AN2DL Homework 1

Aleksandra Krajnovic, Iva Milojkovic, and Mariusz Wiśniewski

Team name: Three convolutioneers

Politecnico di Milano

November 29, 2021

1 Introduction

The goal of this assignment was to train a convolutional neural network able to distinguish between leaves of various plants and classify them as 14 classes.

2 Research and Preprocessing

We started the project by researching existing approaches and solutions to similar problems. We considered constructing our own neural network architecture, but most of the examples we were able to find proved that the most effective path is to use transfer learning when problems grew in complexity [8]. The size of the dataset and the number of data entries were also not very vast, which led us to the decision to augment the dataset using the *ImageDataGenerator* from *Keras* library [1].

In the next phase, we observed the contents of the dataset, which turned out to be highly imbalanced, as the Tomato class was highly overrepresented in comparison to the rest of the classes (as shown in the figure 1).

We decided to try 3 different approaches to treat these imbalances; the first one being not doing anything else but trusting the data augmentation, the second one being leaving out a part of training images of the Tomato class (as shown in the figure 2), and the third one being the adjusting of weights based on the class count.

3 Training the Models

The pre-trained *VGG* [9] model we saw in class was computationally demanding and would take up a lot of the GPU time we had available, so we opted for the options with less trainable parameters. Moreover, newer solutions that are more effective and can be trained faster exist, so we tried those first. Our starting points were the *Inception V3* [12], *Xception* [2], and *ResNet50 V2* [5] models. Due to the imbalanced dataset, we decided to include

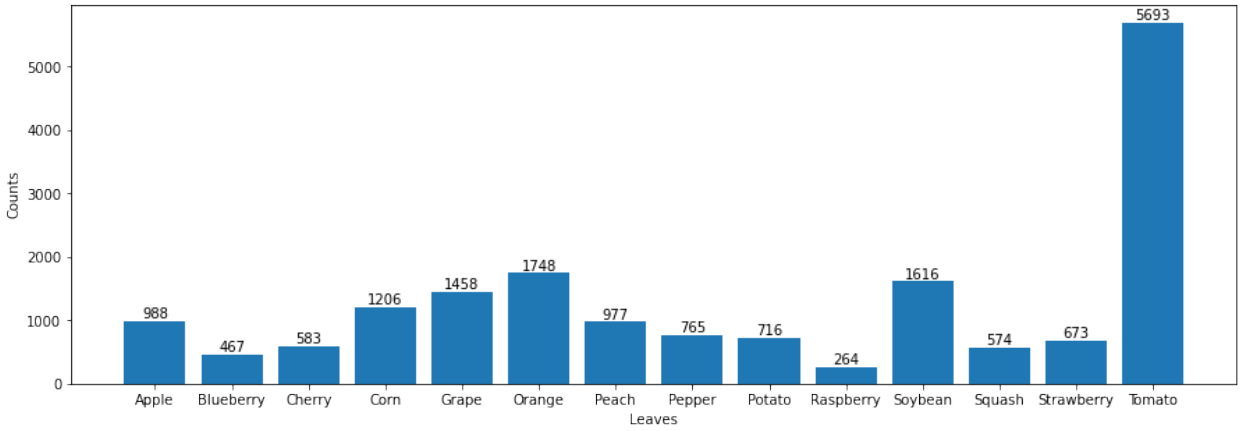

Figure 1: Population of each class depicts data imbalance.

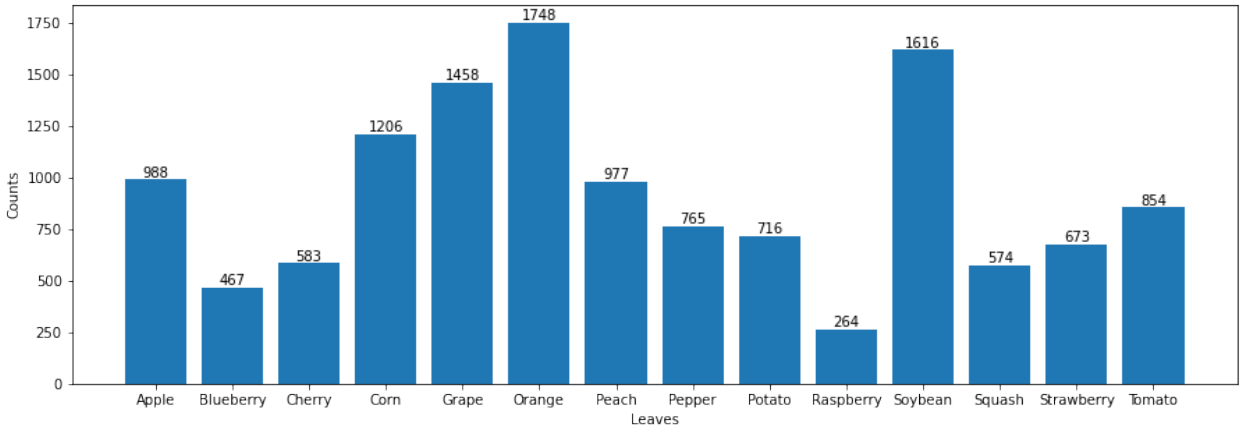

Figure 2: Population of each class with 85% reduction of the majority class.

precision and recall metrics into the training process, which was important especially in the case where we did not cut off the data from the Tomato class, and where that could potentially mislead the learning process to just increase the accuracy by predicting Tomato, the majority class, as output.

The dataset was augmented by flipping the images horizontally and vertically, zooming and rotating them, changing the brightness level, and shifting them. Moreover, the data was preprocessed using specific functions adapted to different model classes and included in the Keras library. All the models proved to be powerful in predicting the correct outputs, however, *ResNet50 V2* performed best. We, therefore, decided to pursue this one and then started to explore various options to fine-tune this model. The first decision was to unfreeze the first 2 convolutional blocks, i.e., freezing the first 86 layers and unfreezing the rest. We compared the results with a model where we unfroze more convolutional blocks, but the results did not improve.

As the results on our own validation sets never surpassed an 86% accuracy, we then decided to explore additional options proven useful in the literature. We changed the top layers added to the pre-trained model, replacing one of the dense layers with a *PReLu* activation layer [4], adding a *BatchNormalization* [6], as well as increasing the number of *Dense* layers and the *Dropout* rate [10]. This did not increase the accuracy even with the fine-tuned versions of the *ResNet50 V2* model. Furthermore, we explored the option to use the *focal loss* function [7].

The focal loss function did not improve on the accuracy, so we stepped back to the first part and used the remaining GPU time to also try training a model with more parameters, such as the *ResNet101* [3] V2 and *Inception ResNet V2* [11]. At that point, time was very constrained, therefore, we could not investigate potential fine-tuning options that might further increase its performance. Our initial approach with simpler models proved useful, as the *ResNet101* training timed-out towards the end of the 2nd competition phase. Unsurprisingly, the model did not perform exceptionally.

Lastly, we created an *ensemble* model composed of a fine-tuned *ResNet V2* trained on a reduced dataset, *ResNet50 V2* with weights adjustments, and fine-tuned *Xception* with weights adjustments, the latter ones trained on the full dataset.

4 Results and Remarks

During the testing phase we repeated the test-time augmentation approach and performed 5 random modifications on each of the test images. The augmentation operations matched those used previously.

With only 9 submissions in the final phase of the competition, we decided on the following models:

- *ResNet50 V2* with transfer learning, adjusted weights, *PReLu* activation layer, and categorical cross-entropy loss,
- Fine-tuned *ResNet50 V2* with adjusted weights and a *PReLu* activation layer, and categorical cross-entropy loss,

- *ResNet50 V2* with transfer learning and *focal loss*,
- *ResNet101 V2* with adjusted weights, categorical cross-entropy loss, and incomplete training,
- *ResNet50 V2* with transfer learning, categorical cross-entropy loss, and adjusted weights,
- Fine-tuned *ResNet50 V2* with transfer learning, categorical cross-entropy loss, and adjusted weights,
- Fine-tuned *Xception* model with adjusted weights and categorical cross-entropy loss,
- An *ensemble* model with adjusted weights,
- And *ResNet50 V2* with transfer learning, focal loss, and adjusted weights.

The model that scored best with the hidden image set was the fine-tuned *ResNet50* with adjusted weights and a reduced dataset with an accuracy score of 0.8642. We are attaching the notebook with the code in the .zip file.

The models consistently achieved the accuracy over 81%, however, we expected better results from the *ensemble* model. We observed an additional increase in the classification accuracies of already reliably classified labels, and a decrease in the ones that classified poorly before. At the time of submission, we did not have an explanation for that behaviour, but it might be possible to improve it using adjusted weights for models' votes.

We are satisfied with the results, however, we would like to test some other approaches in the future as well. The learning take-outs will definitely be useful in the second competition, where we hope to further improve our team's performance.

5 References

[1] François Chollet et al. *Keras*. `https://keras.io`. 2015.

[2] François Chollet. *Xception: Deep Learning with Depthwise Separable Convolutions*. 2017. arXiv: `1610.02357 [cs.CV]`.

[3] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: `1512.03385 [cs.CV]`.

[4] Kaiming He et al. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. 2015. arXiv: `1502.01852 [cs.CV]`.

[5] Kaiming He et al. *Identity Mappings in Deep Residual Networks*. 2016. arXiv: `1603.05027 [cs.CV]`.

[6] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: `1502.03167 [cs.LG]`.

[7] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. 2018. arXiv: `1708.02002 [cs.CV]`.

[8] Manali Shaha and Meenakshi Pawar. "Transfer Learning for Image Classification". In: *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. 2018, pp. 656–660. DOI: 10.1109/ICECA.2018.8474802.

[9] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[10] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[11] Christian Szegedy et al. *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*. 2016. arXiv: 1602.07261 [cs.CV].

[12] Christian Szegedy et al. *Rethinking the Inception Architecture for Computer Vision*. 2015. arXiv: 1512.00567 [cs.CV].