Use of Convolutional Neural Network for Wild Animal Classification

Alan Afif Helal
IT Department
Computer Engineering
Federal University of Espírito Santo
Vitória, Espírito Santo
Email: alan@helal.com.br

***Abstract*—The idea to build intelligent systems that model human behavior has been studied for over a century. Despite all the progress in the field, there is a long way to go in trying to model how the human brain works. This work aims to build a convolutional neural network for the classification of wild animals captured by trap cameras. The neural network architecture used was VGG16 with a data-set for training with 4,550 images of three classes of mammal species. Four different scenarios were tested for the best accuracy. After training the neural network was able to correctly classify 92.80% of the photos in the test data-set using Data Augmentation and a Dropout layer.**

I. Introduction

Mammals represent the largest-bodied elements of the world's surviving megafauna and provide key ecosystems services [1]. Indeed, they are considered umbrella species for conservation [2], and may encompass several taxonomic groups within their large home ranges when target of conservation strategies [3].

According to [1], Brazilian Atlantic Forest mammals are under high levels of threat due to historical population losses that continue today, demanding a conservation action plan to prevent this biome from becoming an even "emptier forest", severely compromising ecological processes and ecosystem functioning. Also, although clear signs of defaunation, the extent to which this biome has lost its mammal fauna remains poorly understood.

The understanding of mammal ecology has always been hindered by the difficulties of observing species in closed tropical forests, which can be partially solved using camera trapping [4]. This technology has become a major advance for monitoring terrestrial mammals in biodiversity rich ecosystems, because they enable collecting wildlife pictures inexpensively and frequently.

This project aims to solve a real-life problem. Trap cameras are placed around farms capturing images of local wildlife. They are activated when they detect movement, however, they are often activated due to wind, tree movement, or even falling fruits. Furthermore, one of the most recurrent problems faced by biologists and other scientists were the activity of checking one by one of the thousands of photos taken by trap cameras and classifying which animals appear.

In order to reduce manual processing time disposed by scientists to identify photos taken by trap cameras, we propose using convolutional neural network to classify three types of animals: armadillo, deer, and wild pig. For training and validation of the neural network, a data-set with more than 5,000 photos taken by trap cameras was used. Training requires computing power hardly found in personal computers in our homes. To solve this problem, Google Colab was used for training and validation.

It was proposed the creation of a neural network that would be trained with the photos taken by trap cameras to classify the animal species. In this way, automating the work of classifying animals performed by a human. This work was structured as follows:

- Section 2 explains the creation of the neural network, the handling of training and testing datasets, and the techniques used to improve the performance of the neural network.
- Section 3 shows the results found in each of the four analyzed scenarios using the network accuracy as a metric both in training and testing.
- Section 4 discuss the results and possible explanations for the problems encountered during the neural network training.
- Section 5 presents the conclusion and suggestions for future work.

II. Materials and Methods

For this work, more than 60,000 photos from 37 different trap cameras were made available. The cameras are located on farms in South Bahia, Brazil. The first problem encountered was that several photos had no animals or had people walking on the farm, for example, placing the trap cameras. Besides the fact that the photos take up to 62 GB of space on the hard drive.

A forestry engineer helped to catalog the photos according to the animal species. After the process of analyzing each of the 60,000 photos, 15 categories were obtained, namely: tapir, cutia, squirrel, opossum, wild cat, jacu, monkey, jaguar, paca, wild pig, anteater, armadillo, and deer.

Due to limitations in computational power and the need to have at least one thousand photos of each category to enable a good training of the neural network, the three species that had the most photos were selected: wild pig, deer, and armadillo.

As we are using the free version of Google Colab, we do not have access to all the computational power available by Google and faced a computational bottleneck. There was not enough memory to load all the images from the training data-set. The training data-set was reduced to 4,550 images, with 1600 armadillo images, 1,700 wild pig images, and 1,250 deer images. Thus, we had a considerable amount to carry out good neural network training.

The fact that they are photos of animals taken in the real world, they do not show a certain regularity concerning the position of the animal. Some photos have the animal completely from the side, others from the front, some only the head or only the hind legs. A good example is shown in Figure 1 where we have four different photos of the same animal showing different parts of its body.

Fig. 1. Four different positions of the same animal captured by the trap camera.

The VGG16[5] architecture, chosen for the creation of the neural network, is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet[1], which is a dataset of over 14 million images belonging to 1,000 classes. It consists of five convolutional blocks followed by a fully connected layer as shown in Figure 2.

Activation functions are used in neural networks to compute the weighted sum of input and biases, which is used to decide if a neuron can be fired or not. As shown in Figure 2, the VGG16 architecture uses two activation functions: ReLu in the convolution layer and fully connected layers, and softmax in the output layer.

The rectified linear unit (ReLU) activation function has been the most widely used activation function for deep learning applications. It is a faster activation function with better performance and generalization in deep learning[6]. The ReLu

[1]https://www.image-net.org

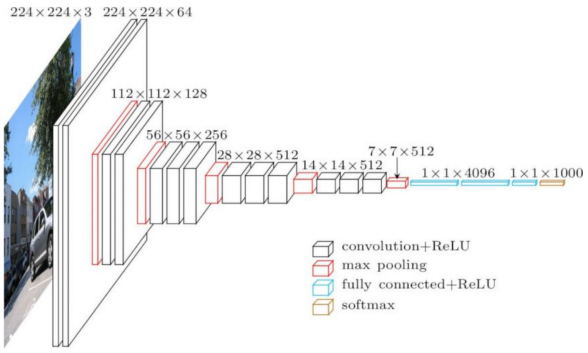

Fig. 2. VGG16 architecture.

activation function is given by Equation 1.

$$f(x) = max(0, x) = \begin{cases} x_i, \ if \ x_i \geq 0 \\ 0, \ if \ x_i < 0 \end{cases} \tag{1}$$

The Softmax function is an activation function used for multivariate classification tasks that compute the probability distribution from a vector of real numbers. The Softmax function produces an output which is a range of values between 0 and 1, with the sum of the probabilities been equal to 1[6]. The softmax activation function is showed in Equation 2.

$$f(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)} \tag{2}$$

Python[2] programming language with Keras[3], Tensorflow[4] and Numpy[5] libraries were used to program the neural network. Since we need a lot o computational power, we used the Google Colaboratory[6] platform. That way, we do not need to set up an environment for the training.

It is known that we can use GPU[7] as a hardware accelerator to perform faster training. Unfortunately, not everyone has access to High-End GPUs at home. But this issue has been worked around through Google Colab. For a few limited time, for free, we chose GPU as our hardware accelerator to perform the neural network training.

A. Batch Normalization

Training a neural network is a very expensive and complicated task. For this, several techniques make this task less complicated. Batch normalization[8] allows you to reach training thresholds with fewer training steps. Therefore, this technique was used to reduce time and computational cost in training.

[2]https://www.python.org
[3]https://keras.io
[4]https://www.tensorflow.org
[5]https://numpy.org
[6]https://colab.research.google.com

B. Data Augmentation

Fortunately, the training data-set can be easily expanded using the Data Augmentation[9] technique. With it, new images created digitally just by rotating the original image horizontally can be introduced in training. These digitally created images closely match the photos in the data-set, as we have several photos with the same animal walking around the camera. The use of Data Augmentation is an effective technique to considerably improve the performance of convolutional neural network training. Figure 3 shows an example of how Data Augmentation works when applied to one of the images from the training data-set.

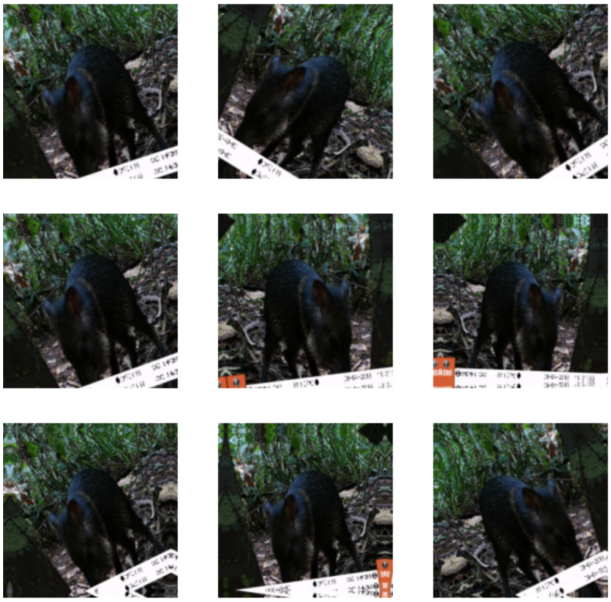

Fig. 3. Example of a digitally created image from the training data-set

C. The Dropout Dilemma

The dropout technique is a simple way to avoid neural network overfitting after the training. It improves the performance of the neural network training in some cases[10]. On the other hand, the use of Dropout on Convolutional Neural Networks fails to obtain a noticeable performance improvement[11]. The effectiveness of dropout for convolutional neural networks is further reduced[11] if you use other regularization techniques such as the one discussed in the previous subsections on this paper. It is not a general rule, but in our tests, the use of the dropout technique resulted in a simple improvement in the accuracy of the network. Therefore, it was decided to keep a dropout layer between the first and second dense layer.

D. Configuring the data-set for Performance

Since we are dealing with a huge amount of data, we need to configure the data-set for performance. We applied buffered prefetching to avoid I/O blocking when loading data. For this, we used two methods defined in the Keras library: cache() and prefetch(). The cache() method keeps the images in memory after they're loaded off disk during the first epoch of training. This technique ensures the data-set does not become a bottleneck while training the model. Given that the dataset is too large to fit into memory, this method is used to create a performant on-disk cache. The prefetch() method overlaps data preprocessing and model execution while training.

E. Optimizer and Loss Function

Before the model compilation, we need to choose the optimizer and the loss function. The Adam optimizer is often used due to its computational efficiency and little memory requirements. Additionally, it is well suited for problems that are large in terms of data and/or parameters[12]. Since we have three classes, the loss function used is the Sparse Categorical Cross Entropy.

F. Training and Validation

For training, the data-set was divided between training and validation. We used 80% of the images in the training data-set to train the neural network, i.e. 3640 images, and 20% for validation, i.e. 910 images. The training consisted of 10 epochs of learning.

G. The Neural Network

If you wish to use the neural network presented in this paper, there is a step-by-step guide provided in the author's GitHub[7].

III. Results

Four scenarios were tested to decide which techniques would be used for the neural network training:

- Scenario 1: No Data Augmentation and no Dropout layer
- Scenario 2: No Data Augmentation with a Dropout layer
- Scenario 3: Data Augmentation and no Dropout layer
- Scenario 4: Data Augmentation with a Dropout layer

The main idea was to verify whether the inclusion of these techniques would improve the accuracy of the trained neural network. For each of the scenarios described above, the neural network was trained from scratch. As we were looking for the configuration that would result in more correct answers in the classification of images, the accuracy of the network in training and validation was analyzed. To test the accuracy of the neural network, a data-set containing 570 images of the 3 classes (wild pig, armadillo, and deer) was used. The test data-set was divided as follows:

- 248 armadillo images
- 184 wild pig images
- 154 deer images

A. Scenario 1

This scenario had the worst accuracy among all. After the 10 training epochs, the neural network had 81.12% accuracy with 79.66% accuracy on validation. The results after the network test are shown in Table I. Despite the good accuracy in detecting armadillos, the neural network was deficient in detecting deer.

[7]https://github.com/HelalBR/Deep-Learning-2020-1/tree/main/Alan%20Helal

	Armadillo	Wild Pig	Deer	Accuracy
Armadillo	227	0	21	91,15%
Wild Pig	2	152	30	82,60%
Deer	4	70	80	51,19%

TABLE I: Confusion Matrix of test results in scenario 1.

B. Scenario 2

After introducing a Dropout layer, the accuracy of the network showed a slight improvement. This neural network had an accuracy of 83.47% with a validation accuracy of 82.01%. The results of the neural network test are shown in Table II, where it can be seen that the neural network still had difficulty in distinguishing deer.

	Armadillo	Wild Pig	Deer	Accuracy
Armadillo	230	0	18	92,74%
Wild Pig	1	158	25	85,86%
Deer	15	59	82	53,24%

TABLE II: Confusion Matrix of test results in scenario 2.

C. Scenario 3

With the inclusion of Data Augmentation in the training data-set, the network had seen a significant improvement. After training, the neural network presented an accuracy of 90.28% with a validation accuracy of 91.15%. Table III shows the results of the neural network test with an improvement in deer detection after the inclusion of the Data Augmentation technique.

	Armadillo	Wild Pig	Deer	Accuracy
Armadillo	235	3	10	94,75%
Wild Pig	5	172	11	93,47%
Deer	13	21	120	77,92%

TABLE III: Confusion Matrix of test results in scenario 3.

D. Scenario 4

This was the configuration that resulted in the best network performance. After using the Dropout layer and Data Augmentation in the training data-set, the neural network had an accuracy of 92.80% and a validation accuracy of 91.65%. Table IV shows the results after the network tests with satisfactory accuracy for each of the trained classes.

	Armadillo	Wild Pig	Deer	Accuracy
Armadillo	241	7	0	97,18%
Wild Pig	0	184	0	100,00%
Deer	14	14	126	81,28%

TABLE IV: Confusion Matrix of test results in scenario 4.

IV. Discussion

The neural network that showed the best result was the one using Data Augmentation in the training data-set and a Dropout layer in the dense layers. As expected, the use of the Dropout layer showed a small improvement in the accuracy of the neural network, probably because other techniques are already being used to improve network performance, such as Batch Normalization and Data Augmentation.

The low accuracy in detecting deer was probably because the data-set images are quite heterogeneous, with different parts of the animal appearing in the scenes. Due to the characteristics of the deer, depending on its position in the camera, we have completely different perceptions of the same animal. Such a situation can be easy to identify for a human but for a neural network it would need tens of thousands of photos to learn what a deer is with its various possible representations.

In an attempt to keep the training dataset and test more trustworthy in reality, neither a treatment nor a selection of the best images was carried out. Thus, the neural network was trained with images that may have influenced the low accuracy or difficulty in detecting animals. Even images that do not have the entire animal, images that are blurry, or images affected by a sunbeam were used in training and testing, since it was possible to identify the specie. The reason for keeping these images is because, in practice, such images are analyzed by experts to be categorized. Thus, it was expected that the neural network would be able to identify a wild pig even if only its head appeared, for example.

V. Conclusion

We conclude that deep learning could enable the inexpensive, high-volume, and even real-time collection of different animal species in the wild. During its elaboration, it was possible to experience several difficulties that could be found when we leave a controlled environment and go to the real world. The neural network with VGG16 architecture proved to be a good convolutional neural network for image classification, delivering an acceptable accuracy.

A. Future Works

For future work, it is suggested to use a training data-set with more photos and/or without photos that do not present a good representation of the animals. You can also add more animal classes, however, in this case, more computational power would be needed. In terms of comparison works, the use of other architectures such as ResNet or Inception can be an option.

Acknowledgment

The author would like to thank Prof. Dr. Alberto Ferreira de Souza (Federal University of Espírito Santo) for all help during neural network programming, Dr. Marcelo Magioli (ICMBio/CENAP) and MSc. Elson Fernandes de Lima (Casa da Floresta) for providing the dataset images, and Dra. Tathiane Santi Sarcinelli (Suzano S.A.) for helping with species identification.

REFERENCES

[1] J. Bogoni et al., *Wish you were here: How defaunated is the Atlantic Forest biome of its medium- to large-bodied mammal fauna?* , PLoS One, 2018.

[2] C. Jenkins and L. Pimm and N. Joppa, *Global patterns of terrestrial vertebrate diversity and conservation*, Procedures of National Academy of Science, 2013.

[3] M. Magioli et al., *The role of protected and unprotected forest remnants for mammal conservation in a megadiverse Neotropical hotspot*, Biological Conservation, 2021.

[4] F. Lima et al., *ATLANTIC-CAMTRAPS: a dataset of medium and large terrestrial mammal communities in the Atlantic Forest of South America* , Ecology, 2017.

[5] K. Simonyan and A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, California, USA: 3rd International Conference on Learning Representations, 2015.

[6] C. Nwankpa and W. Ijomah and A. Gachagan and S. Marshall, *Activation Functions: Comparison of Trends in Practice and Research for Deep Learning*, 2018.

[7] A. Gajurel and S. Louis and F. Harris, *GPU Acceleration of Sparse Neural Networks*, 2020.

[8] S. Ioffe and S. Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, Lille, France: 3Proceedings of the 32nd International Conference on Machine Learning, 2015.

[9] A. Mikolajczyk and M. Grochowski, *Data augmentation for improving deep learning in image classification problem*, International Interdisciplinary PhD Workshop (IIPhDW), 2018.

[10] N. Srivastava and G. Hinton and A. Krizhevsky and I. Sutskever and R. Salakhutdinov, *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*, Journal of Machine Learning Research 15, 2014.

[11] S. Cai and Y. Shu and W. Wang and G. Chen and B. Ooi, *Efficient and Effective Dropout for Deep Convolutional Neural Networks*, 2020.

[12] D. Kingma and P. Diederik and J. Ba, *Adam: A Method for Stochastic Optimization*, California, USA: 3rd International Conference for Learning Representations, 2015.