Artificial Neural Network and Deep Leaning

Homework 1: Image Classification

Giovanni Dispoto, Matteo Sacco

November 22, 2020

The first model used for solving the problem was a simple model with 5 convolutional block and each block has one convolutional layer with 3x3 kernel, a ReLU activation function and one Max-pool layer with size 2x2 and stride 1. The Fully Connected part was made only by one hidden layer with 512 neurons. For the Data Augmentation we choose to do simple transformation such as zoom, horizontal flip, vertical flip and rotation. This model could not fit the data well, after several epochs the training loss remained high so we concluded the that the bias with respect to the task was too high and decided to try adjust some of the hyper parameters.

Then we tried to adjust batch size, reducing the value from 32 to 16. We also modified the learning rate, reducing it from 1e-4 to 1e-5, but with little improvement even after many epochs.

After this, we tried to modify a bit the model in order to augment the parameters of the network.

The second model proposed is made of 5 identical "blocks" of layers, each consists of 2 Conv. layers with 3x3 kernels. The number of filters goes from 16 in the first block to 80 in the last block and at the end of each block there is a Max-pool layer with size 2x2 and stride 2. The five blocks' output is fed to a FC part, made of one hidden layer with 512 neurons. The size was decided after having trained different models. We noticed that values beyond 512 make the network overfit quickly on the training set.

To help the model generalize we added to the ImageDataGenerator a shear_range.

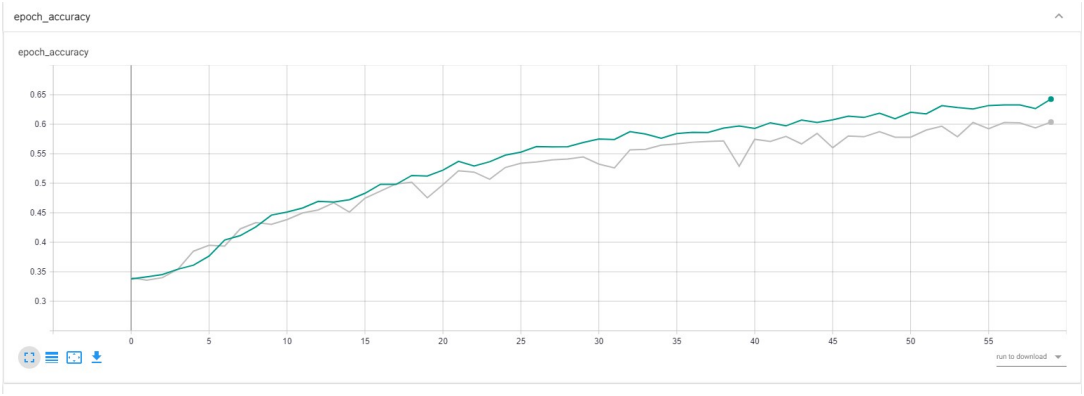

Figure 1: *Accuracy of the model trained from scratch after 60 epochs*

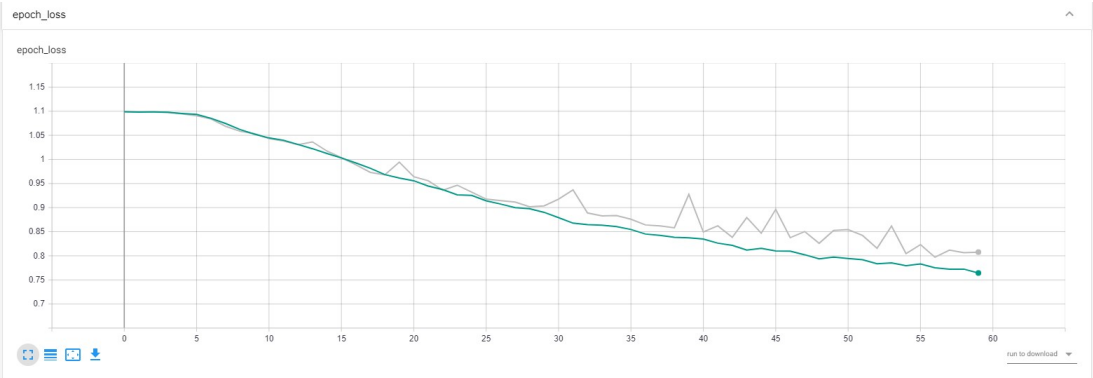

Figure 2: *Loss of the model trained from scratch after 60 epochs*

This model allows us to reach a quite good result, with accuracy ≈ 0.65 on validation set and accuracy ≈ 0.66 on test set. For regularization we have used **Early Stopping** with patience 5 with option `restore_best_wights=true` in order to reload the best weights learned and one Dropout layer in the fully connected with p = 0.5.

Looking at the graph we saw that the model tends to underfit, so we tried to make it more complex by augmenting the depth of the CNN, the size of the filters, moreover we changed the structure of the CNN, as we had hoped we obtained better accuracy.

```
#Creating a CNN from scratch
start_f = 16
depth = 6


  # Features extraction
for i in range(depth):
    if i == 0:
         input_shape = [img_h, img_w, 3]
    else:
         input_shape=[None]

    if (i > 2):
        model.add(tf.keras.layers.Conv2D(filters=start_f*2,
                                         kernel_size=(1, 1),
                                         strides=(1, 1),
                                         padding='same',
                                         input_shape=input_shape))

        model.add(tf.keras.layers.ReLU())

    model.add(tf.keras.layers.Conv2D(filters=start_f,
                                         kernel_size=(3, 3),
                                         strides=(1, 1),
                                         padding='same',
                                         input_shape=input_shape))
    model.add(tf.keras.layers.ReLU())
    model.add(tf.keras.layers.Conv2D(filters=start_f,
                                         kernel_size=(3, 3),
                                         strides=(1, 1),
                                         padding='same',
                                         input_shape=input_shape))
    model.add(tf.keras.layers.ReLU())


    model.add(tf.keras.layers.MaxPool2D(pool_size=(2, 2), strides=(2,2)))

    start_f *= 2
```

Figure 3: *Another model tried*

Then we tried to train some models with a similar, yet deeper, CNN and we also tried to adjust the kernel size, however none of the models performed better.

After having trained these models, we have tried to use Transfer Learning. The first model trained was a CNN based on VGG16 trained on ImageNet. We kept the data augmentation equals to previous model, excepts for an additional`preprocessing_function`to add noise to the training images. For regularization we used Early stopping with patience 5 and a dropout layer at 0.3. After having trained some models we settled on the following hyper-parameters: first 15 layers of VGG16 freezed, leaerning rate 1e-5, kernel size 3x3.

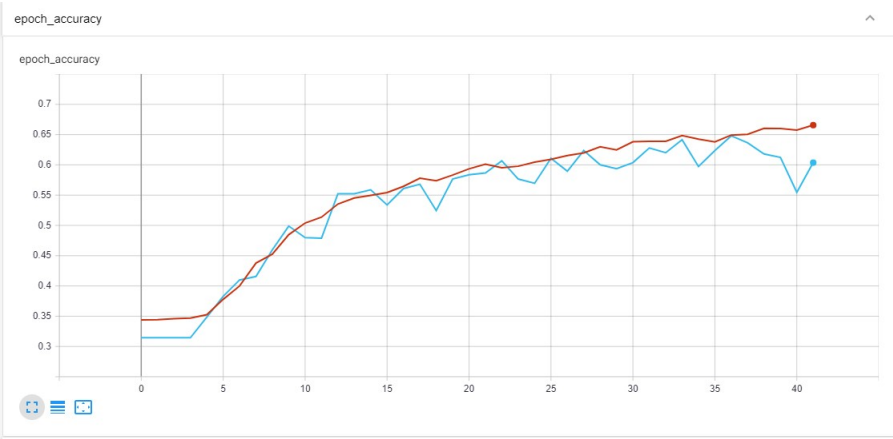

Figure 4: *Accuracy of the model tested*

The last model proposed is a model based on VGG19. We kept data augmentation parameters equals to previous test. The FC part is made only of one hidden layer with 128 neurons and one dropout layer for regularization at 0.3. We keep also using Early stopping with patience 5 in order to stop the training when it started overfitting the dataset or simply stopped improving.

VGG19 was fine-tuned from 13-th layer, because some experiments showed that was the layer to start from to obtain the best performance. After having trained several models we ended up on the following hyper-parameters: dropout 0.3, batch size 16, learning rate 5e-5, This model allowed us to reach a good result with accuracy ≈ 0.875 on validation set and an accuracy ≈ 0.897 on test set.

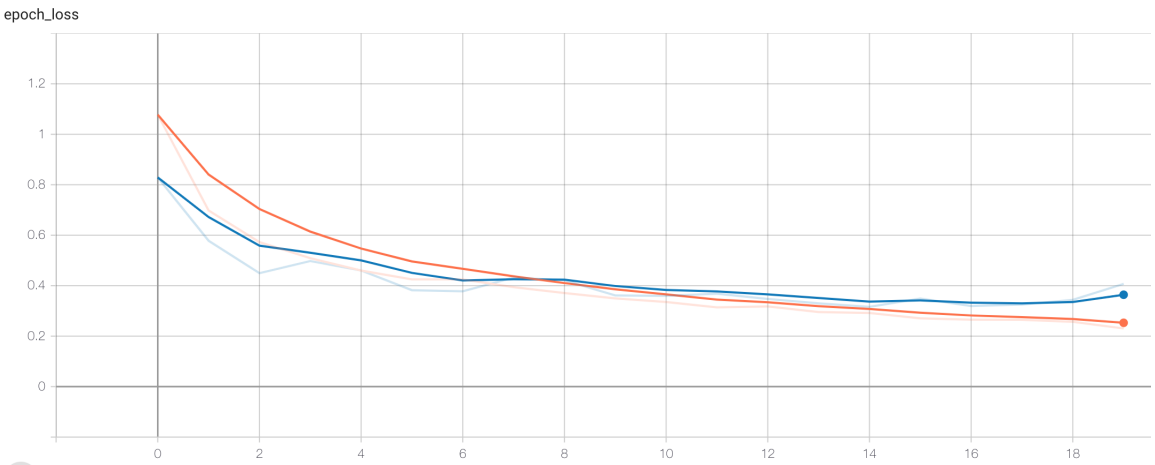

Figure 5: *Loss of the model with transfer learning on VGG19*

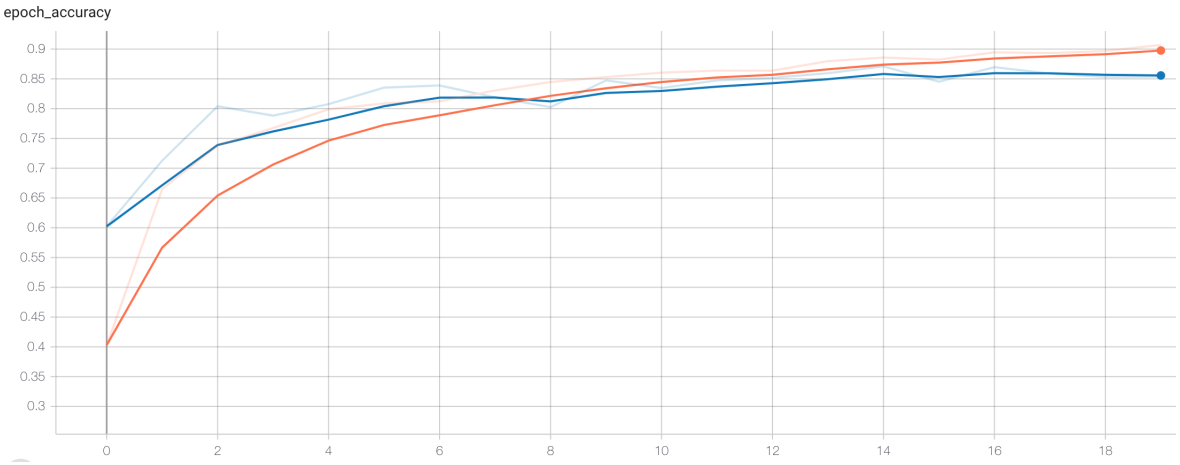

Figure 6: *Accuracy of the model with transfer learning on VGG19*