NonDormoLucido:
Anastasi Nicola - 10807261
Casciani Daniele - 10638165
Terraneo Leonardo - 10530883

Homework 1 - Report:

Image classification on leaves dataset

Problem description:

The aim of the first homework was to build and train a Convolutional Neural Network that could perform image classification on a dataset of leaves' pictures taken in the wild.

Approaching the problem:

In the early stage of the challenge we decided to focus on the assigned dataset, by looking through what kind of data were stored in it.The data was partitioned in 8 classes each one containing pictures of a different type of leaves taken in a natural setting.
Due to the limited amount of daily submissions we decided to work locally most of the time and submit only after obtaining positive results in our local environments, to do this we organized the dataset into three subsets (namely *train*, *val* and *test*) storing respectively the 75%, 15% and 10% of the whole data, to locally perform different attempts and strategies before testing our models on the platform. Local evaluation was performed using a combination of metrics such as accuracy, precision and recall and confusion matrices to visualize the classes on which the networks performed better.
The first model was based on a hand-made network, among the ones proposed as an example during one of the practical lectures, this was expected to be, though with lower performances, faster during training which was crucial to receive quick feedback on how well the data split and augmentation was impacting the performances. It turned out that using the splitted dataset alone, without any sort of augmentation, the model worked better on some classes than others, specifically the ones with more samples, which was to be expected. The classes were in fact unbalanced in the number of samples, that were 186 in the smallest class and 537 in the biggest one, making the application train better on the recognition of images from the most populous classes and misclassify the ones that had fewer.
To avoid this problem we created a python script that performed undersampling of all the classes, getting random samples each time and setting the number of samples equal to the lowest one to make the classes balanced and let the network train on a uniform distribution. As expected the performances improved especially on the aforementioned critical classes. However, since the average statistics is low, it suggests that the usage of a small portion of the database implies a too big loss of information. Thus, a different approach was adopted. At this point we have made several attempts using databases with different fixed numbers of samples for each class performing both under- and oversampling: the former when the original data were higher in number while the latter when the number was lower. Since we

introduced the same images many times using oversampling, performing data augmentation was needed to avoid the possibility of overfitting on the few images that we had.
Recalling that the type of data we had, *i.e.*, plants, we considered it appropriate to try different types of augmentation, namely *rotation*, *flip*, *shift*, *brightness*, *fill mode*, *black background* and *zoom*. At first these augmentation techniques were tried by themselves to note how the performance changed when they were present and from the data we saw that zooming levels and brightness levels didn't enhance nor diminish the performances of the network while the rotation, shift and fill mode impacted a lot the accuracy of the model.
Augmenting the data wasn't the only technique that was applied to improve the performance score, we also dedicated time in researching transfer learning and all the networks that were available to be used in order to solve this classification problem, the networks we worked with are the following:

VGG16

The first attempt to use a transfer learning approach and the one on which we spent most time experimenting. We adopted vgg16 as a supernet to which were attached one flattening layer followed by two dropout and two dense layers.
What we have done is to re-propose all the experiments done with the simpler hand-made network to find the best augmentation to exploit this architecture at maximum.
This network alone reached an accuracy of 65% on our local environment that increased up to 83% after fine tuning of the last 15 layers. Very confident about the obtained result we decided to test our model on the platform reaching an average score around the 80%.
From this moment, since we already found a very good model, our goal became to find a way to improve this score.
We thought that trying different hyperparameters could be a good idea. Therefore we tried to use: lower learning rate, longer early stopping and performing the fine tuning on different numbers of layers. After this analysis we reached what we can define the best setup probably for this network, but without increasing significantly the accuracy that remains always around 80%. This took us to train also different well known networks with the expectation to find some better performance.

Ensembles:

Another approach we tried was building an ensemble classifier made of eight one-vs-rest binary classifiers in the form of networks trained to recognize one species vs all the others. We decided that this approach could be viable seeing the limited number of classes and samples, which meant that the entire classifier would take around two hours to train. These networks had the same architecture as in our VGG16 experiments, except for the last layer, which was composed of a single neuron with a sigmoid activation function. The idea was that, for each sample, if the n-th network had an output higher than all others, the classifier would predict class n. However we ran into the problem of the networks tasked with recognizing classes 1 and 6 having very few positive samples to train on, leading to very low recall for those two classifiers and therefore to the ensemble classifier having very low accuracy on the aforementioned classes. We attempted to solve this issue by using the data augmentation techniques that had worked best so far, however, probably due to the extreme class imbalance, they did not perform as well as we expected and ended up introducing a lot of overfitting on the minority class. Therefore this approach was eventually scrapped.

InceptionV3:

InceptionV3 was a really promising pick, promising good performances on image classification but the first tests weren't as good as expected yielding a 45% score on codalab. The first tests were performed on the balanced and augmented dataset The improvements came while increasing the amount of parameters of the network, this was possible by training more and more layers of the model, up to 22 million parameters, until the best result came after adding 2 new dense layers, one with 128 units and the other with 32, and applying a fine tuning procedure on the last 294 layers of the inception model. This model, trained on the augmented dataset with the lowest possible learning rate and a patience value of 50, was then tuned on the same set and obtained a 72% accuracy rating on codalab, marking the best result obtained using this network for transfer learning. Since it wasn't as good as vgg16 though it was left behind in order to improve the results with the best performing network.

VGG19 and ResNet

After our experiments on vgg16 we decided to move to these two models. They were very similar to the one presented before, i.e. two dropout and two dense layers that follow the respective supernet. The basic idea was to see how these networks work with the found configuration, and after that improve that performance by re-proposing the same techniques already done for the vgg16. However, the model found with the vgg16 still was the best.
We noticed that all models, though very different both in terms of theoretical construction and expected performances, showed almost the same results: all of them assessing around an 80% in accuracy.
This led us to think that more accurate work with our dataset was needed, and by a further analysis, it turned out that many of our images were characterized by the presence of many shadows that probably make our model more difficult to correctly classify them. Hence, we thought of a different augmentation, namely *cutout*, i.e. removing some of the pixels from original images to make our already described models (Vgg16, Vgg19 and ResNet) more robust.
Adopting this technique with the vgg16 we reached a score around 82%, thus obtaining a slight increase.

Best Solution

In the end, from the dashboard of Codalab, we noticed that our model, i.e. the Vgg16, was still performing badly on two particular classes, i.e. 1 and 8. Thus, our latest strategy was to no longer use a balanced training set, but a very unbalanced one in favor of these two classes to reduce their misclassification.
Our current "mix" includes 3400 samples for classes 1 and 8, while a number between 600 and 800 for the others. Our current best model, using this training set, reaches an average accuracy around 84%.