Massimiliano Arca
Gianmarco Aulicino
Alfonso Catelli
27 November 2021

Artificial Neural Networks and Deep Learning

Image Classification Challenge

The objective of this challenge was to correctly classify images of leafs based on the species of the plant they belong. With this objective in mind, our team started from the very basics, developing a CNN from scratch and gradually growing to a Transfer Learning approach.
We developed three different solutions for this problem :

- Convolutional Neural Network from scratch
- Transfer Learning approach with Adam optimiser
- Transfer Learning approach with SGD optimiser

Preprocessing Data

Starting from the common parts, we used Split Folders library to divide the dataset into two different ways :

- with test part : with a distribution between train, val and test of respectively 80-15-5, this split was used earlier in the process to train and test locally the networks to have a general idea on the quality of the models
- without test part : with a distribution between train and val of respectively 80-20, this was used to have a better train phase for the latest versions of the models

Analysing the dataset came out that the classes were unbalanced, in fact some of them had more images than the other ones. To face this problem has been done class balancing assigning a weight to each class in order to give less relevance to the classes with a higher number of samples with respect to the ones with a smaller number.

Another common part is the data augmentation, useful to generate variation in the data and improve the model's stability. We took inspiration from the lab notebook techniques but changing the single parameters multiple times to know which were the best and also added some other techniques like brightness since we noticed it improved a lot the accuracy.
We used also a learning rate schedule approach in order to have a more efficient training by decreasing the learning rate every few epochs.

Custom CNN

Let now explore the CNN from scratch, with it we reached a score of 0.877 and this is our proudest solution even though is the one with the lowest accuracy.
To reach this result we started by taking the CNN shown at the class laboratory with 5 convolutional layers, a flatten layer and a dense layer, that initially gave us a 0.3 accuracy. Before switching to a transfer learning algorithm we wanted to push this version a little bit higher in the score by gradually adding new convolutional layers and changing some of the hyper-parameters. When reached 7 convolutional layers and set a customised number of filters, the first discrete results arrived with a 0.66 accuracy. We thought that this model could have been a good starting point to reach some better scores. Continuing to optimise the model gradually by switching the flatten layer with global average pooling layer and changing all the activation functions from ReLU to Leaky ReLU, the accuracy jump up to 0.79. We then decided to regularise the model in order to avoid overfitting adding dropout and batch normalization, arriving at a 0.86. The last tried thing was the weight decay regularisation and we performed hyper-parameter selection by training three different models with respectively $\lambda=0.01$, $\lambda=0.001$ and $\lambda=0.0001$. We saw an improvement in accuracy to 0.877 with $\lambda=0.001$. Arriving at this point we tried to add another convolutional layer in order to capture bigger pattern in the images with the result of a drop in accuracy, so our final choice was to stay with 7 convolutional layers.

Transfer Learning

While improving the CNN model, we decided to start in parallel a transfer learning approach using the VGG16. As done with the previous model, the first thing was just to look at the laboratory notebook did in class, by copying it and bringing some little modifications a score of 0.72 was reached in the first attend. Watching our steps in the other model we tried to change from flatten to GAP layer, to change in leaky ReLU all the ReLU activation functions, to modify the number of neurons in dense layer and to fine tuning the number of trainable layers of the VGG16 net. Finally reaching a 0.93 score.

Monitoring the history of the training, the accuracy and loss were promising but the hidden test shown slightly lower accuracy so we decided to regularise the model to not overfit and improve the performance, so we added a batch normalization layer and a kernel regularized l2 norm with rate 0.001 and we got 0.932 of accuracy.

At some point we wanted to try to use new pre-trained models to improve even more our accuracy. We moved to Xception as supernet instead of VGG16, 46 layers trainable over 133, and with the previous design we got 0.943 of accuracy. Finally we added activity kernel l1 norm and changed ratio of dropout from 0.3 to 0.5 but the results were worse, got back to 0.92 of accuracy and discarded this solution and our final decision was the model with Xception and a score of 0.943.

Last thing tried was using Inception_v3 with 100 layers set to trainable and SGD as optimiser and we got our final best score: 0.9603.

Final Submission

We submitted as final model the one with Xception and Adam as optimiser because in the second phase of the evaluation the two transfer learning approach were giving the same result.