Image Classification

Homework 1 – Artificial Neural Networks and Deep Learning 2020

DATASET LOADING

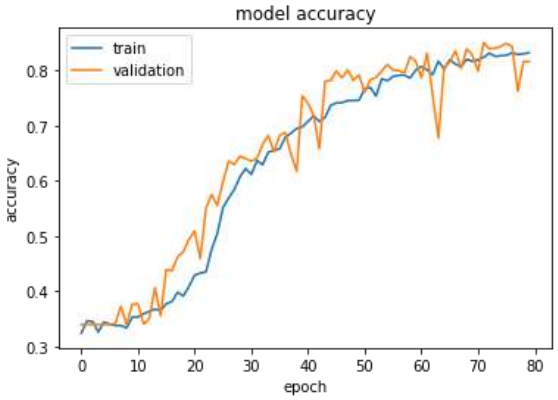
The images of the training set have been automatically arranged in three separate folders, one per label, by executing the simple Python script *JSONToFolders*. The training-validation split is operated by the function *shuffle_validation_weighted*, which takes the percentage of examples to be used for validation as input and moves the files in a hierarchical folder structure, such that it works with *flow_from_directory*. The validation examples are selected at random, depending on the *numpy* seed, which is fixed at the start of each notebook. The training set and the validation set are then loaded through *ImageDataGenerator* objects, resized, pre-processed and augmented.

FIRST TESTS WITH BASIC NETWORK

The model is a simple, LeNet-inspired convolutional neural network, composed of an alternating sequence of *Conv2D* and *MaxPool2D* layers for the feature extractor part, followed by a *Flatten*, and two *Dense* layers for the classifier. All layers use the ReLU activation, except for the last one, which uses Softmax.

The following hyperparameters were tuned:

- Depth (number of layers)
- Width (number of filters per layer)
- Resolution of the input images
- Learning rate
- Batch size
- Number of neurons in the first *Dense* layer

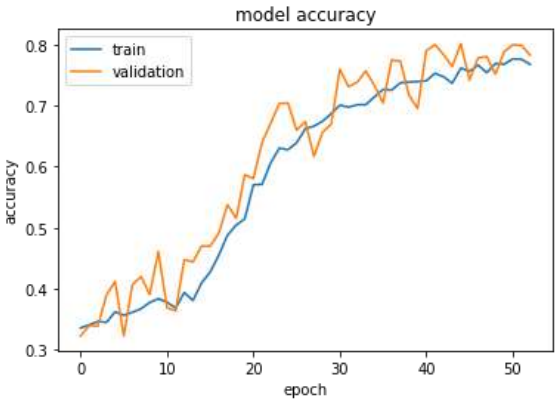


Furthermore, the addition of a *Dropout* layer between the two *Dense* ones was tested with various rates. The Early Stopping technique was employed to speed up the testing process. The SGD optimizer was initialised to a momentum of 0.9 to better escape local minima.

The best score was **85%** in validation accuracy.

INCEPTION-LIKE CUSTOM MODEL

It has been observed that the face masks the network is supposed to detect can occupy a portion of the image of variable size. Therefore, a model which employs several filters with different dimensions at the same time was considered especially suitable for the problem. The basic network was modified by integrating modules inspired by the Inception architecture, consisting of 1x1, 3x3 and 5x5 parallel convolutional layers with *same* padding and a *Concatenate* layer, followed by max pooling. The 3x3 and 5x5 convolutions were preceded by 1x1 convolutions to enhance performance by reducing the number of input channels.

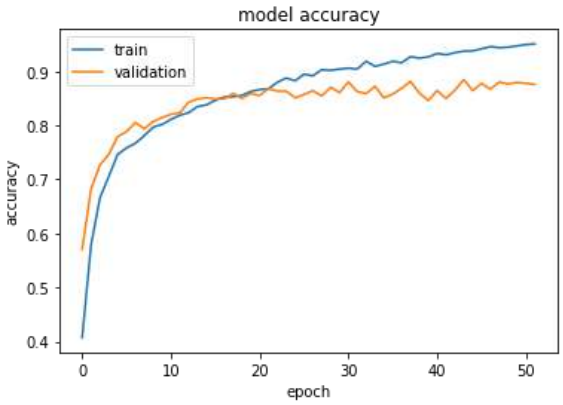


Additionally, *BatchNormalization* layers were added after each concatenation, with the purpose of limiting overfitting. The hyperparameters were selected through a simple grid search.

The model achieved a maximum validation accuracy of **80%** with a depth of 4 (stacked modules).

XCEPTION

The next step was trying to use transfer learning on the Xception network architecture.



The starting weights were obtained on the ImageNet dataset. The classifier was rebuilt by adding a *GlobalAveragePooling2D* layer and a variable number of *Dense* + *Dropout* couples before the final Softmax *Dense*. The optimizer was changed to Adam.

First, the network was retrained in its entirety. This approach scored a maximum of **88%** validation accuracy.

A great performance improvement was obtained by locking all layers of the convolutional part, except for the last two blocks, which were retrained together with the classifier. This allowed the network to achieve a validation accuracy of **93%** in just 15 epochs, which was later increased to **94%** by unlocking the rest of the network and retraining everything together.

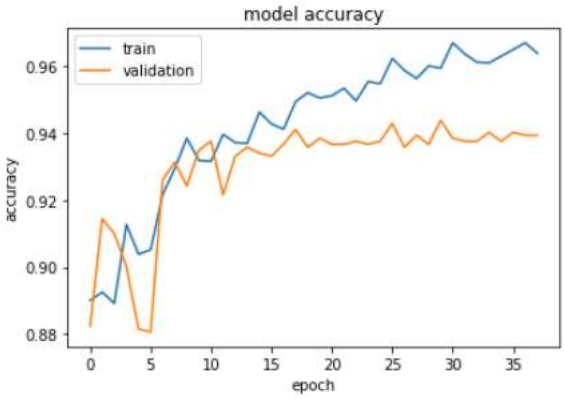
Further advancements were made possible by three, main intuitions:

1. Applying L2 kernel regularisation to the *Dense* layers to reduce overfitting;
2. Increasing the resolution of the input images to 512x512;
3. Weighting the loss function to compensate the imbalance in the number of samples per class in the dataset, as to reduce bias.

These observations allowed us to score a **96%** validation accuracy and a **95.77%** accuracy on the test set.

VGG19

The much simpler VGG19 architecture was also tested extensively for transfer learning.



The classifier was constructed by inserting a *GlobalAveragePooling2D*, two *Dense* ReLUs with *Dropout* and a final Softmax *Dense*. The network was later modified by adding *BatchNormalization* layers after the convolutional ones. L2 kernel regularisation and weighted loss function were used.

The process consisted in training, unblocking a small number of convolutional layers and retraining the network gradually, in four total steps.

VGG19 performed similarly to Xception, although it required much more time to train.

The best result was a **95%** validation accuracy and a **94.44%** accuracy on the test set.

FURTHER TESTING

Other popular architectures were tested, most notably: EfficientNetB0, MobileNetV2, ResNet-50.

See the notebooks for more details.

Marco Premi – 941388

Alessio Martignetti – 968430

Francesco Sammarco – 966229