REPORT HOMEWORK-1

Heart of Gold: Bianchi Emanuele, Huang Junchong, Manenti Carlo

- Introduction to the task

To apply our knowledge of Deep Learning we took part in the first Deep Learning Homework. We were asked to provide a model capable of correctly classifying 8 different species of plants. To assess the performance of our models two hidden test sets were provided and accuracy was the metric of choice. This setup, resembling a general working request, was the perfect environment to refine our abilities.

- Understanding the data

We were not given any relevant information about the different plant species; we were provided with only 3542 samples of 96 x 96 pixels RGB images, already separated into the respective classes' subfolders. First, we noticed that the dataset was unbalanced, since both Species1 and 6 were underrepresented. Then, having a closer look at each image, we appreciated their heterogeneous features. Indeed, images had been shot at different times of the day, from different perspectives and with very different exposure. Some shadows were so prominent and peculiar that they could have defined class-specific patterns. Moreover, other images had a strong shift in the red channel. Also, the background was interesting, being easy to recognise and specific to some species. All these problems foreshadowed a challenging task.

- Approaching the task

We defined training and validation sets with a stratified split of 80% to 20%, respectively. Using a stratified split was essential to keep an equal distribution of images in both test and validation. Hence, we avoided 'unlucky' splits for which the validation set will not be consistent due to a complete lack of images from a given species. Then we built the simplest baseline model that we could: an input layer followed by a flattening layer and a soft-max output layer with 8 neurons, one for each species. It achieved around 25% accuracy, thus was slightly better than random guessing. The baseline was compared to an input-independent model, which was a copy of the baseline trained on inputs completely set to 0, thus having no information to learn from. This comparison verified that the model could extract meaningful information from the data, since the input-independent model performed worse than the baseline.

- Transfer learning and fine tuning

After setting the baseline model we focused on solving one of the main problems of the data set, data scarcity. One solution is to adapt a pre-trained model to our specific task leveraging fine tuning. Instead, training from scratch very deep models would be unfeasible, since it would mean to learn millions of parameters with limited data. Also, fine tuning extremely complex models still pose a hard challenge due to the same reason. In the end, we fine tuned a range of different model coming from Keras Applications, by freezing all the layers and training the model for only few epochs, thus train only the classification layers, and only then unfreeze everything and training the model with a very low learning rate. In the end we selected the two best models according to validation accuracy, VGG and ResNet50.

- ResNet50 vs VGG16

ResNet50 and VGG16 are two competition winners and highly used backbone models. Comparing the architecture of these models, we can observe that only the latter leverages residual modules, thus (in theory) enabling higher performance. Since on validation VGG16 and ResNet50 had similar results, we compared them on the hidden test set. As expected, ResNet50 took the edge. Hence, being better suited for our task, both theoretically and practically, we adopted a fined tuned ResNet50 trained on ImageNet for almost all the subsequent steps.

- From the ground up

We also tried to build a model from the ground up. We used as reference 'A recipe for training Neural Networks'* which gave us some good insights. Moreover, we decided to normalize the inputs to achieve better training. We were also curious of how the model perceives and learns from the data. Does the model classify each image exploiting global features, like the background, or it focuses on details, like sizes and shapes of leaves? The deeper the network, the bigger its receptive field and the more complex structures can recognise, thus (theoretically) achieving a better latent representation. To test this idea, we compared a full VGG16, which has a receptive field of 32, with a VGG16 without its last pooling layer, thus having only half the size of the receptive field (16). As expected, the full VGG-16 achieved a better accuracy. So, we decided to focus on deeper architectures. Also, complex fully connected layers did not improve the performance of the model compared to a single layer with a softmax activation. Eventually, we defined a simple model made of 6 convolutions, each followed by a 2x2 max-pooling layer and at the end a simple flattening layer plus a SoftMax to classify the latent representation. All the filters used were 3 x 3 with padding same and their numbers doubled with each convolution going from 16 up to 256. The activation function of choice was ReLU. Surprisingly, such a simple model achieved an accuracy of 77% on validation, but it was still far from the simplest fine tuned model. So, we focused our effort on improving a fine-tuned ResNet50.

- **Hyper-parameters tuning**

Tuning the hyper-parameters of a model is considered by some a dark-art, which could lead to a significant performance boost. We tried various configurations and techniques, the most relevant ones being different batch sizes and the use of Hyper-ResNet. Regarding the batch size, a smaller one (4, 8) may bring more variation between batches of images, introducing more noise in the learning procedure, thus helping to jump out of a local minimum towards a better sub-optimal status. Instead, bigger batch sizes (16 and 32) may help the learning by providing the model with a more homogeneous sample of images. Surprisingly, on the validation set a small batch size works best, while a larger one leads to slightly better results on the test set. Thus, we resort to bigger batch sizes to train our models. With no major improvement we tuned the hyper-parameters of a ResNet50 through HyperResnet. The main drawback was the time required to extensively explore the search space.

- **Augmentations**

One of the best ways to tackle data scarcity is to generate more data leveraging augmentation. Our first approach was to augment the training set with specific augmentations, thus avoid changes in the image's label. So, we applied vertical and horizontal flips coupled with small hight and width shifts. This led to a performance boost, which was even higher if used also in the first step of fine tuning. Since we had less examples from species 1, we would have liked to make the model 'pay more attention' to them. To do so, in the keras fit function, we set the class weight, which takes a list of values and it uses them to weight the loss function accordingly. This approach led us to better results for species 1 (going from 65% of accuracy to 71%), while leading to an overall reduction in the performance of the model. Hence, we tried something different. Another approach was first using class specific augmentation for Species1 with the help of Albumentations and only then augment the training set. Hence, the training set would have been more balanced and so the model would have been able to better classify Species1, without compromising its general performance. The validation results were quite promising, with the model reaching even 90% of validation accuracy, but the test results turned out to be much lower, with a drop of around 5%. Some of this discrepancy was probably due to overfitting but could be mainly accounted for balancing Species1 this way. Experimenting with augmentations we also tried Cutmix, which augment an image by an overlapping patch from another one while also updating the label to reflect the mixed content. For example, in a binary classification task, an image of class 0 with label [1,0] is augmented with a patch from an image of class 1 which cover 40% of the original one, the new label will be defined as [0.6, 0.4]. With cutmix the model achieved a closer delta between train and validation set, but compared with the same model architecture, instead trained on normal data, it did not improve the validation score. We had tried to combine normal and augmented images in different proportions and by putting constraints on the patch size, however we never got an improvement on the validation accuracy.

- TTA & Ensemble

To enhance the performance of our model we used Test-Time Augmentations (TTA). TTA feeds multiple times the same image to the model, each time with a different augmentation; by averaging the results the model defines the final prediction. We have applied TTA to different models and with different augmentation strategies, but we always found marginal improvement on the validation score. Furthermore, we also tried ensembling different models to boost even further the performance. By training multiple models and averaging their outputs we can reduce the variance of the prediction, thus resulting in a better model than the individual ones. Our best ensemble was a combination of ResNet50, VGG16 and MobileNet, with an improvement of 3% in accuracy. But even with a bigger ensemble the overall performance was not so different.

- Let's get creative

Double model: Splitting a difficult task in simpler ones proved to be a good solution in many situations. With this idea in mind, we trained a ResNet50 to classify Species1 against all the other species, achieving an accuracy of 92% on the validation set. Then we trained another ResNet50 to perform multi-class classification of the 7 remaining species, resulting in 90% validation accuracy. To merge the two models, we resort to conditional probability. Unfortunately, on both validation and test set the model did not seem promising, having an accuracy of only 84%.

Synthetic data augmentation: Trying to come up with a good approach to generate new images we stumbled upon Generative Adversarial Network (GAN). So, we tried to train a Conditional-GAN (C-GAN). Due to the very long time needed to train such models and the limited GPU run time, we were able to test only a couple of versions, which either did not produce a useful image or ended up suffering from mode-collapse. Given the results** we avoided training a classifier with the artificial examples, even though we believe it could be an interesting idea to be further explored.

- Our best model

After all the testing with fine tuning, augmentations, Species1 balancing and creative approaches our best models turned out to be a straightforward combination of augmentations and fine tuning. We used as backbone a ResNet50 trained on ImageNet on which we used a Global Averaging Pooling layer followed by the softmax output layer. Then we augmented the training set using vertical and horizontal flips combined with hight and width shifts. At first, we froze all the layers of ResNet, and trained the output layer for 25 epochs. Then we un-froze all the layers and fine-tuned them with a very small learning rate (1e-5) to avoid overfitting our data avoid. At this point we used a larger batch size of 64 images, which proved to give the best results on test set.The training lasted around 120 epochs before triggering early stop. The model reached a pick in the validation accuracy of 90% and once tested on the first hidden set scored 86,8% of accuracy, while on the final test set lost a little bit of accuracy, thus ending up at 85,3%. This discrepancy could be due to a low level of regularization of the model. Probably finding better augmentations strategies could have benefit the performance of the model. Anyway, we were quite happy with the result achieved by such a simple model.

- Conclusions

In this homework we applied our knowledge of Deep Learning by trying various approaches and then combining them in a simple, yet effective model. Of course, there are still some open issues that we were not able to fully address. Our major issue was achieving good accuracy scores on Species1. This was probably due to us focusing too much on generating new images to reduce the gap between training and validation accuracy. Meanwhile, an ensemble or TTA technique could have led us to better results.

Reference:
*https://karpathy.github.io/2019/04/25/recipe/
**C-GANS results: https://drive.google.com/drive/folders/1zraJYs39ig3pjqTMo6BLYHbhJya-bd7u?usp=sharing