Deep convolutional neural networks for taxonomic classification from images

Calo Oliveira, Sergio

Facultade de Física USC, Santiago de Compostela, España

E-mail: `sergio.calo@rai.usc.es`

May, 2020

Abstract. In this work, we will try to approach the taxonomic classification problem using computer vision and deep learning techniques. Convolutional neural networks proved to be exceptional when it comes to imaging. Therefore, we will use diferent pre-trained convolutional models and discuss which of them yields the best results on a multi-class database built as part of the project. Finally, the model will be implemented in a web application through the client-server architecture.

Keywords: Deep learning, Computer vision, CNN, image classification, ResNet, Web application.

1. Introduction and objectives

Biodiversity is increasingly at the center of debates about the environment and its conservation. In this work we focus on solving an image classification problem applied to the taxonomy of species that we can find in the territory of Galicia (Spain). For this goal we will use a convolutional neural network that, after being trained with a sufficient number of images of each taxon, will return the class to which the living being that appears in the image we use as input belongs.

In addition to the above, the second objective of the project is to provide citizens possibility to use techniques based on machine learning in a simple way. For this, a web application was designed, where the convolutional neural network classification was implemented. The user will be able, through this application, to classify any image, as long as it is a species already known by the network, obtaining as a result its taxonomic classification together with valuable information about it (habitat, state of conservation, etc.). We hope that, in addition, the use of this application increases curiosity about a vital topic like this, providing a fun, interactive and educational way of knowing the user's natural environment.

2. The dataset

To obtain these images, the iNaturalist [1] database was used. This database is the result of a citizen scientific effort, where

Table 1. Dataset details

Taxon	Train	Test	Total	Imbalance
Acacia dealbata	99	56	155	1.303
Acacia melanoxylon	103	66	169	1.195
Apis melifera	120	82	202	1
Vespa velutina	126	71	197	1.025
Formicidae	123	70	193	1.047
Total	571	345	916	

users share their observations on biodiversity around the world, as well as their location and date. The observations are verified by the scientific community, awarding the label "research degree" when certain criteria are met, including a consensus in support of the validity by members of the taxon of such observation, the provision of visual or sound evidence and geolocation [2] This database contains a total of 46.5 million observations, of which 24.3 million meet the requirements to be considered suitable for scientific research, distributed among tens of thousands of different species. 5 of them were chosen with special relevance in our territory (*Acacia dealbata, Acacia melanoxylon, Vespa velutina, Apis melífera* and the genus *Formica*). Within these 5 classes, and making a manual selection from those that have a Attribution-Noncommercial license, the Dataset was built with those that were considered most useful for network training, discarding those of dubious graphic quality.

To solve as much as possible the problem of image sample size, we will use the data augmentation technique, widely used in the field of deep learning, with very good results [3]. In this case we will use two different types of transformations: change the size of the cutout and horizontal flipping. These transformations are applied only to those images dedicated to training, and not to those intended for validation.

3. The model

To approach this problem, we will make use of pretrained models [4, 5]. The pytorch library offers a number of pre-trained networks. We will compare in this section only those architectures that fit especially well to our case, although there are many more available to the user.

The models we will compare are: Resnet18, resnet34, resnet50 [7], alexnet [8].

To train the model we will be using stochastic gradient descent (SGD) with a mini-batch of 4 images. In addition, we make use of a learning rate scheduler, wich decays the learning rate of each parameter group by gamma every step_size epochs, with a multiplicative factor of learning rate decay γ of 0.1 and a *step_size* of 7.

After studying the candidate networks, we observed the highest success rate presented corresponds to ResNet50. This network, being deeper, has a computation time and a weight slightly higher than the others (13.91% slower than ResNet34, the

Table 2. Comparison of the models (Accuracy (%), Tamaño (MB), Parameters (total number of trainable parameters), t (time per epoch in seconds))

Model	Accuracy	Size	Parameters	t CPU	t GPU
ResNet18	84.71	44.7	11179077	202.33	11.675
ResNet34	88.00	83.3	21287237	503.40	13.475
ResNet50	88.86	97.8	23518277	705.13	15.35
AlexNet	76.00	233	61105845	140.66	14.96

second with the highest success rate, per season). However, we consider this difference in computation time to be acceptable, prioritizing the 0.8571 extra percentage points of hits offered by ResNet50. Therefore, this architecture will be used to solve the proposed goal. As already mentioned, we will initialize the weights of this network with the values obtained after the unsupervised pre-training on the ImageNet database.

4. Results and Discussion

The first aspect that was studied was to establish the optimal learning coefficient and momentum. For this, trainings were performed for different values of the learning rate keeping the momentum constant. The equivalent was done for the momentum case, this time keeping the learning rate constant. The results obtained are presented in Figure 1. The best results were obtained using the values $\alpha = 0.0005$ and $m = 0.99$. For this combination, we will set out in more detail the statistical estimates of network efficiency below (*Table 3*)

4.1. Confidence threshold

In view of the values thrown by the *fall out* we are faced with a ratio of false positives that is around 3 percentage points. However, we consider this value not good enough as would be desirable, as each false positive would involve providing false information to the user. For this reason, we believe it is necessary to establish a confidence threshold that can assure us a lower number of false positives, although this leads to having to dismiss certain predictions that do not meet this quality standard. To set this threshold, the values of κ, *accuracy* and the discarded prediction rate against the confidence threshold were compared (See annex, *Figure 2*). In view of these graphs we can see how the increase in the threshold, as expected, implies an improvement in the predictions given by the network. We also observe small fluctuations that are due solely to the reduction in the size of the validation sample as this threshold is increased. We can now set for this parameter the value we consider most appropriate based on our preferences. For this particular case, it was considered best to opt for a high threshold that minimizes false positives. We can observe in *Figure 2.b* how there is a clear change in slope in the growth of the discard rate from approximately threshold = 0.94. It was considered unprofitable to continue raising the threshold from this point, so it was decided to take this value as the optimal one.

The estimators of the quality of the predictions obtained for this case are shown in *Table 4*. This implies that only those predictions for which the model establishes a probability equal to or greater than 94% will be taken as valid.

Table 3. Model estimators without threshold

	A. dealbata	A. melanoxylon	Apis melifera	Formicidae	V. velutina
Fall out	0.0277	0.0387	0.0149	0.0321	0.0251
Sensitivity	0.9180	0.9242	0.8537	0.8857	0.8732
Specificity	0.9723	0.9613	0.9851	0.9679	0.9749
		κ	0.8606		

Table 4. Model estimators with a threshold of 0.94

	A. dealbata	A. melanoxylon	Apis melifera	Formicidae	V. velutina
Fall out	0.0153	0.00535	0.00529	0	0
Sensitivity	0.9778	0.9444	1	0.9811	1
Specificity	0.9847	0.9947	0.9947	1	1
		κ	0.9740		

5. Web application

To facilitate the use of this classification algorithm and bring it closer to all those potential users who are not familiar with programming languages, we will implement this tool in an interface that allows the less experienced user to classify any image presented as input in a simple and interactive way. For this, a web application has been developed that will meet this goal.

To fulfill this purpose use was made of the client-server architecture. In this model, the client (user) sends a request to the server (where the neural network is already trained and ready) along with the information it needs as input, in this case, an image. After this, the algorithm is responsible for classifying such an image and returning to the client the calculated output (the class).

We can divide the process followed in the development of the web application into three different phases: server creation, model implementation, and web page development.

In the server creation phase, it was developed from the *framework* 'Express', which is included in the Node js [10] environment. This environment, based on the JavaScript language and event-oriented, provides the tools needed to create a web application. The function of the server will be to act as a support to meet user requests, receive and send data.

In the second phase, the implementation phase, the trained model was incorporated into the server. To achieve this, a program was designed, written in the python language, which received orders to: load the model from where it is stored, receive as input the path of the image to be classified, convert that image so that it can be used by the network, make classification and return the prediction. This program will work hand in hand with the aforementioned server, which will allow you to receive and send the data.

In turn, a third piece is needed, the website. This will act as an interface, allowing the user, among other things, to send the data required by the model easily. This website was written in the html programming language, developed in the early 1990s at CERN.

After the prediction, the user is redirected to a subsection where relevant information about that taxon is exposed.

6. Conclusion

In this work, a convolutional neural network model was built, based on the ResNet architecture and with deep learning techniques, with the aim of the taxonomic classification of species present in Galicia. For the database used as a sample, an initial success rate of 88.86 % was obtained, corresponding to $\kappa = 0.8606$. Establishing a confidence threshold in the probability of network output has been shown to significantly improve outcomes, while increasing the number of discarded predictions. It was considered appropriate to set a confidence threshold of 0.94, after which the κ obtained became 0.9740.

In addition, we found that the ResNet50 architecture was slightly above in terms of performance of other deep convolutional networks for training with our database. This database gathered a total of 916 images, divided into 5 different classes.

Subsequently, a web application was developed where this model was integrated. This application allows the user to perform identifications easily from the path of the image that is previously stored on their computer.

7. Proposals for the future of the project

After verifying in this work that it is possible to make a taxon identification between as many classes as possible, the next goal is to continue developing the project until reach an attractive web application that can be used by anyone through the network and that offers new possibilities. The model can be improved as well, using the architecture *Faster R-CNN* to obviate background, and develop a multi-level algorithm to fix better to taxonomic tree multi-level problem.

To achieve this goal, a database containing a significant number of different species must be developed. The procedure for training in this case will be very similar to that followed in this work, although the values of κ, hit rate and confidence threshold will probably change significantly.

We can also set ourselves challenges that go beyond simple identification. We will try to develop a storage system that takes into account the predictions that the model makes of the images entered by users, so that they can analyze them, contrast them with the opinion of an expert and that they can be used in future studies that have to do with the fauna and flora of Galicia.

In short, the ultimate goal of the project will be to build a reliable identifier for the fauna and flora of Galicia, integrated in an intuitive and comfortable web application. Along with this, to encourage the construction of a community of biodiversity-loving users, which can help to spread and increase the knowledge that exists about our natural heritage.

[1] iNaturalist. 2020. Inaturalist. [online] Available at: ¡https://www.inaturalist.org/¿ [Accessed 8 June 2020].

[2] Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., ... Belongie, S. (2018). The inaturalist species classification and detection dataset. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 8769-8778).

[3] Wang, J., Perez, L. (2017). The effectiveness of data augmentation in image classification using deep learning. Convolutional Neural Networks Vis. Recognit, 11.

[4] Erhan, D., Bengio, Y., Courville, A., Manzagol, P. A., Vincent, P., Bengio, S. (2010). Why does unsupervised pre-training help deep learning?. Journal of Machine Learning Research, 11(Feb), 625-660.

[5] Erhan, D., Manzagol, P. A., Bengio, Y., Bengio, S., Vincent, P. (2009, April). The difficulty of training deep architectures and the effect of unsupervised pre-training. In Artificial Intelligence and Statistics (pp. 153-160).

[6] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). Ieee.

[7] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[8] Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[9] Olga Russakovsky et al. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV), 115(3):211–252, 2015. 20(1), 37–46 (1960)

[10] Node.js. http://nodejs.org

Annex: Figures

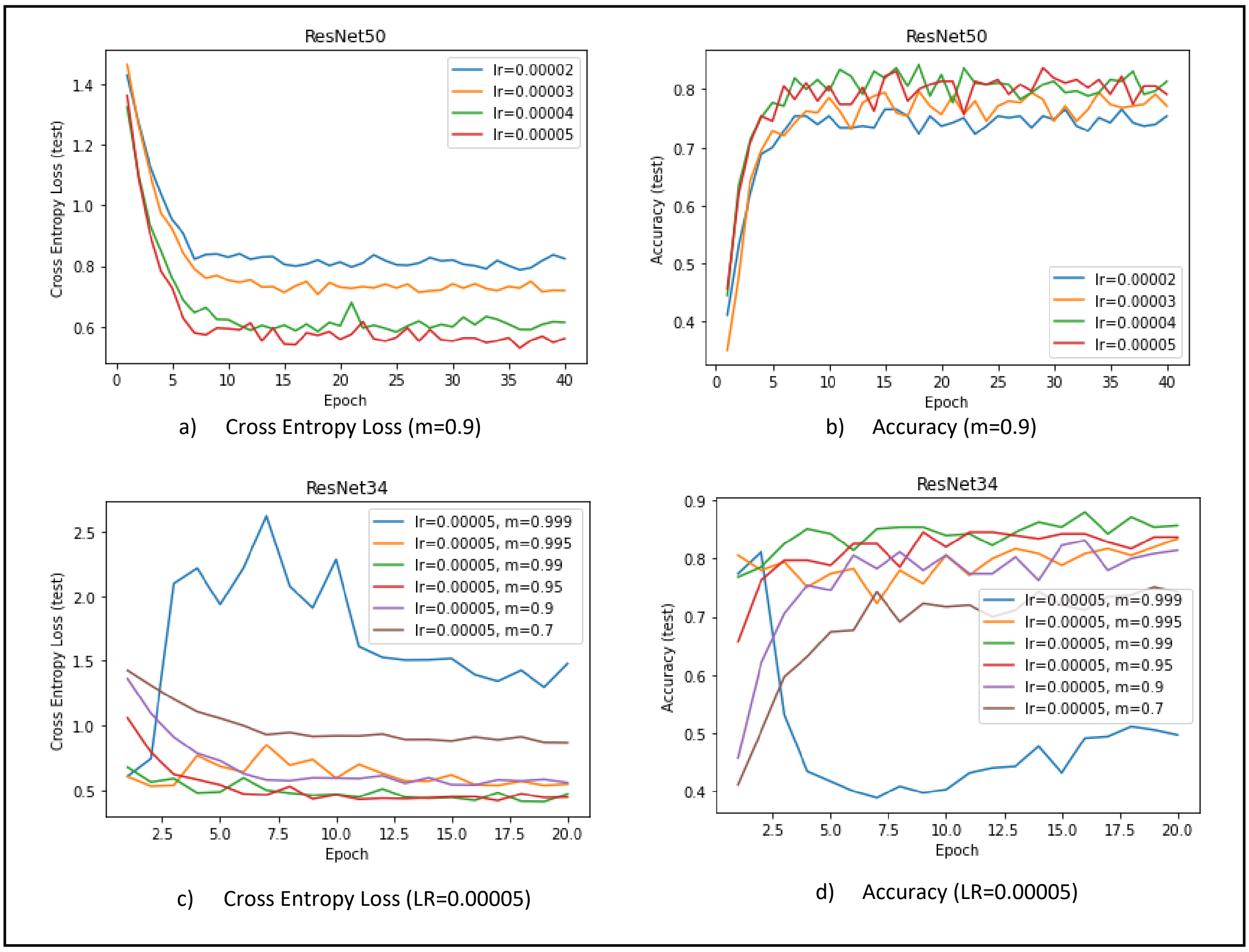


Figure 1: Accuracy and loss dependence with learning rate and momentum

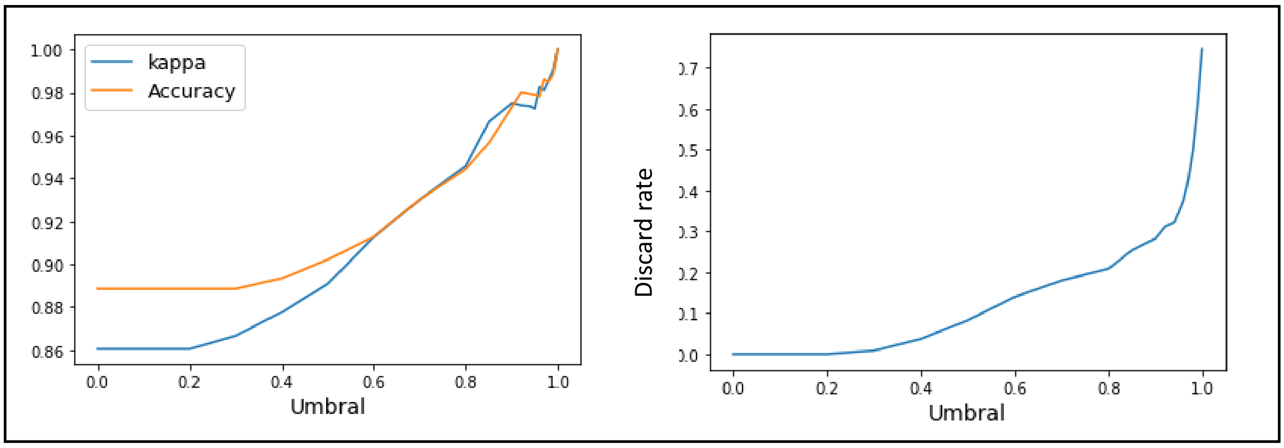
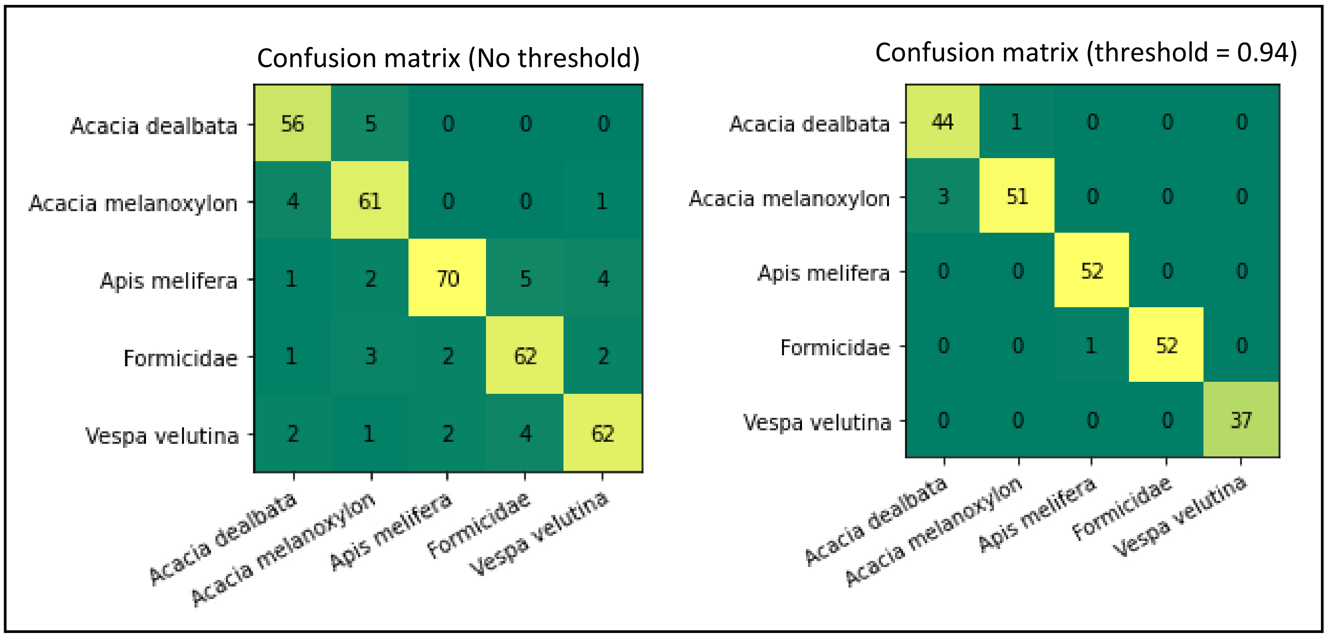


Figure 2: Threshold effect on model results



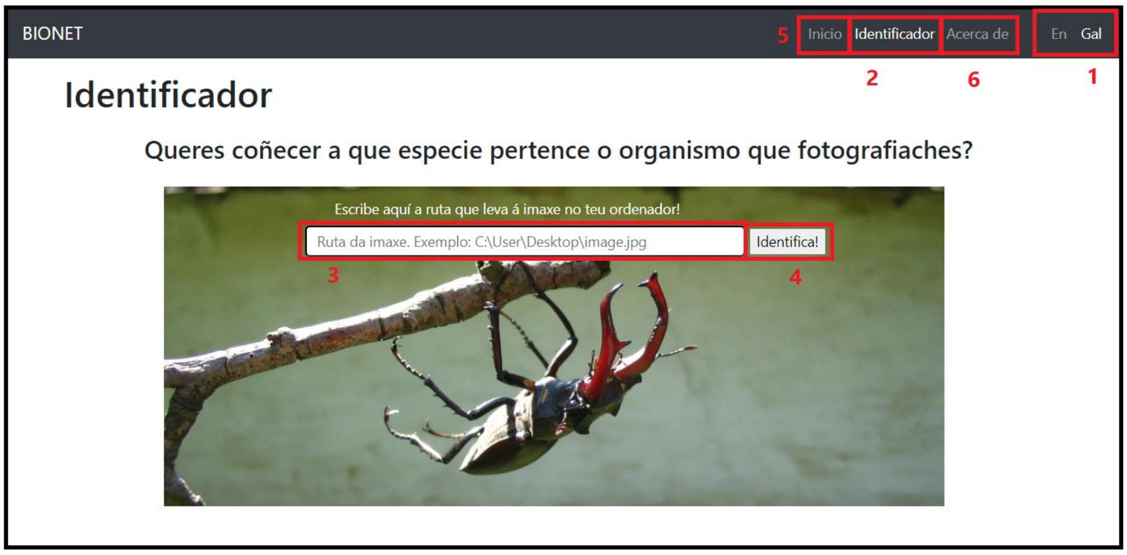
Confusion matrix (No threshold)

	Acacia dealbata	Acacia melanoxylon	Apis melifera	Formicidae	Vespa velutina
Acacia dealbata	56	5	0	0	0
Acacia melanoxylon	4	61	0	0	1
Apis melifera	1	2	70	5	4
Formicidae	1	3	2	62	2
Vespa velutina	2	1	2	4	62

Confusion matrix (threshold = 0.94)

	Acacia dealbata	Acacia melanoxylon	Apis melifera	Formicidae	Vespa velutina
Acacia dealbata	44	1	0	0	0
Acacia melanoxylon	3	51	0	0	0
Apis melifera	0	0	52	0	0
Formicidae	0	0	1	52	0
Vespa velutina	0	0	0	0	37

Figure 3: Confusion matrix



Identificador

Queres coñecer a que especie pertence o organismo que fotografiaches?

Escribe aquí a ruta que leva á imaxe no teu ordenador!

Figure 4: Web app screenshot (1: language selection, 2: classifier, 3: image path, 4: run button, 5: home page, 6: about page)