REPORT AN2DL

Tommaso Lucarelli, Emanuele Baldelli, Tecla Perenze.
Novembre 2021

1 INTRODUCTION

We used Kaggle as an environment for training the NNs and uploaded the given database on the platform.
We noticed that the database was scarce and unbalanced, thus in order to overcome this problem we implemented Data augmentation and Weights Balance during the training phase.
Training and Validation sets have been split with ImageDataGenerator with a rate of (80-20).
To achieve our final result, we designed and trained our Neural Networks with different techniques and implementation, that will be better explained in the following paragraphs.
We discovered that a combination of the mentioned above led to our best result.

2 EXPERIMENTS

2.1 FIRST MODEL WITHOUT TRANSFER LEARNING

First, we opted for a training of a Neural Network without implementing theTransfer- Learning, in order to make a first acquaintance with the subject. Since, the success of a machine learning project is often crucially dependenton the choice of good hyperparameters, we decided to use Keras Tuner, an hyperparameter optimization framework, in order to implement the hypertuning, namely train different models for spotting the best model andconsequently running the descriptive one.
The parameters obtained by Keras Tuner have been implemented to create our model.
We used 3 Convolutional Layers with MaxPooling for feature extraction and [Flattening - Dropout (0.3) – Dense (128 neurons) – Dropout (0.3) - Output] layers for final classification.
The outcome we first obtained by following this approach is (0.40) on theonline test.
Furthermore, we reached (0.569) by adding two more convolutional layersand adding more neurons to the fully connected one.

2.2 TRANSFER LEARNING

The model was tested extensively by applying changes to all its parts.
We experimented with many different CNNs (VGG16, Inception, Xception), for each of them we did Transfer Learning by freezing the convolutional layers and training the fully connected ones and then Fine Tuning by unfreezing all the convolutional layers, lowering the learning rate and retrained the whole model.

2.3 DATA AUGMENTATION & WEIGHT BALANCE

As mentioned [1], we implemented the techniques of Data Augmentation andWeight Balance to solve the imbalance and the scarcity of the dataset.
At first, we used basic features of ImageDataGenerator (rotation, shift, zoom, flip, rescale, brightness) but since data were too clean compared to the one on the online test set, we tried to dirt them by using self made preprocessing functions.
We combined a function that transforms RGB colors to HSV and another function to add noise to the image by adding a value to each pixel following a normal distribution.

3 BEST MODEL

Our best model has been achieved through Transfer Learning with Inception.The final structure is the following one:

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 256, 256, 3)]	0
inception_v3 (Functional)	(None, 6, 6, 2048)	21802784
Flattening (Flatten)	(None, 73728)	0
dense (Dense)	(None, 512)	37749248
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 512)	262656
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 14)	7182

Total params: 59,821,870
Trainable params: 59,787,438
Non-trainable params: 34,432

The first time we submitted our model we reached 0.86. Thus, by modifying Data Augmentation parameters, preprocessing functions and by retraining the model several time we managed to reach 0.91.

4 MISTAKES AND LESSONS LEARNED

To create a balanced dataset, we decided to downsize all the classes to have the same number of samples. Despite obtaining a balanced dataset, we noticed that we were wasting lots of useful images, and that it was better to balance the weight loss for each category instead.

In order to improve our model we re-trained it by applying preprocessing functions (add_noise, color_change) to the entire dataset. The problem was that the images were so different from the original ones that the model began to unlearn, and the validation accuracy dropped drastically. So,in the following training we decided to apply this kind of preprocessing just to 80% of the dataset.

5 FINAL MODEL

The final model has been achieved by a composition of the two best models obtained through Xception (0.86) and InceptionV3(0.91). We used a Keras Average Layer to average the output of the two models in order to make better predictions.
The end result was [0.92] in phase 1 and [0.91] in phase 2.