Instituto Superior de Agronomia, ULisboa

Master's in Green Data Science 2022-2023

Practical Machine Learning

Automated Tree Detection and Condition using Image Segmentation

Instructor: Manuel Campagnolo

Authors: Afonso Marques 25385
Diogo Pinto 24832

Introduction

In this project, we aim to develop an image segmentation and classification system for the detection and evaluation of trees. The primary objective is to analyze, in this practical case, images captured in the field and evaluate three different parameters: the presence of a tree, the absence of a tree and the condition of the tree (whether it is dead or alive).

By utilizing image segmentation techniques, we can extract meaningful information from the images and provide insights into tree distribution and health without extensive manual efforts, in a given area.

Data

The dataset used for this project consists of cell phone images taken in various field locations, recording as many photos as possible to train our model, distinguishing: tree, non-tree and dead tree. Therefore, the da dataset includes images representing different landscapes and environmental conditions, providing a diverse range of scenarios for analysis.

Before utilizing the dataset, we performed necessary data cleaning and transformation steps.

The images were checked for errors, inconsistencies, and formats since the iPhone images came in .heic format and we had to convert them to .jpeg format, otherwise the script wouldn't work. In addition, we resize the images and annotated each image with labels corresponding to tree presence, absence, or tree death. This preprocessing step was crucial to ensure the reliability and quality of the dataset for training and evaluation purposes.

Methods and Results

For addressing the image segmentation and classification problem, we employed a deep learning approach. Specifically, we utilized a convolutional neural network (CNN) model due to its effectiveness in analyzing image data.

Hyperparameters and Architecture Choices

We utilized a Convolutional Neural Network (CNN) architecture known as ResNet34.

During the training of th model, we explored different hyperparameters and architecture adjustments to optimize the model's performance. Some of the key hyperparameters and architecture choices explored are:

- Batch Size: we set the batch size to 64, wich means that the model weights are updated every 64 processed images.
- Data Transformations: we applied resizing and data augmentation to improve the model’s generalization and avoid overfitting.
- Transfer Learning: we utilized the technique to transfer learning by initializing the model with pre-trained weights from ResNet34.

- Learning Rate: we tested different learning rates to find the optimal rate for model convergence.

Training, Validation, and Test Sets

The training set consisted of images from our own data, organized into three folders: "arvore", "sem_arvore", and "arvore_morta". The validation set was created by randomly (RandomSplitter) selecting 20% of the images from each class.

The steps involved in creating these sets were:

- Counting Images per Folder: We created a script to count the number of images in each of the three folders, "arvore_heic," "sem_arvore_heic," and "arvore_morta_heic." This gave us knowledge about the dataset sizes.

```
Folder: arvore_heic
Number of images: 178

Folder: sem_arvore_heic
Number of images: 54

Folder: arvore_morta_heic
Number of images: 37

Image counting completed.
```

Figure 1- print from 'script_pml_proj.ipynb' about number of images per folder

- Conversion of .heic images to .jpeg: We used a script to convert the photographs from the .heic format, in which they were originally stored, to the .jpeg format. The CNN architecture and the utilized libraries were made compatible by this modification.

- Creation of DataBlock and DataLoaders: We used the Fastai library to build the DataBlock and DataLoaders. These parts load the data, do any necessary modifications, then divide it into training, validation, and test sets.

Figure 2- print from 'script_pml_proj.ipynb' about dataloaders

- Data Augmentation: We used several approaches, including random flips, rotations, and zooms, to enhance the training data. The goal of this expanded dataset was to increase the model's generalization and capacity to tolerate variances in the images.

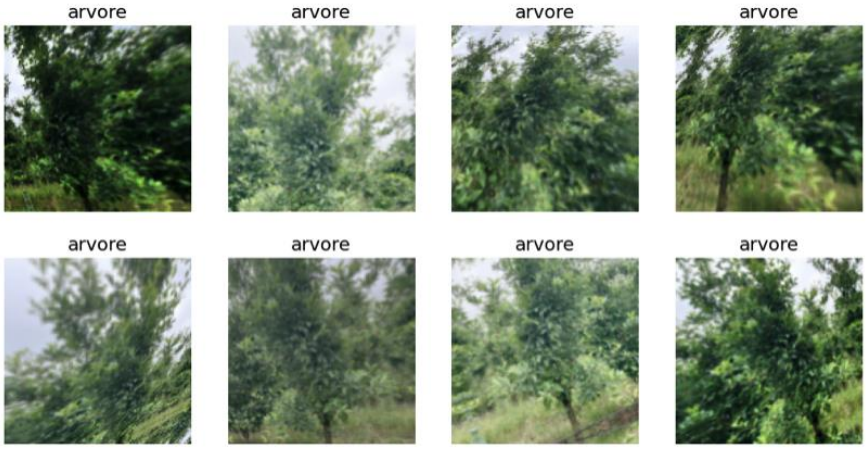

Figure 3- print from 'script_pml_proj.ipynb' about data augmentation

- Model Training: The model was trained on the training set and its effectiveness on the validation set was monitored. In order to reduce the loss function and increase the desired evaluation metrics, such as accuracy, multiple training epochs (6) were performed.

epoch	train_loss	valid_loss	accuracy	time
0	1.878874	1.666162	0.358491	01:46
epoch	train_loss	valid_loss	accuracy	time
0	1.228955	0.991616	0.471698	02:04
1	1.108405	0.840727	0.735849	02:04
2	1.024566	0.980072	0.811321	02:04
3	0.913863	0.646195	0.905660	02:10
4	0.834005	0.555908	0.905660	02:03
5	0.774698	0.504559	0.905660	02:04

Figure 4- print from 'script_pml_proj.ipynb' about the training table

- Model Evaluation: Using the unused data from the test set, we evaluated the trained model. This evaluation allowed a proper appreciation of the model's performance and adaptability to new data.

Analysis

To begin, we tracked the number of photographs in each folder to get a sense of how the dataset was put together. We noticed that the dataset was imbalanced, with 178 photographs of 'arvore_heic', 54 images of 'sem_arvore_heic', and 37 images of 'arvore_morta_heic'.

Dealing with imbalanced data can present challenges during model training and evaluation. In our example, a bias toward tree detection may result from the greater quantity of "arvore" photos compared to the other classifications. One solution to this problem would be to use methods like oversampling minority classes, undersampling majority classes, or employing artificial data generating techniques to balance the dataset.

Besides that, during model training we achieved a final accuracy of 90%, indicating that our model performed well in accurately segmenting and classifying the presence, absence, and death of trees in the images. This high accuracy demonstrates the effectiveness of the chosen methodologies, including data preprocessing, data augmentation, and the use of a convolutional neural network (CNN) architecture.

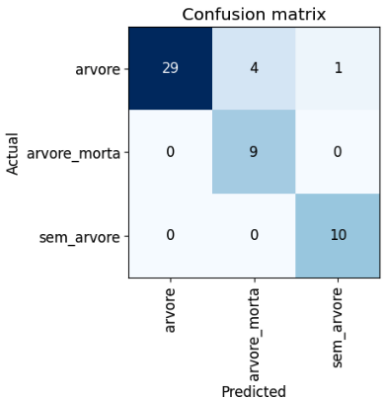

Figure 5- print from 'script_pml_proj.ipynb' about Confusion Matrix

Conclusions

In conclusion, our research was effective in creating a system for segmenting and categorizing photos that can be used to analyze pictures from the field. The results analysis showed the model's capacity to recognize the existence of trees with high accuracy and to successfully differentiate the distribution of trees, their existence or absence, and their health.

To gain even greater accuracy and robustness, other data sources can be added, and the model architecture can be improved. These additional upgrades and enhancements can be explored in the future.