	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Combretum	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
Croton	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Dipteryx	0	1	2	0	0	0	0	23	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0
Eucalyptus	0	0	0	0	0	0	0	0	27	0	0	0	3	0	0	0	2	0	0	0	1	0	0
Faramea	0	0	0	0	0	0	0	0	0	28	0	0	1	0	3	1	0	2	0	0	2	0	1
Hyptis	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0
Mabea	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0
Matayba	0	0	0	0	0	0	0	0	3	0	0	0	24	0	0	0	0	0	0	0	0	0	0
Mimosa	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0
Myrcia	0	1	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	1	0	1	1	0	1
Protium	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	1	0	0	0	0	1
Qualea	0	1	1	0	0	0	0	2	0	1	0	0	2	0	0	0	27	0	0	0	0	0	0
Schinus	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	23	0	0	0	0	0
Senegalia	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0	0
Serjania	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0
Syagrus	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	0
Tridax	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0
Urochloa	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	26

Figure 30: Confusion matrix for the validation results of setup A (feature extraction and linear discriminant).



	Anadenanthera	Arecaceae	Arrabidaea	Cecropia	Chromolaena	Combretum	Croton	Dipteryx	Eucalyptus	Faramea	Hyptis	Mabea	Matayba	Mimosa	Myrcia	Protium	Qualea	Schinus	Senegalia	Serjania	Syagrus	Tridax	Urochloa
Anadenanthera	29	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Arecaceae	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	2
Arrabidaea	0	0	26	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cecropia	0	0	1	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chromolaena	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Combretum	0	0	0	0	0	30	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Croton	0	0	0	0	0	0	28	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Dipteryx	0	0	1	0	0	0	0	27	0	1	0	0	0	0	0	0	3	0	1	0	0	0	0
Eucalyptus	0	0	0	0	0	0	0	0	26	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Faramea	0	0	0	0	0	0	0	0	0	24	0	0	3	0	0	1	1	1	0	0	0	0	0
Hyptis	0	0	0	0	0	0	0	0	0	0	28	1	0	0	0	0	0	0	0	0	0	0	0
Mabea	0	1	0	0	0	0	1	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0
Matayba	0	3	0	0	0	0	0	0	3	1	0	0	26	0	0	0	0	0	0	0	0	0	0
Mimosa	0	0	0	5	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0
Myrcia	0	1	0	0	0	0	0	0	1	1	0	0	0	0	27	1	0	2	0	0	0	0	1
Protium	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0	0	1	0	0	0
Qualea	0	0	1	0	0	0	1	0	0	1	2	0	0	0	0	0	26	0	0	0	0	0	0
Schinus	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	2	0	25	0	0	0	0	0
Senegalia	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	29	0	0	0	1
Serjania	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	1	0	29	0	0	0
Syagrus	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	1
Tridax	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0
Urochloa	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	25

Figure 31: Confusion matrix for the validation results of setup B (transfer learning).



	Anadenanthera	Arecaceae	Arrabidaea	Cecropia	Chromolaena	Combretum	Croton	Dipteryx	Eucalyptus	Faramea	Hyptis	Mabea	Matayba	Mimosa	Myrcia	Protium	Qualea	Schinus	Senegalia	Serjania	Syagrus	Tridax	Urochloa
Anadenanthera	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Arecaceae	0	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	3
Arrabidaea	2	0	29	0	0	0	0	1	0	0	0	0	2	0	0	0	0	0	1	0	0	0	0
Cecropia	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Chromolaena	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Combretum	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Croton	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Dipteryx	1	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	2	0	1	0	0	0	0
Eucalyptus	0	0	0	0	0	0	0	0	28	0	0	0	4	0	0	0	0	0	0	0	0	0	0
Faramea	0	0	0	0	0	0	0	0	0	29	0	0	0	0	3	0	0	2	0	0	0	0	0
Hyptis	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0	0	0	0
Mabea	0	0	0	0	0	0	0	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	0
Matayba	0	0	0	0	0	0	0	0	2	0	0	0	24	0	0	0	0	0	0	0	0	0	0
Mimosa	0	0	0	1	0	0	0	0	0	0	0	0	0	30	0	0	0	0	0	0	0	0	0
Myrcia	0	2	0	0	0	0	0	0	0	0	0	0	0	0	24	2	0	2	0	0	0	0	0
Protium	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	25	0	0	0	1	0	0	0
Qualea	0	0	1	0	0	0	0	1	0	1	0	1	0	0	0	0	28	0	0	0	0	0	0
Schinus	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	26	0	0	0	0	0
Senegalia	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0	0
Serjania	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	29	0	0	0
Syagrus	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	1
Tridax	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30	0
Urochloa	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	26

Figure 32: Confusion matrix for the validation results of setup C (transfer learning, feature extraction and linear discriminant).

6 Discussion

In this paper, we present three deep learning classifiers of pollen grain images, based on a pre-trained convolutional neuronal network (Alexnet) to which we apply transfer learning and extract features that are subsequently classified by a linear discriminant classifier.

Any of the three hereby proposed solutions considerably improve the results presented in [3] for the same POLEN23E dataset. In that work, researchers obtain the best results by applying a technique known as bag-of-words (BOW) as feature extractor. However, the results obtained by this technique are poor, with a median CCR around 66% for the best non-human classifier. Humans reached a 67% median CCR in their experiment.

To explain these results, the nature of the POLLEN23E images must be considered. These images are centered on the frame and occupy most of it. In addition, there are similarities between different classes that make the classification process difficult by itself. The BOW technique is used to classify images but is generally better suited for images that contain different features and are not centered inside the frame (landscapes or objects of a very diverse nature). This has to do with the fact that BOW is based on a k-nearest neighbor algorithm, which makes it outstand, for example, in handwritten digit recognition because the space of digit images (and thus the possible variation amongst them) is much smaller than the space of all possible images.

However, for the POLEN23E dataset, features are spread almost uniformly throughout the feature space, not concentrated on or near any lower dimensional manifold. Hence, given the high similarity of the images, many of the features that the k-nearest neighbor algorithm must classify will be very similar. On the other hand, BOW extracts by default 500 “words”, generating a high dimensionality problem for the classifier. Considering that many of these “words” are similar because of the images similarity, even amongst different classes, it incurs in what is known as the curse of dimensionality. As mentioned in [22], with this setup, in high dimensions all examples look alike, implying that the choice of the nearest neighbor (and therefore the class) tends to be random.

The solution proposed in this paper is based on deep learning convolutional filters. These filters are capable of extracting specific and discriminating features of each image to work properly with a simple classifier in high dimensions. The way in which these filters extract the image features makes possible to obtain the most relevant characteristics to each class, avoiding those common to several classes that introduce noise in the classifier. Therefore, the instances (features) are sufficiently distant between their corresponding classes for the classifier to correctly identify them.

Fig. 33 shows the cross-validation average CCR obtained by setup C for each class. We note that there are three classes (*Matayba guianensis*, *Myrcia* and *Schinus terebinthifolius*) which registered a low CCR value around 33% in some of the folds, while the same classes reach CCR values of 100% in other folds. This happens because the number of observations contained in the POLEN23E set is small, making the results of each fold very sensitive to the training/validation split of the dataset. This effect would be diminished by increasing the number of observations per class, allowing to obtain a greater number of features through convolutional filters that would help to identify each class with more accuracy. It is also remarkable that the proposed setup manages to identify 100% of the examples in almost all of the folds for two thirds of the available classes. For these fifteen classes (*Arrabidaea*, *Cecropia pachystachya*, *Chromolaena laevigata*, *Combretum discolor*, *Croton urucurana*, *Dipteryx alata*, *Eucalyptus*, *Faramea*, *Hyptis*, *Mabea fistulifera*, *Mimosa somnians*, *Qualea multiflora*, *Senegalia plumosa*, *Serjania laruotteana* and *Tridax procumbens*) the first and third quartiles are equal to the median, which is set at 100%. They are thus perfectly identified by the model, which is able to fully aprehend the particular characteristics of each class.

In a similar take, in [13] an approach based on pre-trained CNN transfer learning is presented, obtaining results around 94% of CCR for data sets obtained by light-microscopy and scanning electron microscopy. However, they do not specify if the results are obtained over the training set or the validation set, and hence they are not directly comparable with ours. They also use data augmentation to increase the number of samples for their datasets, a technique that could probably improve our results as well.

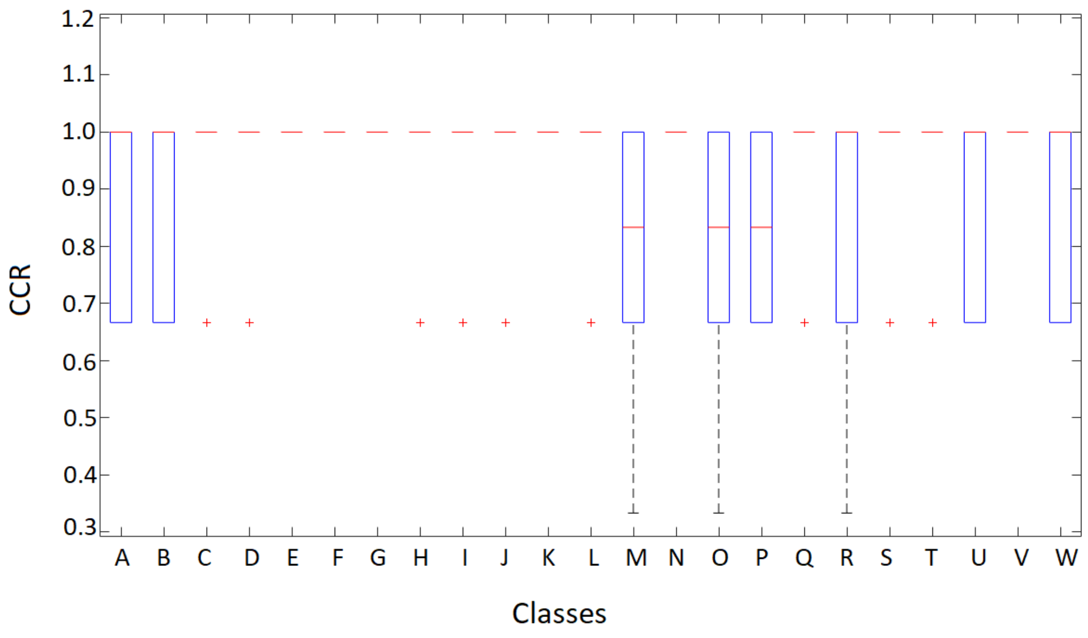


Figure 33: CCR obtained with setup C for the different classes: A) *Anadenanthera colubrina*, B) *Arecaceae*, C) *Arrabidaea*, D) *Cecropia pachystachya*, E) *Chromolaena laevigata*, F) *Combretum discolor*, G) *Croton urucurana*, H) *Dipteryx alata*, I) *Eucalyptus*, J) *Faramea*, K) *Hyptis*, L) *Mabea fistulifera*, M) *Matayba guianensis*, N) *Mimosa somnians*, O) *Myrcia*, P) *Protium heptaphyllum*, Q) *Qualea multiflora*, R) *Schinus terebinthifolius*, S) *Senegalia plumosa*, T) *Serjania laruotteana*, U) *Syagrus*, V) *Tridax procumbens*, W) *Urochloa decumbens*.

7 Conclusions and future work

In this work, three different models based on deep learning convolutional neural networks and capable of satisfactorily classifying the POLEN23E dataset images have been presented. The three solutions have exceeded the 95% threshold of correct classifications. Amongst them, the most accurate is obtained after applying transfer learning to the AlexNet pre-trained network and then using a linear discriminant classifier to the extracted features. This solution perfectly classifies two thirds of the pollen types, reaching up to a global 97% of correctly classified samples. Finding an automated solution to pollen classification implies significant potential advantages in different areas, including allergy-related clinical applications or industries like honey production.

These excellent results prompt us to try the same ideas with different datasets, but also different pre-trained networks and also networks trained from scratch. Expanding the number of images available through data-augmentation also looks promising in order to increase the classification capacity of the proposed solutions.

8 Supporting Information

8.1 S1 DataSet

POLEN23E (http://dx.doi.org/10.6084/m9.figshare.1525086).

References

1. American College of Allergy, Asthma, and Immunology. The Rise Of Spring Allergies: Fact Or Fiction? | ACAAI Public Website; 2017. website. Available from: `http://acaai.org/news/rise-spring-allergies-fact-or-fiction`.

2. Holt KA, Bennett KD. Principles and methods for automated palynology. New Phytologist. 2014;203(3):735–742. Available from: `http://onlinelibrary.wiley.com/doi/10.1111/nph.12848/abstract`.

3. Gonalves AB, Souza JS, Silva GGd, Cereda MP, Pott A, Naka MH, et al. Feature Extraction and Machine Learning for the Classification of Brazilian Savannah Pollen Grains. PLOS ONE. 2016;11(6):e0157044. Available from: `http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0157044`.

4. Treloar WJ, Taylor GE, Flenley JR. Towards automation of palynology 1: analysis of pollen shape and ornamentation using simple geometric measures, derived from scanning electron microscope images. Journal of Quaternary Science. 2004;19(8):745–754. Available from: `http://dx.doi.org/10.1002/jqs.871`.

5. Garca NM, Chaves VAE, Briceo JC, Travieso CM. In: Corchado E, Snel V, Abraham A, Woniak M, Graa M, Cho SB, editors. Pollen Grains Contour Analysis on Verification Approach. Springer Berlin Heidelberg; 2012. p. 521–532. DOI: 10.1007/978-3-642-28942-2_47. Available from: `https://doi.org/10.1007/978-3-642-28942-2_47`.

6. Travieso CM, Briceo JC, Ticay-Rivas JR, Alonso JB. Pollen classification based on contour features. In: 2011 15th IEEE International Conference on Intelligent Engineering Systems; 2011. p. 17–21. Available from: `http://ieeexplore.ieee.org/document/5954712/?reload=true`.

7. Fernndez-Delgado M, Carrin P, Cernadas E, Galvez JF. Improved Classification of Pollen Texture Images Using SVM and MLP. In: 3rd IASTED International Conference on Visualization, Imaging and Image Processing (VIIP2003). vol. 2; 2003. Available from: `https://www.researchgate.net/publication/228441194_Improved_Classification_of_Pollen_Texture_Images_Using_SVM_and_MLP`.

8. Silva DSd, Quinta LNB, Gonalves AB, Pistori H, Borth MR. Application of wavelet transform in the classification of pollen grains. African Journal of Agricultural Research;9(10):908–913. Available from: `http://www.academicjournals.org/journal/AJAR/article-abstract/F01A46046172`.

9. Rodriguez-Damian M, Cernadas E, Formella A, Fernandez-Delgado M, Sa-Otero PD. Automatic detection and classification of grains of pollen based on shape and texture. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews);36(4):531–542. Available from: `http://ieeexplore.ieee.org/document/1643845/authors`.

10. Andrade W, Quinta L, Gonalves A, Cereda MP, Pistori H. Segmentação baseada em Textura e Watershed aplicada a Imagens de Pólen. In: SIBGRAPI 2012 - Workshop of Undergraduate Work (WUW). Ouro Preto, MG; 2012. .

11. Ticay-Rivas JR, Pozo-Baos Md, Travieso CM, Arroyo-Hernndez J, Prez ST, Alonso JB, et al. In: Pollen Classification Based on Geometrical, Descriptors and Colour Features Using Decorrelation Stretching Method. IFIP Advances in Information and Communication Technology. Springer, Berlin, Heidelberg;. p. 342–349. DOI: 10.1007/978-3-642-23960-1_41. Available from: `https://link.springer.com/chapter/10.1007/978-3-642-23960-1_41`.

12. Chica M. Authentication of bee pollen grains in bright-field microscopy by combining one-class classification techniques and image processing. Microscopy Research and Technique;75(11):1475–1485. Available from: `http://onlinelibrary.wiley.com/doi/10.1002/jemt.22091/abstract`.

13. Daood A, Ribeiro E, Bush M. Pollen Grain Recognition Using Deep Learning. In: Bebis G, Boyle R, Parvin B, Koracin D, Porikli F, Skaff S, et al., editors. Advances in Visual Computing. vol. 10072. Cham: Springer International Publishing; 2016. p. 321–330. DOI: 10.1007/978-3-319-50835-1_30. Available from: `http://link.springer.com/10.1007/978-3-319-50835-1_30`.

14. France I, Duller AWG, Duller GAT, Lamb HF. A new approach to automated pollen analysis. Quaternary Science Reviews;19(6):537–546. Available from: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.4.5503&rep=rep1&type=pdf`.

15. Gonalves AB, Rodrigues CNM, Cereda MP, Hemerson P. Identificação computadorizada de tipos polínicos através de ”Bag of Words”. Cadernos de Agroecologia. 2013;8(2). Available from: http://www.aba-agroecologia.org.br/revistas/index.php/cad/article/download/14634/9554.

16. Lozano-Vega G, Benezeth Y, Marzani F, Boochs F. In: Iliadis L, Maglogiannis I, Papadopoulos H, editors. Analysis of Relevant Features for Pollen Classification. Berlin, Heidelberg: Springer Berlin Heidelberg; 2014. p. 395–404. Available from: https://doi.org/10.1007/978-3-662-44654-6_39.

17. Fukushima K. Neocognitron. Scholarpedia. 2007;2(1):1717. Revision #91558. Available from: http://www.scholarpedia.org/article/Neocognitron.

18. LeCun Y, Boser BE, Denker JS, Henderson D, Howard RE, Hubbard WE, et al. Handwritten Digit Recognition with a Back-Propagation Network. In: Touretzky DS, editor. Advances in Neural Information Processing Systems 2. Morgan-Kaufmann; 1990. p. 396–404. Available from: http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf.

19. Ciregan D, Meier U, Schmidhuber J. Multi-column deep neural networks for image classification. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition; 2012. p. 3642–3649.

20. Aghdam HH, Heravi EJ. Guide to convolutional neural networks: a practical application to traffic-sign detection and classification. Springer International Publishing; 2017. Available from: http://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=1520793.

21. MathWorks. Introducing Deep Learning with MATLAB;. Available from: https://www.mathworks.com/content/dam/mathworks/tag-team/Objects/d/80879v00_Deep_Learning_ebook.pdf.

22. Domingos P. A Few Useful Things to Know About Machine Learning. Commun ACM. 2012 Oct;55(10):78–87. Available from: http://doi.acm.org/10.1145/2347736.2347755.

9 Appendix

9.1 PLOS One Article Publication

PLOSONE (https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0201807).

9.2 PMC Article Publication

PMC (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6138340/).