ANNDL Homework 1 Report

Introduction

In the following document we will illustrate the process through which we built our model, reaching an accuracy of 95% on the given problem.
Since we decided to work on Google Colab we were restrained in the complexity and computational cost.
We started from the notebooks shown in the lab sessions and tried to tweak most of the hyperparameters and variables. Doing so we noticed that some of them didn't affect the final result, e.g. the Adams optimizer and the Categorical Cross Entropy loss function, which seems to be the standard for simple classification models.

Simple CNN

The first approach we tried was a simple Convolutional Neural Network with 5 convolutional layers, each followed by a ReLU activation layer and a 2x2 MaxPooling. We also tried changing the structure of the model several times, adding and removing layers, and some regularization techniques such as Dropout and changing the activation function to a Leaky ReLU and noticed some minor improvements on the performance but still far from the desired outcome, as we only got a 54% training accuracy. Hence, we decided to switch to transfer learning.

Transfer Learning

By leveraging previously trained models and including them in the structure of our new model we were able to obtain a noticeably higher accuracy.
The first model we used was VGG16, which, after several tries and substantial tweaking surprised us with the astounding accuracy of 87%.
We also tried VGG19 and made some changes to the structure of the layers: by thorough experimentation and inspired by famous architectures, we decided to add a Global Average Pooling layer right after the convolutional part of the network, substituting the previous MLP network. This reduced the number of parameters to be trained and obtained equally good results as we already got enough nonlinearity in the convolutional part. With this approach we reached an accuracy of 90%.
To further improve the performance we tried some newer models, Inception and Xception.
We then proceeded to tune the hyperparameters and structure of the model. Following are some of the parameters that mostly influenced it:
- the learning rate was changed from 1e-5 to 5e-5 and eventually lowered to 2e-5 to reduce oscillations
- the number of layers of the fully connected network, in particular the addition of a Dropout layer seemed to slightly increase the model's performance
- image resizing of height and width: we found that increasing from 256 to 512 boosted the accuracy

- batch size: we tried to increase it to reduce overfitting, but that significantly lowered the results, and we settled on a bs of 8
- Data Augmentation parameters: the only parameter which seemed to influence the accuracy was the preprocessing function designed for the Inception model which we included in the ImageDataGenerator

The final changes to our model, that helped us to reach a test accuracy of 95% were:
- adding a Learning Rate Plateau, that allowed us to dynamically change the model's learning rate whenever the val_loss parameter didn't change
- increasing the patience of the Early Stopping (the training often stopped too early due to a low value of patience)
- choice of the percentage for the splitting of the dataset into training and validation. We changed from an initial 30% into 20% test size.
- number of frozen layers in the model used for Transfer Learning, the right value was found heuristically.

In conclusion, an additional feature which would have probably enhanced the accuracy was cross-validation in order to minimize overfitting, but the training would have been much slower.