## **Environment Set Up**

This section is made to configure the environment, import useful libraries and set up the directories:

In [ ]:

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

import pandas as pd

import numpy as np

from sklearn.metrics import classification\_report, confusion\_matrix

from keras import layers

from keras.layers import Dense

from keras.layers import Flatten

from keras.layers import Dropout, GlobalAveragePooling2D

from keras.models import Sequential, Model

from keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications import resnet50, MobileNet,VGG16

from tensorflow.keras.applications.resnet50 import preprocess\_input as preprocess\_input\_resnet50

from tensorflow.keras.applications.mobilenet import preprocess\_input as preprocess\_inputMN

from tensorflow.keras.applications.vgg16 import preprocess\_input as preprocess\_inputVG

In [ ]:

*# Import main libraries*

from time import time

from datetime import datetime

from google.colab import drive

import cv2

import tensorflow as tf

from tensorflow import keras

from keras.preprocessing.image import load\_img

print('TensorFlow version:', tf.\_\_version\_\_)

import zipfile

from shutil import copyfile

from matplotlib import pyplot as plt

import numpy as np

import random as python\_random

import glob

import shutil

from random import seed

from random import random

import os

import os.path

from IPython.display import Javascript

TensorFlow version: 2.8.2

In [ ]:

*# Set seed for reproducibility*

!PYTHONHASHSEED = 0

np.random.seed(0)

python\_random.seed(0)

tf.random.set\_seed(0)

/bin/bash: PYTHONHASHSEED: command not found

In [ ]:

*# Mount GDrive*

drive.mount('/content/gdrive/', force\_remount = True)

Mounted at /content/gdrive/

# **2) Modeling: Transfer Learning**

## **Principal Steps**

The principal steps to define a Neural Network are the following:

1. Loading of the Pre-trained Neural Network;
2. FFN Architecture Definition;
3. Compilation: definition of network macro-elements (optimizer, learning rate in the optimizer, loss function, additional metrics, parameters, etc.).
4. Training (epochs, batch size, etc.)
5. Summary and Visualization of the results

### **2.1) ResNet50v1**

In [ ]:

base\_path = '/content/gdrive/MyDrive/FDL2022Project/Dati/Sample/Train/'

To achieve better performance than those obtained on the networks previously developed in notebook 1 we use **Transfer learning** (fine tuning) approach. The bese\_model (aka pretrained model) choose for this work is the ResNet50.

To make the classificatory model as generalizable as possible on new data, the technique of data augmentation is used. Data augmentation creates a more complete and consistent set of data.

The batch size is set to 32, will be larger in the next step. Moreover, the image are rescale 1/255 (rgb) and risize from 250x250 to 224x224 dimension (iterpolation = nearest), this is done to better adapt the dataset to the base model, that is train on imagnet (224x224)

In [ ]:

size=224

channels=3

batch\_size = 32

num\_classes = 5

train\_processing = ImageDataGenerator(rescale=1.0/255,

validation\_split=0.2,

rotation\_range = 25,

width\_shift\_range = .2,

height\_shift\_range = .2,

horizontal\_flip = True,

zoom\_range = .2)

validation\_datagen = ImageDataGenerator(rescale=1.0/255,

validation\_split=0.2)

train\_generator = train\_processing.flow\_from\_directory (base\_path,

target\_size=(size, size),

batch\_size=batch\_size,

shuffle=True,

class\_mode='categorical',

subset='training',

color\_mode="rgb",

interpolation='nearest')

validation\_generator = validation\_datagen.flow\_from\_directory(base\_path,

target\_size=(size,size),

subset='validation',

color\_mode="rgb",

batch\_size=batch\_size,

class\_mode='categorical',

shuffle=False,

interpolation='nearest')

Found 3298 images belonging to 5 classes.

Found 823 images belonging to 5 classes.

In [ ]:

for image\_batch, labels\_batch in train\_generator:

print(image\_batch.shape)

print(labels\_batch.shape)

break

(32, 224, 224, 3)

(32, 5)

In [ ]:

pretrained\_model = resnet50.ResNet50(

weights='imagenet',

include\_top=False,

input\_shape=(size, size, 3),

pooling='avg',

)

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50\_weights\_tf\_dim\_ordering\_tf\_kernels\_notop.h5

94773248/94765736 [==============================] - 1s 0us/step

94781440/94765736 [==============================] - 1s 0us/step

In [ ]:

print('Numero layers ResNet50: {}'.format(len(pretrained\_model.layers)))

Numero layers ResNet50: 176

In [ ]:

layer\_dict = dict([(layer.name, layer) for layer in pretrained\_model.layers])

for layer in pretrained\_model.layers:

layer.trainable = False

After the basemodel is add a FFN built with the following layers:

* Dense layer 2048 neurons, activation function "relu".
* Dense layer 512 neurons, activation function "relu".
* Dropout layer 0.15.
* Output layer 5 output neurons, activation function "softmax".

In [ ]:

x1 = pretrained\_model.output

x1 = Dense(2048, activation = 'relu')(x1)

x1 = Dense(512, activation='relu')(x1)

x1 = Dropout(0.2)(x1)

pred1 = Dense(5, activation = 'softmax')(x1)

In [ ]:

model1 = Model(inputs=pretrained\_model.input, outputs=pred1)

In [ ]:

model1.summary()

Model: "model"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param # Connected to

==================================================================================================

input\_1 (InputLayer) [(None, 224, 224, 3 0 []

)]

conv1\_pad (ZeroPadding2D) (None, 230, 230, 3) 0 ['input\_1[0][0]']

conv1\_conv (Conv2D) (None, 112, 112, 64 9472 ['conv1\_pad[0][0]']

)

conv1\_bn (BatchNormalization) (None, 112, 112, 64 256 ['conv1\_conv[0][0]']

)

conv1\_relu (Activation) (None, 112, 112, 64 0 ['conv1\_bn[0][0]']

)

pool1\_pad (ZeroPadding2D) (None, 114, 114, 64 0 ['conv1\_relu[0][0]']

)

pool1\_pool (MaxPooling2D) (None, 56, 56, 64) 0 ['pool1\_pad[0][0]']

conv2\_block1\_1\_conv (Conv2D) (None, 56, 56, 64) 4160 ['pool1\_pool[0][0]']

conv2\_block1\_1\_bn (BatchNormal (None, 56, 56, 64) 256 ['conv2\_block1\_1\_conv[0][0]']

ization)

conv2\_block1\_1\_relu (Activatio (None, 56, 56, 64) 0 ['conv2\_block1\_1\_bn[0][0]']

n)

conv2\_block1\_2\_conv (Conv2D) (None, 56, 56, 64) 36928 ['conv2\_block1\_1\_relu[0][0]']

conv2\_block1\_2\_bn (BatchNormal (None, 56, 56, 64) 256 ['conv2\_block1\_2\_conv[0][0]']

ization)

conv2\_block1\_2\_relu (Activatio (None, 56, 56, 64) 0 ['conv2\_block1\_2\_bn[0][0]']

n)

conv2\_block1\_0\_conv (Conv2D) (None, 56, 56, 256) 16640 ['pool1\_pool[0][0]']

conv2\_block1\_3\_conv (Conv2D) (None, 56, 56, 256) 16640 ['conv2\_block1\_2\_relu[0][0]']

conv2\_block1\_0\_bn (BatchNormal (None, 56, 56, 256) 1024 ['conv2\_block1\_0\_conv[0][0]']

ization)

conv2\_block1\_3\_bn (BatchNormal (None, 56, 56, 256) 1024 ['conv2\_block1\_3\_conv[0][0]']

ization)

conv2\_block1\_add (Add) (None, 56, 56, 256) 0 ['conv2\_block1\_0\_bn[0][0]',

'conv2\_block1\_3\_bn[0][0]']

conv2\_block1\_out (Activation) (None, 56, 56, 256) 0 ['conv2\_block1\_add[0][0]']

conv2\_block2\_1\_conv (Conv2D) (None, 56, 56, 64) 16448 ['conv2\_block1\_out[0][0]']

conv2\_block2\_1\_bn (BatchNormal (None, 56, 56, 64) 256 ['conv2\_block2\_1\_conv[0][0]']

ization)

conv2\_block2\_1\_relu (Activatio (None, 56, 56, 64) 0 ['conv2\_block2\_1\_bn[0][0]']

n)

conv2\_block2\_2\_conv (Conv2D) (None, 56, 56, 64) 36928 ['conv2\_block2\_1\_relu[0][0]']

conv2\_block2\_2\_bn (BatchNormal (None, 56, 56, 64) 256 ['conv2\_block2\_2\_conv[0][0]']

ization)

conv2\_block2\_2\_relu (Activatio (None, 56, 56, 64) 0 ['conv2\_block2\_2\_bn[0][0]']

n)

conv2\_block2\_3\_conv (Conv2D) (None, 56, 56, 256) 16640 ['conv2\_block2\_2\_relu[0][0]']

conv2\_block2\_3\_bn (BatchNormal (None, 56, 56, 256) 1024 ['conv2\_block2\_3\_conv[0][0]']

ization)

conv2\_block2\_add (Add) (None, 56, 56, 256) 0 ['conv2\_block1\_out[0][0]',

'conv2\_block2\_3\_bn[0][0]']

conv2\_block2\_out (Activation) (None, 56, 56, 256) 0 ['conv2\_block2\_add[0][0]']

conv2\_block3\_1\_conv (Conv2D) (None, 56, 56, 64) 16448 ['conv2\_block2\_out[0][0]']

conv2\_block3\_1\_bn (BatchNormal (None, 56, 56, 64) 256 ['conv2\_block3\_1\_conv[0][0]']

ization)

conv2\_block3\_1\_relu (Activatio (None, 56, 56, 64) 0 ['conv2\_block3\_1\_bn[0][0]']

n)

conv2\_block3\_2\_conv (Conv2D) (None, 56, 56, 64) 36928 ['conv2\_block3\_1\_relu[0][0]']

conv2\_block3\_2\_bn (BatchNormal (None, 56, 56, 64) 256 ['conv2\_block3\_2\_conv[0][0]']

ization)

conv2\_block3\_2\_relu (Activatio (None, 56, 56, 64) 0 ['conv2\_block3\_2\_bn[0][0]']

n)

conv2\_block3\_3\_conv (Conv2D) (None, 56, 56, 256) 16640 ['conv2\_block3\_2\_relu[0][0]']

conv2\_block3\_3\_bn (BatchNormal (None, 56, 56, 256) 1024 ['conv2\_block3\_3\_conv[0][0]']

ization)

conv2\_block3\_add (Add) (None, 56, 56, 256) 0 ['conv2\_block2\_out[0][0]',

'conv2\_block3\_3\_bn[0][0]']

conv2\_block3\_out (Activation) (None, 56, 56, 256) 0 ['conv2\_block3\_add[0][0]']

conv3\_block1\_1\_conv (Conv2D) (None, 28, 28, 128) 32896 ['conv2\_block3\_out[0][0]']

conv3\_block1\_1\_bn (BatchNormal (None, 28, 28, 128) 512 ['conv3\_block1\_1\_conv[0][0]']

ization)

conv3\_block1\_1\_relu (Activatio (None, 28, 28, 128) 0 ['conv3\_block1\_1\_bn[0][0]']

n)

conv3\_block1\_2\_conv (Conv2D) (None, 28, 28, 128) 147584 ['conv3\_block1\_1\_relu[0][0]']

conv3\_block1\_2\_bn (BatchNormal (None, 28, 28, 128) 512 ['conv3\_block1\_2\_conv[0][0]']

ization)

conv3\_block1\_2\_relu (Activatio (None, 28, 28, 128) 0 ['conv3\_block1\_2\_bn[0][0]']

n)

conv3\_block1\_0\_conv (Conv2D) (None, 28, 28, 512) 131584 ['conv2\_block3\_out[0][0]']

conv3\_block1\_3\_conv (Conv2D) (None, 28, 28, 512) 66048 ['conv3\_block1\_2\_relu[0][0]']

conv3\_block1\_0\_bn (BatchNormal (None, 28, 28, 512) 2048 ['conv3\_block1\_0\_conv[0][0]']

ization)

conv3\_block1\_3\_bn (BatchNormal (None, 28, 28, 512) 2048 ['conv3\_block1\_3\_conv[0][0]']

ization)

conv3\_block1\_add (Add) (None, 28, 28, 512) 0 ['conv3\_block1\_0\_bn[0][0]',

'conv3\_block1\_3\_bn[0][0]']

conv3\_block1\_out (Activation) (None, 28, 28, 512) 0 ['conv3\_block1\_add[0][0]']

conv3\_block2\_1\_conv (Conv2D) (None, 28, 28, 128) 65664 ['conv3\_block1\_out[0][0]']

conv3\_block2\_1\_bn (BatchNormal (None, 28, 28, 128) 512 ['conv3\_block2\_1\_conv[0][0]']

ization)

conv3\_block2\_1\_relu (Activatio (None, 28, 28, 128) 0 ['conv3\_block2\_1\_bn[0][0]']

n)

conv3\_block2\_2\_conv (Conv2D) (None, 28, 28, 128) 147584 ['conv3\_block2\_1\_relu[0][0]']

conv3\_block2\_2\_bn (BatchNormal (None, 28, 28, 128) 512 ['conv3\_block2\_2\_conv[0][0]']

ization)

conv3\_block2\_2\_relu (Activatio (None, 28, 28, 128) 0 ['conv3\_block2\_2\_bn[0][0]']

n)

conv3\_block2\_3\_conv (Conv2D) (None, 28, 28, 512) 66048 ['conv3\_block2\_2\_relu[0][0]']

conv3\_block2\_3\_bn (BatchNormal (None, 28, 28, 512) 2048 ['conv3\_block2\_3\_conv[0][0]']

ization)

conv3\_block2\_add (Add) (None, 28, 28, 512) 0 ['conv3\_block1\_out[0][0]',

'conv3\_block2\_3\_bn[0][0]']

conv3\_block2\_out (Activation) (None, 28, 28, 512) 0 ['conv3\_block2\_add[0][0]']

conv3\_block3\_1\_conv (Conv2D) (None, 28, 28, 128) 65664 ['conv3\_block2\_out[0][0]']

conv3\_block3\_1\_bn (BatchNormal (None, 28, 28, 128) 512 ['conv3\_block3\_1\_conv[0][0]']

ization)

conv3\_block3\_1\_relu (Activatio (None, 28, 28, 128) 0 ['conv3\_block3\_1\_bn[0][0]']

n)

conv3\_block3\_2\_conv (Conv2D) (None, 28, 28, 128) 147584 ['conv3\_block3\_1\_relu[0][0]']

conv3\_block3\_2\_bn (BatchNormal (None, 28, 28, 128) 512 ['conv3\_block3\_2\_conv[0][0]']

ization)

conv3\_block3\_2\_relu (Activatio (None, 28, 28, 128) 0 ['conv3\_block3\_2\_bn[0][0]']

n)

conv3\_block3\_3\_conv (Conv2D) (None, 28, 28, 512) 66048 ['conv3\_block3\_2\_relu[0][0]']

conv3\_block3\_3\_bn (BatchNormal (None, 28, 28, 512) 2048 ['conv3\_block3\_3\_conv[0][0]']

ization)

conv3\_block3\_add (Add) (None, 28, 28, 512) 0 ['conv3\_block2\_out[0][0]',

'conv3\_block3\_3\_bn[0][0]']

conv3\_block3\_out (Activation) (None, 28, 28, 512) 0 ['conv3\_block3\_add[0][0]']

conv3\_block4\_1\_conv (Conv2D) (None, 28, 28, 128) 65664 ['conv3\_block3\_out[0][0]']

conv3\_block4\_1\_bn (BatchNormal (None, 28, 28, 128) 512 ['conv3\_block4\_1\_conv[0][0]']

ization)

conv3\_block4\_1\_relu (Activatio (None, 28, 28, 128) 0 ['conv3\_block4\_1\_bn[0][0]']

n)

conv3\_block4\_2\_conv (Conv2D) (None, 28, 28, 128) 147584 ['conv3\_block4\_1\_relu[0][0]']

conv3\_block4\_2\_bn (BatchNormal (None, 28, 28, 128) 512 ['conv3\_block4\_2\_conv[0][0]']

ization)

conv3\_block4\_2\_relu (Activatio (None, 28, 28, 128) 0 ['conv3\_block4\_2\_bn[0][0]']

n)

conv3\_block4\_3\_conv (Conv2D) (None, 28, 28, 512) 66048 ['conv3\_block4\_2\_relu[0][0]']

conv3\_block4\_3\_bn (BatchNormal (None, 28, 28, 512) 2048 ['conv3\_block4\_3\_conv[0][0]']

ization)

conv3\_block4\_add (Add) (None, 28, 28, 512) 0 ['conv3\_block3\_out[0][0]',

'conv3\_block4\_3\_bn[0][0]']

conv3\_block4\_out (Activation) (None, 28, 28, 512) 0 ['conv3\_block4\_add[0][0]']

conv4\_block1\_1\_conv (Conv2D) (None, 14, 14, 256) 131328 ['conv3\_block4\_out[0][0]']

conv4\_block1\_1\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block1\_1\_conv[0][0]']

ization)

conv4\_block1\_1\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block1\_1\_bn[0][0]']

n)

conv4\_block1\_2\_conv (Conv2D) (None, 14, 14, 256) 590080 ['conv4\_block1\_1\_relu[0][0]']

conv4\_block1\_2\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block1\_2\_conv[0][0]']

ization)

conv4\_block1\_2\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block1\_2\_bn[0][0]']

n)

conv4\_block1\_0\_conv (Conv2D) (None, 14, 14, 1024 525312 ['conv3\_block4\_out[0][0]']

)

conv4\_block1\_3\_conv (Conv2D) (None, 14, 14, 1024 263168 ['conv4\_block1\_2\_relu[0][0]']

)

conv4\_block1\_0\_bn (BatchNormal (None, 14, 14, 1024 4096 ['conv4\_block1\_0\_conv[0][0]']

ization) )

conv4\_block1\_3\_bn (BatchNormal (None, 14, 14, 1024 4096 ['conv4\_block1\_3\_conv[0][0]']

ization) )

conv4\_block1\_add (Add) (None, 14, 14, 1024 0 ['conv4\_block1\_0\_bn[0][0]',

) 'conv4\_block1\_3\_bn[0][0]']

conv4\_block1\_out (Activation) (None, 14, 14, 1024 0 ['conv4\_block1\_add[0][0]']

)

conv4\_block2\_1\_conv (Conv2D) (None, 14, 14, 256) 262400 ['conv4\_block1\_out[0][0]']

conv4\_block2\_1\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block2\_1\_conv[0][0]']

ization)

conv4\_block2\_1\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block2\_1\_bn[0][0]']

n)

conv4\_block2\_2\_conv (Conv2D) (None, 14, 14, 256) 590080 ['conv4\_block2\_1\_relu[0][0]']

conv4\_block2\_2\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block2\_2\_conv[0][0]']

ization)

conv4\_block2\_2\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block2\_2\_bn[0][0]']

n)

conv4\_block2\_3\_conv (Conv2D) (None, 14, 14, 1024 263168 ['conv4\_block2\_2\_relu[0][0]']

)

conv4\_block2\_3\_bn (BatchNormal (None, 14, 14, 1024 4096 ['conv4\_block2\_3\_conv[0][0]']

ization) )

conv4\_block2\_add (Add) (None, 14, 14, 1024 0 ['conv4\_block1\_out[0][0]',

) 'conv4\_block2\_3\_bn[0][0]']

conv4\_block2\_out (Activation) (None, 14, 14, 1024 0 ['conv4\_block2\_add[0][0]']

)

conv4\_block3\_1\_conv (Conv2D) (None, 14, 14, 256) 262400 ['conv4\_block2\_out[0][0]']

conv4\_block3\_1\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block3\_1\_conv[0][0]']

ization)

conv4\_block3\_1\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block3\_1\_bn[0][0]']

n)

conv4\_block3\_2\_conv (Conv2D) (None, 14, 14, 256) 590080 ['conv4\_block3\_1\_relu[0][0]']

conv4\_block3\_2\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block3\_2\_conv[0][0]']

ization)

conv4\_block3\_2\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block3\_2\_bn[0][0]']

n)

conv4\_block3\_3\_conv (Conv2D) (None, 14, 14, 1024 263168 ['conv4\_block3\_2\_relu[0][0]']

)

conv4\_block3\_3\_bn (BatchNormal (None, 14, 14, 1024 4096 ['conv4\_block3\_3\_conv[0][0]']

ization) )

conv4\_block3\_add (Add) (None, 14, 14, 1024 0 ['conv4\_block2\_out[0][0]',

) 'conv4\_block3\_3\_bn[0][0]']

conv4\_block3\_out (Activation) (None, 14, 14, 1024 0 ['conv4\_block3\_add[0][0]']

)

conv4\_block4\_1\_conv (Conv2D) (None, 14, 14, 256) 262400 ['conv4\_block3\_out[0][0]']

conv4\_block4\_1\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block4\_1\_conv[0][0]']

ization)

conv4\_block4\_1\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block4\_1\_bn[0][0]']

n)

conv4\_block4\_2\_conv (Conv2D) (None, 14, 14, 256) 590080 ['conv4\_block4\_1\_relu[0][0]']

conv4\_block4\_2\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block4\_2\_conv[0][0]']

ization)

conv4\_block4\_2\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block4\_2\_bn[0][0]']

n)

conv4\_block4\_3\_conv (Conv2D) (None, 14, 14, 1024 263168 ['conv4\_block4\_2\_relu[0][0]']

)

conv4\_block4\_3\_bn (BatchNormal (None, 14, 14, 1024 4096 ['conv4\_block4\_3\_conv[0][0]']

ization) )

conv4\_block4\_add (Add) (None, 14, 14, 1024 0 ['conv4\_block3\_out[0][0]',

) 'conv4\_block4\_3\_bn[0][0]']

conv4\_block4\_out (Activation) (None, 14, 14, 1024 0 ['conv4\_block4\_add[0][0]']

)

conv4\_block5\_1\_conv (Conv2D) (None, 14, 14, 256) 262400 ['conv4\_block4\_out[0][0]']

conv4\_block5\_1\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block5\_1\_conv[0][0]']

ization)

conv4\_block5\_1\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block5\_1\_bn[0][0]']

n)

conv4\_block5\_2\_conv (Conv2D) (None, 14, 14, 256) 590080 ['conv4\_block5\_1\_relu[0][0]']

conv4\_block5\_2\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block5\_2\_conv[0][0]']

ization)

conv4\_block5\_2\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block5\_2\_bn[0][0]']

n)

conv4\_block5\_3\_conv (Conv2D) (None, 14, 14, 1024 263168 ['conv4\_block5\_2\_relu[0][0]']

)

conv4\_block5\_3\_bn (BatchNormal (None, 14, 14, 1024 4096 ['conv4\_block5\_3\_conv[0][0]']

ization) )

conv4\_block5\_add (Add) (None, 14, 14, 1024 0 ['conv4\_block4\_out[0][0]',

) 'conv4\_block5\_3\_bn[0][0]']

conv4\_block5\_out (Activation) (None, 14, 14, 1024 0 ['conv4\_block5\_add[0][0]']

)

conv4\_block6\_1\_conv (Conv2D) (None, 14, 14, 256) 262400 ['conv4\_block5\_out[0][0]']

conv4\_block6\_1\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block6\_1\_conv[0][0]']

ization)

conv4\_block6\_1\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block6\_1\_bn[0][0]']

n)

conv4\_block6\_2\_conv (Conv2D) (None, 14, 14, 256) 590080 ['conv4\_block6\_1\_relu[0][0]']

conv4\_block6\_2\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block6\_2\_conv[0][0]']

ization)

conv4\_block6\_2\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block6\_2\_bn[0][0]']

n)

conv4\_block6\_3\_conv (Conv2D) (None, 14, 14, 1024 263168 ['conv4\_block6\_2\_relu[0][0]']

)

conv4\_block6\_3\_bn (BatchNormal (None, 14, 14, 1024 4096 ['conv4\_block6\_3\_conv[0][0]']

ization) )

conv4\_block6\_add (Add) (None, 14, 14, 1024 0 ['conv4\_block5\_out[0][0]',

) 'conv4\_block6\_3\_bn[0][0]']

conv4\_block6\_out (Activation) (None, 14, 14, 1024 0 ['conv4\_block6\_add[0][0]']

)

conv5\_block1\_1\_conv (Conv2D) (None, 7, 7, 512) 524800 ['conv4\_block6\_out[0][0]']

conv5\_block1\_1\_bn (BatchNormal (None, 7, 7, 512) 2048 ['conv5\_block1\_1\_conv[0][0]']

ization)

conv5\_block1\_1\_relu (Activatio (None, 7, 7, 512) 0 ['conv5\_block1\_1\_bn[0][0]']

n)

conv5\_block1\_2\_conv (Conv2D) (None, 7, 7, 512) 2359808 ['conv5\_block1\_1\_relu[0][0]']

conv5\_block1\_2\_bn (BatchNormal (None, 7, 7, 512) 2048 ['conv5\_block1\_2\_conv[0][0]']

ization)

conv5\_block1\_2\_relu (Activatio (None, 7, 7, 512) 0 ['conv5\_block1\_2\_bn[0][0]']

n)

conv5\_block1\_0\_conv (Conv2D) (None, 7, 7, 2048) 2099200 ['conv4\_block6\_out[0][0]']

conv5\_block1\_3\_conv (Conv2D) (None, 7, 7, 2048) 1050624 ['conv5\_block1\_2\_relu[0][0]']

conv5\_block1\_0\_bn (BatchNormal (None, 7, 7, 2048) 8192 ['conv5\_block1\_0\_conv[0][0]']

ization)

conv5\_block1\_3\_bn (BatchNormal (None, 7, 7, 2048) 8192 ['conv5\_block1\_3\_conv[0][0]']

ization)

conv5\_block1\_add (Add) (None, 7, 7, 2048) 0 ['conv5\_block1\_0\_bn[0][0]',

'conv5\_block1\_3\_bn[0][0]']

conv5\_block1\_out (Activation) (None, 7, 7, 2048) 0 ['conv5\_block1\_add[0][0]']

conv5\_block2\_1\_conv (Conv2D) (None, 7, 7, 512) 1049088 ['conv5\_block1\_out[0][0]']

conv5\_block2\_1\_bn (BatchNormal (None, 7, 7, 512) 2048 ['conv5\_block2\_1\_conv[0][0]']

ization)

conv5\_block2\_1\_relu (Activatio (None, 7, 7, 512) 0 ['conv5\_block2\_1\_bn[0][0]']

n)

conv5\_block2\_2\_conv (Conv2D) (None, 7, 7, 512) 2359808 ['conv5\_block2\_1\_relu[0][0]']

conv5\_block2\_2\_bn (BatchNormal (None, 7, 7, 512) 2048 ['conv5\_block2\_2\_conv[0][0]']

ization)

conv5\_block2\_2\_relu (Activatio (None, 7, 7, 512) 0 ['conv5\_block2\_2\_bn[0][0]']

n)

conv5\_block2\_3\_conv (Conv2D) (None, 7, 7, 2048) 1050624 ['conv5\_block2\_2\_relu[0][0]']

conv5\_block2\_3\_bn (BatchNormal (None, 7, 7, 2048) 8192 ['conv5\_block2\_3\_conv[0][0]']

ization)

conv5\_block2\_add (Add) (None, 7, 7, 2048) 0 ['conv5\_block1\_out[0][0]',

'conv5\_block2\_3\_bn[0][0]']

conv5\_block2\_out (Activation) (None, 7, 7, 2048) 0 ['conv5\_block2\_add[0][0]']

conv5\_block3\_1\_conv (Conv2D) (None, 7, 7, 512) 1049088 ['conv5\_block2\_out[0][0]']

conv5\_block3\_1\_bn (BatchNormal (None, 7, 7, 512) 2048 ['conv5\_block3\_1\_conv[0][0]']

ization)

conv5\_block3\_1\_relu (Activatio (None, 7, 7, 512) 0 ['conv5\_block3\_1\_bn[0][0]']

n)

conv5\_block3\_2\_conv (Conv2D) (None, 7, 7, 512) 2359808 ['conv5\_block3\_1\_relu[0][0]']

conv5\_block3\_2\_bn (BatchNormal (None, 7, 7, 512) 2048 ['conv5\_block3\_2\_conv[0][0]']

ization)

conv5\_block3\_2\_relu (Activatio (None, 7, 7, 512) 0 ['conv5\_block3\_2\_bn[0][0]']

n)

conv5\_block3\_3\_conv (Conv2D) (None, 7, 7, 2048) 1050624 ['conv5\_block3\_2\_relu[0][0]']

conv5\_block3\_3\_bn (BatchNormal (None, 7, 7, 2048) 8192 ['conv5\_block3\_3\_conv[0][0]']

ization)

conv5\_block3\_add (Add) (None, 7, 7, 2048) 0 ['conv5\_block2\_out[0][0]',

'conv5\_block3\_3\_bn[0][0]']

conv5\_block3\_out (Activation) (None, 7, 7, 2048) 0 ['conv5\_block3\_add[0][0]']

avg\_pool (GlobalAveragePooling (None, 2048) 0 ['conv5\_block3\_out[0][0]']

2D)

dense (Dense) (None, 2048) 4196352 ['avg\_pool[0][0]']

dense\_1 (Dense) (None, 512) 1049088 ['dense[0][0]']

dropout (Dropout) (None, 512) 0 ['dense\_1[0][0]']

dense\_2 (Dense) (None, 5) 2565 ['dropout[0][0]']

==================================================================================================

Total params: 28,835,717

Trainable params: 5,248,005

Non-trainable params: 23,587,712

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The optimizer used is Adam, the loss function is the categorical crossentropy and the metric is the Accuracy:

In [ ]:

model1.compile(

optimizer='adam',

loss='categorical\_crossentropy',

metrics=['accuracy'])

In [ ]:

history = model1.fit\_generator(

generator=train\_generator,

epochs=25,

validation\_data=validation\_generator)

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:4: UserWarning: `Model.fit\_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.

after removing the cwd from sys.path.

Epoch 1/25

104/104 [==============================] - 834s 8s/step - loss: 1.2993 - accuracy: 0.4524 - val\_loss: 0.8563 - val\_accuracy: 0.5638

Epoch 2/25

104/104 [==============================] - 48s 458ms/step - loss: 0.7202 - accuracy: 0.6825 - val\_loss: 0.6224 - val\_accuracy: 0.7290

Epoch 3/25

104/104 [==============================] - 48s 463ms/step - loss: 0.6732 - accuracy: 0.6977 - val\_loss: 0.6652 - val\_accuracy: 0.6731

Epoch 4/25

104/104 [==============================] - 48s 459ms/step - loss: 0.6202 - accuracy: 0.7341 - val\_loss: 0.5169 - val\_accuracy: 0.7886

Epoch 5/25

104/104 [==============================] - 48s 457ms/step - loss: 0.5678 - accuracy: 0.7562 - val\_loss: 0.4289 - val\_accuracy: 0.8360

Epoch 6/25

104/104 [==============================] - 49s 468ms/step - loss: 0.5513 - accuracy: 0.7547 - val\_loss: 0.6192 - val\_accuracy: 0.7060

Epoch 7/25

104/104 [==============================] - 48s 458ms/step - loss: 0.6160 - accuracy: 0.7353 - val\_loss: 0.5333 - val\_accuracy: 0.7776

Epoch 8/25

104/104 [==============================] - 48s 461ms/step - loss: 0.5018 - accuracy: 0.7811 - val\_loss: 0.4869 - val\_accuracy: 0.8019

Epoch 9/25

104/104 [==============================] - 47s 455ms/step - loss: 0.4638 - accuracy: 0.8017 - val\_loss: 0.5886 - val\_accuracy: 0.7776

Epoch 10/25

104/104 [==============================] - 47s 453ms/step - loss: 0.5273 - accuracy: 0.7805 - val\_loss: 0.3695 - val\_accuracy: 0.8469

Epoch 11/25

104/104 [==============================] - 47s 454ms/step - loss: 0.4620 - accuracy: 0.8065 - val\_loss: 0.4296 - val\_accuracy: 0.8384

Epoch 12/25

104/104 [==============================] - 49s 467ms/step - loss: 0.4591 - accuracy: 0.8038 - val\_loss: 0.5315 - val\_accuracy: 0.8007

Epoch 13/25

104/104 [==============================] - 47s 451ms/step - loss: 0.4641 - accuracy: 0.7962 - val\_loss: 0.4805 - val\_accuracy: 0.8141

Epoch 14/25

104/104 [==============================] - 48s 457ms/step - loss: 0.4551 - accuracy: 0.8032 - val\_loss: 0.4366 - val\_accuracy: 0.8275

Epoch 15/25

104/104 [==============================] - 48s 461ms/step - loss: 0.4429 - accuracy: 0.8047 - val\_loss: 0.4806 - val\_accuracy: 0.7910

Epoch 16/25

104/104 [==============================] - 47s 456ms/step - loss: 0.4990 - accuracy: 0.7944 - val\_loss: 0.4429 - val\_accuracy: 0.8153

Epoch 17/25

104/104 [==============================] - 47s 453ms/step - loss: 0.4371 - accuracy: 0.8138 - val\_loss: 0.3649 - val\_accuracy: 0.8651

Epoch 18/25

104/104 [==============================] - 48s 463ms/step - loss: 0.4962 - accuracy: 0.7844 - val\_loss: 0.3453 - val\_accuracy: 0.8445

Epoch 19/25

104/104 [==============================] - 47s 452ms/step - loss: 0.4330 - accuracy: 0.8132 - val\_loss: 0.3585 - val\_accuracy: 0.8372

Epoch 20/25

104/104 [==============================] - 47s 455ms/step - loss: 0.4208 - accuracy: 0.8175 - val\_loss: 0.4213 - val\_accuracy: 0.8262

Epoch 21/25

104/104 [==============================] - 47s 448ms/step - loss: 0.4042 - accuracy: 0.8281 - val\_loss: 0.4602 - val\_accuracy: 0.8019

Epoch 22/25

104/104 [==============================] - 47s 455ms/step - loss: 0.4512 - accuracy: 0.8041 - val\_loss: 0.3836 - val\_accuracy: 0.8554

Epoch 23/25

104/104 [==============================] - 47s 450ms/step - loss: 0.4217 - accuracy: 0.8272 - val\_loss: 0.3128 - val\_accuracy: 0.8870

Epoch 24/25

104/104 [==============================] - 48s 463ms/step - loss: 0.4205 - accuracy: 0.8169 - val\_loss: 0.4742 - val\_accuracy: 0.7910

Epoch 25/25

104/104 [==============================] - 47s 455ms/step - loss: 0.3989 - accuracy: 0.8244 - val\_loss: 0.4153 - val\_accuracy: 0.8238

In [ ]:

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs = range(1, len(acc) + 1)

sns.set\_style("whitegrid")

plt.title('Training and validation accuracy')

plt.plot(epochs, acc, 'red', label='Training acc')

plt.plot(epochs, val\_acc, 'blue', label='Validation acc')

plt.legend()

plt.figure()

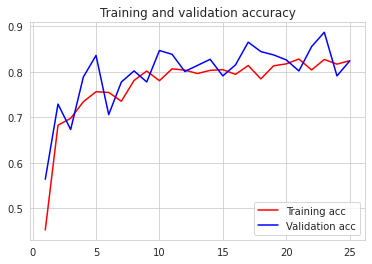
plt.title('Training and validation loss')

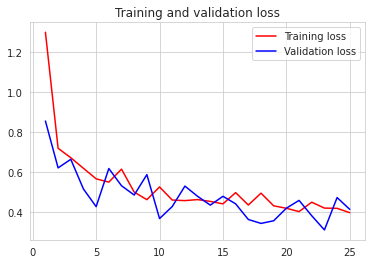
plt.plot(epochs, loss, 'red', label='Training loss')

plt.plot(epochs, val\_loss, 'blue', label='Validation loss')

plt.legend()

plt.show()





The results show a strong overfitting on the train set, Traing loss is always under the Valdidation curve, the same is for the accuracy. Another consideration is that the net seems very unstable. Those consideration are keep to improve the net in the following steps.

In [ ]:

rice\_classes=os.listdir(base\_path)

print(rice\_classes)

['Arborio', 'Basmati', 'Ipsala', 'Jasmine', 'Karacadag']

In [ ]:

Y\_pred = model1.predict(validation\_generator)

y\_pred = np.argmax(Y\_pred, axis=1)

report1 = classification\_report(validation\_generator.classes, y\_pred, target\_names=rice\_classes, output\_dict=True)

df1 = pd.DataFrame(report1).transpose()

df1

Out[ ]:

|  | **precision** | **recall** | **f1-score** | **support** |
| --- | --- | --- | --- | --- |
| **Arborio** | 0.744186 | 0.492308 | 0.592593 | 130.000000 |
| **Basmati** | 0.844037 | 0.978723 | 0.906404 | 188.000000 |
| **Ipsala** | 0.893023 | 1.000000 | 0.943489 | 192.000000 |
| **Jasmine** | 0.981481 | 0.588889 | 0.736111 | 180.000000 |
| **Karacadag** | 0.673469 | 0.992481 | 0.802432 | 133.000000 |
| **accuracy** | 0.823815 | 0.823815 | 0.823815 | 0.823815 |
| **macro avg** | 0.827239 | 0.810480 | 0.796206 | 823.000000 |
| **weighted avg** | 0.842189 | 0.823815 | 0.811439 | 823.000000 |

### **2.2) ResNet50v2**

To improve the previous network the batch size is set to 64 and a preprocessing function it's added to better adapt the data to the ResNet50. it's defined a more complex FFN to try to achive a higher performace on the validation set

In [ ]:

size=224

channels=3

batch\_size = 64

num\_classes = 5

train\_processing = ImageDataGenerator(preprocessing\_function=preprocess\_input\_resnet50,

validation\_split=0.2,

rotation\_range = 25,

width\_shift\_range = .2,

height\_shift\_range = .2,

horizontal\_flip = True,

zoom\_range = .2)

validation\_datagen = ImageDataGenerator(preprocessing\_function=preprocess\_input\_resnet50,

validation\_split=0.2)

train\_generator = train\_processing.flow\_from\_directory (base\_path,

target\_size=(size, size),

batch\_size=batch\_size,

shuffle=True,

class\_mode='categorical',

subset='training',

color\_mode="rgb",

interpolation='nearest')

validation\_generator = validation\_datagen.flow\_from\_directory(base\_path,

target\_size=(size,size),

subset='validation',

color\_mode="rgb",

batch\_size=batch\_size,

class\_mode='categorical',

shuffle=False,

interpolation='nearest')

Found 3840 images belonging to 5 classes.

Found 960 images belonging to 5 classes.

In [ ]:

for image\_batch, labels\_batch in train\_generator:

print(image\_batch.shape)

print(labels\_batch.shape)

break

(64, 224, 224, 3)

(64, 5)

In [ ]:

pretrained\_model = resnet50.ResNet50(

weights='imagenet',

include\_top=False,

input\_shape=(size, size, 3),

pooling='avg',

)

In [ ]:

layer\_dict = dict([(layer.name, layer) for layer in pretrained\_model.layers])

for layer in pretrained\_model.layers:

layer.trainable = False

After the basemodel is add a FFN built with the following layers:

* Dense layer 2048 neurons, activation function "relu".
* Dense layer 512 neurons, activation function "relu".
* Dropout layer 0.15.
* Dense layer 128 neurons, activation function "relu".
* Dense layer 64 neurons, activation function "relu".
* Dropout layer 0.15.
* Dense layer 32 neurons, activation function "relu".
* Dense layer 16 neurons, activation function "relu".
* Output layer 5 output neurons, activation function "softmax".

In [ ]:

x2 = pretrained\_model.output

x2 = Dense(2048,'relu')(x2)

x2 = Dense(512,'relu')(x2)

x2 = Dropout(.15)(x2)

x2 = Dense(128,'relu')(x2)

x2 = Dense(64,'relu')(x2)

x2 = Dropout(.15)(x2)

x2 = Dense(32,'relu')(x2)

x2 = Dense(16,'relu')(x2)

pred2 = Dense(5,'softmax')(x2)

model2 = Model(inputs=pretrained\_model.input, outputs=pred2)

In [ ]:

model2.summary()

Model: "model\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param # Connected to

==================================================================================================

input\_2 (InputLayer) [(None, 224, 224, 3 0 []

)]

conv1\_pad (ZeroPadding2D) (None, 230, 230, 3) 0 ['input\_2[0][0]']

conv1\_conv (Conv2D) (None, 112, 112, 64 9472 ['conv1\_pad[0][0]']

)

conv1\_bn (BatchNormalization) (None, 112, 112, 64 256 ['conv1\_conv[0][0]']

)

conv1\_relu (Activation) (None, 112, 112, 64 0 ['conv1\_bn[0][0]']

)

pool1\_pad (ZeroPadding2D) (None, 114, 114, 64 0 ['conv1\_relu[0][0]']

)

pool1\_pool (MaxPooling2D) (None, 56, 56, 64) 0 ['pool1\_pad[0][0]']

conv2\_block1\_1\_conv (Conv2D) (None, 56, 56, 64) 4160 ['pool1\_pool[0][0]']

conv2\_block1\_1\_bn (BatchNormal (None, 56, 56, 64) 256 ['conv2\_block1\_1\_conv[0][0]']

ization)

conv2\_block1\_1\_relu (Activatio (None, 56, 56, 64) 0 ['conv2\_block1\_1\_bn[0][0]']

n)

conv2\_block1\_2\_conv (Conv2D) (None, 56, 56, 64) 36928 ['conv2\_block1\_1\_relu[0][0]']

conv2\_block1\_2\_bn (BatchNormal (None, 56, 56, 64) 256 ['conv2\_block1\_2\_conv[0][0]']

ization)

conv2\_block1\_2\_relu (Activatio (None, 56, 56, 64) 0 ['conv2\_block1\_2\_bn[0][0]']

n)

conv2\_block1\_0\_conv (Conv2D) (None, 56, 56, 256) 16640 ['pool1\_pool[0][0]']

conv2\_block1\_3\_conv (Conv2D) (None, 56, 56, 256) 16640 ['conv2\_block1\_2\_relu[0][0]']

conv2\_block1\_0\_bn (BatchNormal (None, 56, 56, 256) 1024 ['conv2\_block1\_0\_conv[0][0]']

ization)

conv2\_block1\_3\_bn (BatchNormal (None, 56, 56, 256) 1024 ['conv2\_block1\_3\_conv[0][0]']

ization)

conv2\_block1\_add (Add) (None, 56, 56, 256) 0 ['conv2\_block1\_0\_bn[0][0]',

'conv2\_block1\_3\_bn[0][0]']

conv2\_block1\_out (Activation) (None, 56, 56, 256) 0 ['conv2\_block1\_add[0][0]']

conv2\_block2\_1\_conv (Conv2D) (None, 56, 56, 64) 16448 ['conv2\_block1\_out[0][0]']

conv2\_block2\_1\_bn (BatchNormal (None, 56, 56, 64) 256 ['conv2\_block2\_1\_conv[0][0]']

ization)

conv2\_block2\_1\_relu (Activatio (None, 56, 56, 64) 0 ['conv2\_block2\_1\_bn[0][0]']

n)

conv2\_block2\_2\_conv (Conv2D) (None, 56, 56, 64) 36928 ['conv2\_block2\_1\_relu[0][0]']

conv2\_block2\_2\_bn (BatchNormal (None, 56, 56, 64) 256 ['conv2\_block2\_2\_conv[0][0]']

ization)

conv2\_block2\_2\_relu (Activatio (None, 56, 56, 64) 0 ['conv2\_block2\_2\_bn[0][0]']

n)

conv2\_block2\_3\_conv (Conv2D) (None, 56, 56, 256) 16640 ['conv2\_block2\_2\_relu[0][0]']

conv2\_block2\_3\_bn (BatchNormal (None, 56, 56, 256) 1024 ['conv2\_block2\_3\_conv[0][0]']

ization)

conv2\_block2\_add (Add) (None, 56, 56, 256) 0 ['conv2\_block1\_out[0][0]',

'conv2\_block2\_3\_bn[0][0]']

conv2\_block2\_out (Activation) (None, 56, 56, 256) 0 ['conv2\_block2\_add[0][0]']

conv2\_block3\_1\_conv (Conv2D) (None, 56, 56, 64) 16448 ['conv2\_block2\_out[0][0]']

conv2\_block3\_1\_bn (BatchNormal (None, 56, 56, 64) 256 ['conv2\_block3\_1\_conv[0][0]']

ization)

conv2\_block3\_1\_relu (Activatio (None, 56, 56, 64) 0 ['conv2\_block3\_1\_bn[0][0]']

n)

conv2\_block3\_2\_conv (Conv2D) (None, 56, 56, 64) 36928 ['conv2\_block3\_1\_relu[0][0]']

conv2\_block3\_2\_bn (BatchNormal (None, 56, 56, 64) 256 ['conv2\_block3\_2\_conv[0][0]']

ization)

conv2\_block3\_2\_relu (Activatio (None, 56, 56, 64) 0 ['conv2\_block3\_2\_bn[0][0]']

n)

conv2\_block3\_3\_conv (Conv2D) (None, 56, 56, 256) 16640 ['conv2\_block3\_2\_relu[0][0]']

conv2\_block3\_3\_bn (BatchNormal (None, 56, 56, 256) 1024 ['conv2\_block3\_3\_conv[0][0]']

ization)

conv2\_block3\_add (Add) (None, 56, 56, 256) 0 ['conv2\_block2\_out[0][0]',

'conv2\_block3\_3\_bn[0][0]']

conv2\_block3\_out (Activation) (None, 56, 56, 256) 0 ['conv2\_block3\_add[0][0]']

conv3\_block1\_1\_conv (Conv2D) (None, 28, 28, 128) 32896 ['conv2\_block3\_out[0][0]']

conv3\_block1\_1\_bn (BatchNormal (None, 28, 28, 128) 512 ['conv3\_block1\_1\_conv[0][0]']

ization)

conv3\_block1\_1\_relu (Activatio (None, 28, 28, 128) 0 ['conv3\_block1\_1\_bn[0][0]']

n)

conv3\_block1\_2\_conv (Conv2D) (None, 28, 28, 128) 147584 ['conv3\_block1\_1\_relu[0][0]']

conv3\_block1\_2\_bn (BatchNormal (None, 28, 28, 128) 512 ['conv3\_block1\_2\_conv[0][0]']

ization)

conv3\_block1\_2\_relu (Activatio (None, 28, 28, 128) 0 ['conv3\_block1\_2\_bn[0][0]']

n)

conv3\_block1\_0\_conv (Conv2D) (None, 28, 28, 512) 131584 ['conv2\_block3\_out[0][0]']

conv3\_block1\_3\_conv (Conv2D) (None, 28, 28, 512) 66048 ['conv3\_block1\_2\_relu[0][0]']

conv3\_block1\_0\_bn (BatchNormal (None, 28, 28, 512) 2048 ['conv3\_block1\_0\_conv[0][0]']

ization)

conv3\_block1\_3\_bn (BatchNormal (None, 28, 28, 512) 2048 ['conv3\_block1\_3\_conv[0][0]']

ization)

conv3\_block1\_add (Add) (None, 28, 28, 512) 0 ['conv3\_block1\_0\_bn[0][0]',

'conv3\_block1\_3\_bn[0][0]']

conv3\_block1\_out (Activation) (None, 28, 28, 512) 0 ['conv3\_block1\_add[0][0]']

conv3\_block2\_1\_conv (Conv2D) (None, 28, 28, 128) 65664 ['conv3\_block1\_out[0][0]']

conv3\_block2\_1\_bn (BatchNormal (None, 28, 28, 128) 512 ['conv3\_block2\_1\_conv[0][0]']

ization)

conv3\_block2\_1\_relu (Activatio (None, 28, 28, 128) 0 ['conv3\_block2\_1\_bn[0][0]']

n)

conv3\_block2\_2\_conv (Conv2D) (None, 28, 28, 128) 147584 ['conv3\_block2\_1\_relu[0][0]']

conv3\_block2\_2\_bn (BatchNormal (None, 28, 28, 128) 512 ['conv3\_block2\_2\_conv[0][0]']

ization)

conv3\_block2\_2\_relu (Activatio (None, 28, 28, 128) 0 ['conv3\_block2\_2\_bn[0][0]']

n)

conv3\_block2\_3\_conv (Conv2D) (None, 28, 28, 512) 66048 ['conv3\_block2\_2\_relu[0][0]']

conv3\_block2\_3\_bn (BatchNormal (None, 28, 28, 512) 2048 ['conv3\_block2\_3\_conv[0][0]']

ization)

conv3\_block2\_add (Add) (None, 28, 28, 512) 0 ['conv3\_block1\_out[0][0]',

'conv3\_block2\_3\_bn[0][0]']

conv3\_block2\_out (Activation) (None, 28, 28, 512) 0 ['conv3\_block2\_add[0][0]']

conv3\_block3\_1\_conv (Conv2D) (None, 28, 28, 128) 65664 ['conv3\_block2\_out[0][0]']

conv3\_block3\_1\_bn (BatchNormal (None, 28, 28, 128) 512 ['conv3\_block3\_1\_conv[0][0]']

ization)

conv3\_block3\_1\_relu (Activatio (None, 28, 28, 128) 0 ['conv3\_block3\_1\_bn[0][0]']

n)

conv3\_block3\_2\_conv (Conv2D) (None, 28, 28, 128) 147584 ['conv3\_block3\_1\_relu[0][0]']

conv3\_block3\_2\_bn (BatchNormal (None, 28, 28, 128) 512 ['conv3\_block3\_2\_conv[0][0]']

ization)

conv3\_block3\_2\_relu (Activatio (None, 28, 28, 128) 0 ['conv3\_block3\_2\_bn[0][0]']

n)

conv3\_block3\_3\_conv (Conv2D) (None, 28, 28, 512) 66048 ['conv3\_block3\_2\_relu[0][0]']

conv3\_block3\_3\_bn (BatchNormal (None, 28, 28, 512) 2048 ['conv3\_block3\_3\_conv[0][0]']

ization)

conv3\_block3\_add (Add) (None, 28, 28, 512) 0 ['conv3\_block2\_out[0][0]',

'conv3\_block3\_3\_bn[0][0]']

conv3\_block3\_out (Activation) (None, 28, 28, 512) 0 ['conv3\_block3\_add[0][0]']

conv3\_block4\_1\_conv (Conv2D) (None, 28, 28, 128) 65664 ['conv3\_block3\_out[0][0]']

conv3\_block4\_1\_bn (BatchNormal (None, 28, 28, 128) 512 ['conv3\_block4\_1\_conv[0][0]']

ization)

conv3\_block4\_1\_relu (Activatio (None, 28, 28, 128) 0 ['conv3\_block4\_1\_bn[0][0]']

n)

conv3\_block4\_2\_conv (Conv2D) (None, 28, 28, 128) 147584 ['conv3\_block4\_1\_relu[0][0]']

conv3\_block4\_2\_bn (BatchNormal (None, 28, 28, 128) 512 ['conv3\_block4\_2\_conv[0][0]']

ization)

conv3\_block4\_2\_relu (Activatio (None, 28, 28, 128) 0 ['conv3\_block4\_2\_bn[0][0]']

n)

conv3\_block4\_3\_conv (Conv2D) (None, 28, 28, 512) 66048 ['conv3\_block4\_2\_relu[0][0]']

conv3\_block4\_3\_bn (BatchNormal (None, 28, 28, 512) 2048 ['conv3\_block4\_3\_conv[0][0]']

ization)

conv3\_block4\_add (Add) (None, 28, 28, 512) 0 ['conv3\_block3\_out[0][0]',

'conv3\_block4\_3\_bn[0][0]']

conv3\_block4\_out (Activation) (None, 28, 28, 512) 0 ['conv3\_block4\_add[0][0]']

conv4\_block1\_1\_conv (Conv2D) (None, 14, 14, 256) 131328 ['conv3\_block4\_out[0][0]']

conv4\_block1\_1\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block1\_1\_conv[0][0]']

ization)

conv4\_block1\_1\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block1\_1\_bn[0][0]']

n)

conv4\_block1\_2\_conv (Conv2D) (None, 14, 14, 256) 590080 ['conv4\_block1\_1\_relu[0][0]']

conv4\_block1\_2\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block1\_2\_conv[0][0]']

ization)

conv4\_block1\_2\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block1\_2\_bn[0][0]']

n)

conv4\_block1\_0\_conv (Conv2D) (None, 14, 14, 1024 525312 ['conv3\_block4\_out[0][0]']

)

conv4\_block1\_3\_conv (Conv2D) (None, 14, 14, 1024 263168 ['conv4\_block1\_2\_relu[0][0]']

)

conv4\_block1\_0\_bn (BatchNormal (None, 14, 14, 1024 4096 ['conv4\_block1\_0\_conv[0][0]']

ization) )

conv4\_block1\_3\_bn (BatchNormal (None, 14, 14, 1024 4096 ['conv4\_block1\_3\_conv[0][0]']

ization) )

conv4\_block1\_add (Add) (None, 14, 14, 1024 0 ['conv4\_block1\_0\_bn[0][0]',

) 'conv4\_block1\_3\_bn[0][0]']

conv4\_block1\_out (Activation) (None, 14, 14, 1024 0 ['conv4\_block1\_add[0][0]']

)

conv4\_block2\_1\_conv (Conv2D) (None, 14, 14, 256) 262400 ['conv4\_block1\_out[0][0]']

conv4\_block2\_1\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block2\_1\_conv[0][0]']

ization)

conv4\_block2\_1\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block2\_1\_bn[0][0]']

n)

conv4\_block2\_2\_conv (Conv2D) (None, 14, 14, 256) 590080 ['conv4\_block2\_1\_relu[0][0]']

conv4\_block2\_2\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block2\_2\_conv[0][0]']

ization)

conv4\_block2\_2\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block2\_2\_bn[0][0]']

n)

conv4\_block2\_3\_conv (Conv2D) (None, 14, 14, 1024 263168 ['conv4\_block2\_2\_relu[0][0]']

)

conv4\_block2\_3\_bn (BatchNormal (None, 14, 14, 1024 4096 ['conv4\_block2\_3\_conv[0][0]']

ization) )

conv4\_block2\_add (Add) (None, 14, 14, 1024 0 ['conv4\_block1\_out[0][0]',

) 'conv4\_block2\_3\_bn[0][0]']

conv4\_block2\_out (Activation) (None, 14, 14, 1024 0 ['conv4\_block2\_add[0][0]']

)

conv4\_block3\_1\_conv (Conv2D) (None, 14, 14, 256) 262400 ['conv4\_block2\_out[0][0]']

conv4\_block3\_1\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block3\_1\_conv[0][0]']

ization)

conv4\_block3\_1\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block3\_1\_bn[0][0]']

n)

conv4\_block3\_2\_conv (Conv2D) (None, 14, 14, 256) 590080 ['conv4\_block3\_1\_relu[0][0]']

conv4\_block3\_2\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block3\_2\_conv[0][0]']

ization)

conv4\_block3\_2\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block3\_2\_bn[0][0]']

n)

conv4\_block3\_3\_conv (Conv2D) (None, 14, 14, 1024 263168 ['conv4\_block3\_2\_relu[0][0]']

)

conv4\_block3\_3\_bn (BatchNormal (None, 14, 14, 1024 4096 ['conv4\_block3\_3\_conv[0][0]']

ization) )

conv4\_block3\_add (Add) (None, 14, 14, 1024 0 ['conv4\_block2\_out[0][0]',

) 'conv4\_block3\_3\_bn[0][0]']

conv4\_block3\_out (Activation) (None, 14, 14, 1024 0 ['conv4\_block3\_add[0][0]']

)

conv4\_block4\_1\_conv (Conv2D) (None, 14, 14, 256) 262400 ['conv4\_block3\_out[0][0]']

conv4\_block4\_1\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block4\_1\_conv[0][0]']

ization)

conv4\_block4\_1\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block4\_1\_bn[0][0]']

n)

conv4\_block4\_2\_conv (Conv2D) (None, 14, 14, 256) 590080 ['conv4\_block4\_1\_relu[0][0]']

conv4\_block4\_2\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block4\_2\_conv[0][0]']

ization)

conv4\_block4\_2\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block4\_2\_bn[0][0]']

n)

conv4\_block4\_3\_conv (Conv2D) (None, 14, 14, 1024 263168 ['conv4\_block4\_2\_relu[0][0]']

)

conv4\_block4\_3\_bn (BatchNormal (None, 14, 14, 1024 4096 ['conv4\_block4\_3\_conv[0][0]']

ization) )

conv4\_block4\_add (Add) (None, 14, 14, 1024 0 ['conv4\_block3\_out[0][0]',

) 'conv4\_block4\_3\_bn[0][0]']

conv4\_block4\_out (Activation) (None, 14, 14, 1024 0 ['conv4\_block4\_add[0][0]']

)

conv4\_block5\_1\_conv (Conv2D) (None, 14, 14, 256) 262400 ['conv4\_block4\_out[0][0]']

conv4\_block5\_1\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block5\_1\_conv[0][0]']

ization)

conv4\_block5\_1\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block5\_1\_bn[0][0]']

n)

conv4\_block5\_2\_conv (Conv2D) (None, 14, 14, 256) 590080 ['conv4\_block5\_1\_relu[0][0]']

conv4\_block5\_2\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block5\_2\_conv[0][0]']

ization)

conv4\_block5\_2\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block5\_2\_bn[0][0]']

n)

conv4\_block5\_3\_conv (Conv2D) (None, 14, 14, 1024 263168 ['conv4\_block5\_2\_relu[0][0]']

)

conv4\_block5\_3\_bn (BatchNormal (None, 14, 14, 1024 4096 ['conv4\_block5\_3\_conv[0][0]']

ization) )

conv4\_block5\_add (Add) (None, 14, 14, 1024 0 ['conv4\_block4\_out[0][0]',

) 'conv4\_block5\_3\_bn[0][0]']

conv4\_block5\_out (Activation) (None, 14, 14, 1024 0 ['conv4\_block5\_add[0][0]']

)

conv4\_block6\_1\_conv (Conv2D) (None, 14, 14, 256) 262400 ['conv4\_block5\_out[0][0]']

conv4\_block6\_1\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block6\_1\_conv[0][0]']

ization)

conv4\_block6\_1\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block6\_1\_bn[0][0]']

n)

conv4\_block6\_2\_conv (Conv2D) (None, 14, 14, 256) 590080 ['conv4\_block6\_1\_relu[0][0]']

conv4\_block6\_2\_bn (BatchNormal (None, 14, 14, 256) 1024 ['conv4\_block6\_2\_conv[0][0]']

ization)

conv4\_block6\_2\_relu (Activatio (None, 14, 14, 256) 0 ['conv4\_block6\_2\_bn[0][0]']

n)

conv4\_block6\_3\_conv (Conv2D) (None, 14, 14, 1024 263168 ['conv4\_block6\_2\_relu[0][0]']

)

conv4\_block6\_3\_bn (BatchNormal (None, 14, 14, 1024 4096 ['conv4\_block6\_3\_conv[0][0]']

ization) )

conv4\_block6\_add (Add) (None, 14, 14, 1024 0 ['conv4\_block5\_out[0][0]',

) 'conv4\_block6\_3\_bn[0][0]']

conv4\_block6\_out (Activation) (None, 14, 14, 1024 0 ['conv4\_block6\_add[0][0]']

)

conv5\_block1\_1\_conv (Conv2D) (None, 7, 7, 512) 524800 ['conv4\_block6\_out[0][0]']

conv5\_block1\_1\_bn (BatchNormal (None, 7, 7, 512) 2048 ['conv5\_block1\_1\_conv[0][0]']

ization)

conv5\_block1\_1\_relu (Activatio (None, 7, 7, 512) 0 ['conv5\_block1\_1\_bn[0][0]']

n)

conv5\_block1\_2\_conv (Conv2D) (None, 7, 7, 512) 2359808 ['conv5\_block1\_1\_relu[0][0]']

conv5\_block1\_2\_bn (BatchNormal (None, 7, 7, 512) 2048 ['conv5\_block1\_2\_conv[0][0]']

ization)

conv5\_block1\_2\_relu (Activatio (None, 7, 7, 512) 0 ['conv5\_block1\_2\_bn[0][0]']

n)

conv5\_block1\_0\_conv (Conv2D) (None, 7, 7, 2048) 2099200 ['conv4\_block6\_out[0][0]']

conv5\_block1\_3\_conv (Conv2D) (None, 7, 7, 2048) 1050624 ['conv5\_block1\_2\_relu[0][0]']

conv5\_block1\_0\_bn (BatchNormal (None, 7, 7, 2048) 8192 ['conv5\_block1\_0\_conv[0][0]']

ization)

conv5\_block1\_3\_bn (BatchNormal (None, 7, 7, 2048) 8192 ['conv5\_block1\_3\_conv[0][0]']

ization)

conv5\_block1\_add (Add) (None, 7, 7, 2048) 0 ['conv5\_block1\_0\_bn[0][0]',

'conv5\_block1\_3\_bn[0][0]']

conv5\_block1\_out (Activation) (None, 7, 7, 2048) 0 ['conv5\_block1\_add[0][0]']

conv5\_block2\_1\_conv (Conv2D) (None, 7, 7, 512) 1049088 ['conv5\_block1\_out[0][0]']

conv5\_block2\_1\_bn (BatchNormal (None, 7, 7, 512) 2048 ['conv5\_block2\_1\_conv[0][0]']

ization)

conv5\_block2\_1\_relu (Activatio (None, 7, 7, 512) 0 ['conv5\_block2\_1\_bn[0][0]']

n)

conv5\_block2\_2\_conv (Conv2D) (None, 7, 7, 512) 2359808 ['conv5\_block2\_1\_relu[0][0]']

conv5\_block2\_2\_bn (BatchNormal (None, 7, 7, 512) 2048 ['conv5\_block2\_2\_conv[0][0]']

ization)

conv5\_block2\_2\_relu (Activatio (None, 7, 7, 512) 0 ['conv5\_block2\_2\_bn[0][0]']

n)

conv5\_block2\_3\_conv (Conv2D) (None, 7, 7, 2048) 1050624 ['conv5\_block2\_2\_relu[0][0]']

conv5\_block2\_3\_bn (BatchNormal (None, 7, 7, 2048) 8192 ['conv5\_block2\_3\_conv[0][0]']

ization)

conv5\_block2\_add (Add) (None, 7, 7, 2048) 0 ['conv5\_block1\_out[0][0]',

'conv5\_block2\_3\_bn[0][0]']

conv5\_block2\_out (Activation) (None, 7, 7, 2048) 0 ['conv5\_block2\_add[0][0]']

conv5\_block3\_1\_conv (Conv2D) (None, 7, 7, 512) 1049088 ['conv5\_block2\_out[0][0]']

conv5\_block3\_1\_bn (BatchNormal (None, 7, 7, 512) 2048 ['conv5\_block3\_1\_conv[0][0]']

ization)

conv5\_block3\_1\_relu (Activatio (None, 7, 7, 512) 0 ['conv5\_block3\_1\_bn[0][0]']

n)

conv5\_block3\_2\_conv (Conv2D) (None, 7, 7, 512) 2359808 ['conv5\_block3\_1\_relu[0][0]']

conv5\_block3\_2\_bn (BatchNormal (None, 7, 7, 512) 2048 ['conv5\_block3\_2\_conv[0][0]']

ization)

conv5\_block3\_2\_relu (Activatio (None, 7, 7, 512) 0 ['conv5\_block3\_2\_bn[0][0]']

n)

conv5\_block3\_3\_conv (Conv2D) (None, 7, 7, 2048) 1050624 ['conv5\_block3\_2\_relu[0][0]']

conv5\_block3\_3\_bn (BatchNormal (None, 7, 7, 2048) 8192 ['conv5\_block3\_3\_conv[0][0]']

ization)

conv5\_block3\_add (Add) (None, 7, 7, 2048) 0 ['conv5\_block2\_out[0][0]',

'conv5\_block3\_3\_bn[0][0]']

conv5\_block3\_out (Activation) (None, 7, 7, 2048) 0 ['conv5\_block3\_add[0][0]']

avg\_pool (GlobalAveragePooling (None, 2048) 0 ['conv5\_block3\_out[0][0]']

2D)

dense\_3 (Dense) (None, 2048) 4196352 ['avg\_pool[0][0]']

dense\_4 (Dense) (None, 512) 1049088 ['dense\_3[0][0]']

dropout\_1 (Dropout) (None, 512) 0 ['dense\_4[0][0]']

dense\_5 (Dense) (None, 128) 65664 ['dropout\_1[0][0]']

dense\_6 (Dense) (None, 64) 8256 ['dense\_5[0][0]']

dropout\_2 (Dropout) (None, 64) 0 ['dense\_6[0][0]']

dense\_7 (Dense) (None, 32) 2080 ['dropout\_2[0][0]']

dense\_8 (Dense) (None, 16) 528 ['dense\_7[0][0]']

dense\_9 (Dense) (None, 5) 85 ['dense\_8[0][0]']

==================================================================================================

Total params: 28,909,765

Trainable params: 5,322,053

Non-trainable params: 23,587,712

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In [ ]:

model2.compile(

optimizer=tf.keras.optimizers.Adam(lr=0.0001),

loss='categorical\_crossentropy',

metrics=['accuracy'])

/usr/local/lib/python3.7/dist-packages/keras/optimizer\_v2/adam.py:105: UserWarning: The `lr` argument is deprecated, use `learning\_rate` instead.

super(Adam, self).\_\_init\_\_(name, \*\*kwargs)

In [ ]:

reduce\_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor = 'val\_loss', mode='min',patience=3,

verbose=1, factor=0.5, min\_lr=0.000001)

In [ ]:

EarlyStopping=tf.keras.callbacks.EarlyStopping(

monitor='val\_accuracy',

min\_delta=0,

patience=3,

verbose=0,

mode='auto',

baseline=None,

restore\_best\_weights=True

)

In [ ]:

history = model2.fit\_generator(

generator=train\_generator,

epochs=25,

validation\_data=validation\_generator,

callbacks=[reduce\_lr, EarlyStopping])

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:5: UserWarning: `Model.fit\_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.

"""

Epoch 1/25

60/60 [==============================] - 67s 1s/step - loss: 0.6948 - accuracy: 0.7437 - val\_loss: 0.2462 - val\_accuracy: 0.9135 - lr: 1.0000e-04

Epoch 2/25

60/60 [==============================] - 57s 946ms/step - loss: 0.2334 - accuracy: 0.9234 - val\_loss: 0.1177 - val\_accuracy: 0.9573 - lr: 1.0000e-04

Epoch 3/25

60/60 [==============================] - 57s 952ms/step - loss: 0.1812 - accuracy: 0.9401 - val\_loss: 0.0857 - val\_accuracy: 0.9646 - lr: 1.0000e-04

Epoch 4/25

60/60 [==============================] - 55s 911ms/step - loss: 0.1349 - accuracy: 0.9523 - val\_loss: 0.1120 - val\_accuracy: 0.9615 - lr: 1.0000e-04

Epoch 5/25

60/60 [==============================] - 54s 905ms/step - loss: 0.1235 - accuracy: 0.9583 - val\_loss: 0.0525 - val\_accuracy: 0.9771 - lr: 1.0000e-04

Epoch 6/25

60/60 [==============================] - 54s 904ms/step - loss: 0.0959 - accuracy: 0.9661 - val\_loss: 0.0533 - val\_accuracy: 0.9802 - lr: 1.0000e-04

Epoch 7/25

60/60 [==============================] - 56s 936ms/step - loss: 0.0884 - accuracy: 0.9690 - val\_loss: 0.0519 - val\_accuracy: 0.9781 - lr: 1.0000e-04

Epoch 8/25

60/60 [==============================] - 55s 915ms/step - loss: 0.0924 - accuracy: 0.9695 - val\_loss: 0.0505 - val\_accuracy: 0.9823 - lr: 1.0000e-04

Epoch 9/25

60/60 [==============================] - 54s 907ms/step - loss: 0.0789 - accuracy: 0.9750 - val\_loss: 0.0637 - val\_accuracy: 0.9812 - lr: 1.0000e-04

Epoch 10/25

60/60 [==============================] - 55s 910ms/step - loss: 0.0755 - accuracy: 0.9721 - val\_loss: 0.0716 - val\_accuracy: 0.9750 - lr: 1.0000e-04

Epoch 11/25

60/60 [==============================] - 55s 912ms/step - loss: 0.0664 - accuracy: 0.9771 - val\_loss: 0.0320 - val\_accuracy: 0.9875 - lr: 1.0000e-04

Epoch 12/25

60/60 [==============================] - 56s 931ms/step - loss: 0.0824 - accuracy: 0.9721 - val\_loss: 0.0400 - val\_accuracy: 0.9812 - lr: 1.0000e-04

Epoch 13/25

60/60 [==============================] - 54s 906ms/step - loss: 0.0731 - accuracy: 0.9740 - val\_loss: 0.0574 - val\_accuracy: 0.9812 - lr: 1.0000e-04

Epoch 14/25

60/60 [==============================] - ETA: 0s - loss: 0.0732 - accuracy: 0.9755

Epoch 14: ReduceLROnPlateau reducing learning rate to 4.999999873689376e-05.

60/60 [==============================] - 54s 906ms/step - loss: 0.0732 - accuracy: 0.9755 - val\_loss: 0.0419 - val\_accuracy: 0.9844 - lr: 1.0000e-04

In [ ]:

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs = range(1, len(acc) + 1)

sns.set\_style("whitegrid")

plt.title('Training and validation accuracy')

plt.plot(epochs, acc, 'red', label='Training acc')

plt.plot(epochs, val\_acc, 'blue', label='Validation acc')

plt.legend()

plt.figure()

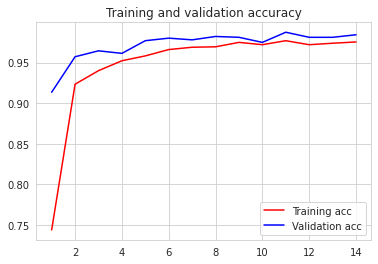
plt.title('Training and validation loss')

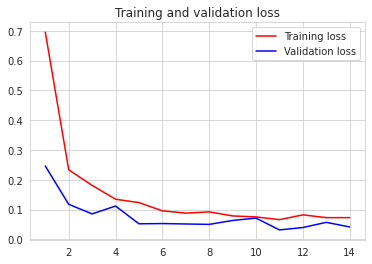
plt.plot(epochs, loss, 'red', label='Training loss')

plt.plot(epochs, val\_loss, 'blue', label='Validation loss')

plt.legend()

plt.show()





In [ ]:

Y\_pred = model2.predict(validation\_generator)

y\_pred = np.argmax(Y\_pred, axis=1)

report1 = classification\_report(validation\_generator.classes, y\_pred, target\_names=rice\_classes, output\_dict=True)

df1 = pd.DataFrame(report1).transpose()

df1

Out[ ]:

|  | **precision** | **recall** | **f1-score** | **support** |
| --- | --- | --- | --- | --- |
| **Arborio** | 0.984615 | 1.000000 | 0.992248 | 192.0000 |
| **Basmati** | 0.994624 | 0.963542 | 0.978836 | 192.0000 |
| **Ipsala** | 0.994819 | 1.000000 | 0.997403 | 192.0000 |
| **Jasmine** | 0.964286 | 0.984375 | 0.974227 | 192.0000 |
| **Karacadag** | 1.000000 | 0.989583 | 0.994764 | 192.0000 |
| **accuracy** | 0.987500 | 0.987500 | 0.987500 | 0.9875 |
| **macro avg** | 0.987669 | 0.987500 | 0.987496 | 960.0000 |
| **weighted avg** | 0.987669 | 0.987500 | 0.987496 | 960.0000 |

the result is very satisfactory, the val acc achieved is 98% and the network never overfitted

### **2.3) MobileNet**

To see how the data behave on another pre-trained network, a MobileNet it's used to fine tune the model; all others parameters and architecture remain the same

In [ ]:

size=224

channels=3

batch\_size = 64

num\_classes = 5

train\_processing = ImageDataGenerator(preprocessing\_function=preprocess\_inputMN,

validation\_split=0.2,

rotation\_range = 25,

width\_shift\_range = .2,

height\_shift\_range = .2,

horizontal\_flip = True,

zoom\_range = .2)

validation\_datagen = ImageDataGenerator(preprocessing\_function=preprocess\_inputMN,

validation\_split=0.2)

train\_generator = train\_processing.flow\_from\_directory (base\_path,

target\_size=(size, size),

batch\_size=batch\_size,

shuffle=True,

class\_mode='categorical',

subset='training',

color\_mode="rgb",

interpolation='nearest')

validation\_generator = validation\_datagen.flow\_from\_directory(base\_path,

target\_size=(size,size),

subset='validation',

color\_mode="rgb",

batch\_size=batch\_size,

class\_mode='categorical',

shuffle=False,

interpolation='nearest')

Found 3840 images belonging to 5 classes.

Found 960 images belonging to 5 classes.

In [ ]:

pretrained\_model = MobileNet(

weights='imagenet',

input\_shape=(size, size, 3),

pooling='avg'

)

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet/mobilenet\_1\_0\_224\_tf.h5

17227776/17225924 [==============================] - 0s 0us/step

17235968/17225924 [==============================] - 0s 0us/step

In [ ]:

print('Numero layers MobileNet: {}'.format(len(pretrained\_model.layers)))

Numero layers MobileNet: 91

In [ ]:

layer\_dict = dict([(layer.name, layer) for layer in pretrained\_model.layers])

for layer in pretrained\_model.layers:

layer.trainable = False

In [ ]:

x3 = pretrained\_model.output

x3 = Dense(2048,'relu')(x3)

x3 = Dense(512,'relu')(x3)

x3 = Dropout(.15)(x3)

x3 = Dense(128,'relu')(x3)

x3 = Dense(64,'relu')(x3)

x3 = Dropout(.15)(x3)

x3 = Dense(32,'relu')(x3)

x3 = Dense(16,'relu')(x3)

pred3 = Dense(5,'softmax')(x3)

model3 = Model(inputs=pretrained\_model.input, outputs=pred3)

In [ ]:

model3.compile(

optimizer=tf.keras.optimizers.Adam(lr=0.0001),

loss='categorical\_crossentropy',

metrics=['accuracy'])

/usr/local/lib/python3.7/dist-packages/keras/optimizer\_v2/adam.py:105: UserWarning: The `lr` argument is deprecated, use `learning\_rate` instead.

super(Adam, self).\_\_init\_\_(name, \*\*kwargs)

In [ ]:

model3.summary()

Model: "model\_2"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

input\_3 (InputLayer) [(None, 224, 224, 3)] 0

conv1 (Conv2D) (None, 112, 112, 32) 864

conv1\_bn (BatchNormalizatio (None, 112, 112, 32) 128

n)

conv1\_relu (ReLU) (None, 112, 112, 32) 0

conv\_dw\_1 (DepthwiseConv2D) (None, 112, 112, 32) 288

conv\_dw\_1\_bn (BatchNormaliz (None, 112, 112, 32) 128

ation)

conv\_dw\_1\_relu (ReLU) (None, 112, 112, 32) 0

conv\_pw\_1 (Conv2D) (None, 112, 112, 64) 2048

conv\_pw\_1\_bn (BatchNormaliz (None, 112, 112, 64) 256

ation)

conv\_pw\_1\_relu (ReLU) (None, 112, 112, 64) 0

conv\_pad\_2 (ZeroPadding2D) (None, 113, 113, 64) 0

conv\_dw\_2 (DepthwiseConv2D) (None, 56, 56, 64) 576

conv\_dw\_2\_bn (BatchNormaliz (None, 56, 56, 64) 256

ation)

conv\_dw\_2\_relu (ReLU) (None, 56, 56, 64) 0

conv\_pw\_2 (Conv2D) (None, 56, 56, 128) 8192

conv\_pw\_2\_bn (BatchNormaliz (None, 56, 56, 128) 512

ation)

conv\_pw\_2\_relu (ReLU) (None, 56, 56, 128) 0

conv\_dw\_3 (DepthwiseConv2D) (None, 56, 56, 128) 1152

conv\_dw\_3\_bn (BatchNormaliz (None, 56, 56, 128) 512

ation)

conv\_dw\_3\_relu (ReLU) (None, 56, 56, 128) 0

conv\_pw\_3 (Conv2D) (None, 56, 56, 128) 16384

conv\_pw\_3\_bn (BatchNormaliz (None, 56, 56, 128) 512

ation)

conv\_pw\_3\_relu (ReLU) (None, 56, 56, 128) 0

conv\_pad\_4 (ZeroPadding2D) (None, 57, 57, 128) 0

conv\_dw\_4 (DepthwiseConv2D) (None, 28, 28, 128) 1152

conv\_dw\_4\_bn (BatchNormaliz (None, 28, 28, 128) 512

ation)

conv\_dw\_4\_relu (ReLU) (None, 28, 28, 128) 0

conv\_pw\_4 (Conv2D) (None, 28, 28, 256) 32768

conv\_pw\_4\_bn (BatchNormaliz (None, 28, 28, 256) 1024

ation)

conv\_pw\_4\_relu (ReLU) (None, 28, 28, 256) 0

conv\_dw\_5 (DepthwiseConv2D) (None, 28, 28, 256) 2304

conv\_dw\_5\_bn (BatchNormaliz (None, 28, 28, 256) 1024

ation)

conv\_dw\_5\_relu (ReLU) (None, 28, 28, 256) 0

conv\_pw\_5 (Conv2D) (None, 28, 28, 256) 65536

conv\_pw\_5\_bn (BatchNormaliz (None, 28, 28, 256) 1024

ation)

conv\_pw\_5\_relu (ReLU) (None, 28, 28, 256) 0

conv\_pad\_6 (ZeroPadding2D) (None, 29, 29, 256) 0

conv\_dw\_6 (DepthwiseConv2D) (None, 14, 14, 256) 2304

conv\_dw\_6\_bn (BatchNormaliz (None, 14, 14, 256) 1024

ation)

conv\_dw\_6\_relu (ReLU) (None, 14, 14, 256) 0

conv\_pw\_6 (Conv2D) (None, 14, 14, 512) 131072

conv\_pw\_6\_bn (BatchNormaliz (None, 14, 14, 512) 2048

ation)

conv\_pw\_6\_relu (ReLU) (None, 14, 14, 512) 0

conv\_dw\_7 (DepthwiseConv2D) (None, 14, 14, 512) 4608

conv\_dw\_7\_bn (BatchNormaliz (None, 14, 14, 512) 2048

ation)

conv\_dw\_7\_relu (ReLU) (None, 14, 14, 512) 0

conv\_pw\_7 (Conv2D) (None, 14, 14, 512) 262144

conv\_pw\_7\_bn (BatchNormaliz (None, 14, 14, 512) 2048

ation)

conv\_pw\_7\_relu (ReLU) (None, 14, 14, 512) 0

conv\_dw\_8 (DepthwiseConv2D) (None, 14, 14, 512) 4608

conv\_dw\_8\_bn (BatchNormaliz (None, 14, 14, 512) 2048

ation)

conv\_dw\_8\_relu (ReLU) (None, 14, 14, 512) 0

conv\_pw\_8 (Conv2D) (None, 14, 14, 512) 262144

conv\_pw\_8\_bn (BatchNormaliz (None, 14, 14, 512) 2048

ation)

conv\_pw\_8\_relu (ReLU) (None, 14, 14, 512) 0

conv\_dw\_9 (DepthwiseConv2D) (None, 14, 14, 512) 4608

conv\_dw\_9\_bn (BatchNormaliz (None, 14, 14, 512) 2048

ation)

conv\_dw\_9\_relu (ReLU) (None, 14, 14, 512) 0

conv\_pw\_9 (Conv2D) (None, 14, 14, 512) 262144

conv\_pw\_9\_bn (BatchNormaliz (None, 14, 14, 512) 2048

ation)

conv\_pw\_9\_relu (ReLU) (None, 14, 14, 512) 0

conv\_dw\_10 (DepthwiseConv2D (None, 14, 14, 512) 4608

)

conv\_dw\_10\_bn (BatchNormali (None, 14, 14, 512) 2048

zation)

conv\_dw\_10\_relu (ReLU) (None, 14, 14, 512) 0

conv\_pw\_10 (Conv2D) (None, 14, 14, 512) 262144

conv\_pw\_10\_bn (BatchNormali (None, 14, 14, 512) 2048

zation)

conv\_pw\_10\_relu (ReLU) (None, 14, 14, 512) 0

conv\_dw\_11 (DepthwiseConv2D (None, 14, 14, 512) 4608

)

conv\_dw\_11\_bn (BatchNormali (None, 14, 14, 512) 2048

zation)

conv\_dw\_11\_relu (ReLU) (None, 14, 14, 512) 0

conv\_pw\_11 (Conv2D) (None, 14, 14, 512) 262144

conv\_pw\_11\_bn (BatchNormali (None, 14, 14, 512) 2048

zation)

conv\_pw\_11\_relu (ReLU) (None, 14, 14, 512) 0

conv\_pad\_12 (ZeroPadding2D) (None, 15, 15, 512) 0

conv\_dw\_12 (DepthwiseConv2D (None, 7, 7, 512) 4608

)

conv\_dw\_12\_bn (BatchNormali (None, 7, 7, 512) 2048

zation)

conv\_dw\_12\_relu (ReLU) (None, 7, 7, 512) 0

conv\_pw\_12 (Conv2D) (None, 7, 7, 1024) 524288

conv\_pw\_12\_bn (BatchNormali (None, 7, 7, 1024) 4096

zation)

conv\_pw\_12\_relu (ReLU) (None, 7, 7, 1024) 0

conv\_dw\_13 (DepthwiseConv2D (None, 7, 7, 1024) 9216

)

conv\_dw\_13\_bn (BatchNormali (None, 7, 7, 1024) 4096

zation)

conv\_dw\_13\_relu (ReLU) (None, 7, 7, 1024) 0

conv\_pw\_13 (Conv2D) (None, 7, 7, 1024) 1048576

conv\_pw\_13\_bn (BatchNormali (None, 7, 7, 1024) 4096

zation)

conv\_pw\_13\_relu (ReLU) (None, 7, 7, 1024) 0

global\_average\_pooling2d (G (None, 1, 1, 1024) 0

lobalAveragePooling2D)

dropout (Dropout) (None, 1, 1, 1024) 0

conv\_preds (Conv2D) (None, 1, 1, 1000) 1025000

reshape\_2 (Reshape) (None, 1000) 0

predictions (Activation) (None, 1000) 0

dense\_10 (Dense) (None, 2048) 2050048

dense\_11 (Dense) (None, 512) 1049088

dropout\_3 (Dropout) (None, 512) 0

dense\_12 (Dense) (None, 128) 65664

dense\_13 (Dense) (None, 64) 8256

dropout\_4 (Dropout) (None, 64) 0

dense\_14 (Dense) (None, 32) 2080

dense\_15 (Dense) (None, 16) 528

dense\_16 (Dense) (None, 5) 85

=================================================================

Total params: 7,429,613

Trainable params: 3,175,749

Non-trainable params: 4,253,864

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In [ ]:

reduce\_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor = 'val\_loss', mode='min',patience=3,

verbose=1, factor=0.5, min\_lr=0.000001)

In [ ]:

EarlyStopping=tf.keras.callbacks.EarlyStopping(

monitor='val\_accuracy',

min\_delta=0,

patience=3,

verbose=0,

mode='auto',

baseline=None,

restore\_best\_weights=True

)

In [ ]:

history = model3.fit\_generator(

generator=train\_generator,

epochs=25,

validation\_data=validation\_generator,

callbacks=[reduce\_lr, EarlyStopping])

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:5: UserWarning: `Model.fit\_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.

"""

Epoch 1/25

60/60 [==============================] - 63s 1s/step - loss: 1.5497 - accuracy: 0.4026 - val\_loss: 1.4257 - val\_accuracy: 0.4615 - lr: 1.0000e-04

Epoch 2/25

60/60 [==============================] - 53s 879ms/step - loss: 1.3393 - accuracy: 0.5005 - val\_loss: 1.1824 - val\_accuracy: 0.5292 - lr: 1.0000e-04

Epoch 3/25

60/60 [==============================] - 53s 888ms/step - loss: 1.1176 - accuracy: 0.5911 - val\_loss: 0.9553 - val\_accuracy: 0.6542 - lr: 1.0000e-04

Epoch 4/25

60/60 [==============================] - 52s 868ms/step - loss: 0.9112 - accuracy: 0.6695 - val\_loss: 0.8074 - val\_accuracy: 0.7115 - lr: 1.0000e-04

Epoch 5/25

60/60 [==============================] - 53s 883ms/step - loss: 0.7821 - accuracy: 0.7198 - val\_loss: 0.7451 - val\_accuracy: 0.7448 - lr: 1.0000e-04

Epoch 6/25

60/60 [==============================] - 54s 895ms/step - loss: 0.6708 - accuracy: 0.7628 - val\_loss: 0.6923 - val\_accuracy: 0.7146 - lr: 1.0000e-04

Epoch 7/25

60/60 [==============================] - 52s 866ms/step - loss: 0.6156 - accuracy: 0.7732 - val\_loss: 0.6675 - val\_accuracy: 0.7521 - lr: 1.0000e-04

Epoch 8/25

60/60 [==============================] - 52s 870ms/step - loss: 0.5608 - accuracy: 0.8042 - val\_loss: 0.6494 - val\_accuracy: 0.7760 - lr: 1.0000e-04

Epoch 9/25

60/60 [==============================] - 52s 869ms/step - loss: 0.5333 - accuracy: 0.8122 - val\_loss: 0.6146 - val\_accuracy: 0.7469 - lr: 1.0000e-04

Epoch 10/25

60/60 [==============================] - 52s 869ms/step - loss: 0.5025 - accuracy: 0.8234 - val\_loss: 0.5711 - val\_accuracy: 0.7927 - lr: 1.0000e-04

Epoch 11/25

60/60 [==============================] - 52s 874ms/step - loss: 0.4776 - accuracy: 0.8305 - val\_loss: 0.5223 - val\_accuracy: 0.8000 - lr: 1.0000e-04

Epoch 12/25

60/60 [==============================] - 53s 877ms/step - loss: 0.4421 - accuracy: 0.8344 - val\_loss: 0.5235 - val\_accuracy: 0.8083 - lr: 1.0000e-04

Epoch 13/25

60/60 [==============================] - 51s 854ms/step - loss: 0.4209 - accuracy: 0.8565 - val\_loss: 0.4925 - val\_accuracy: 0.7937 - lr: 1.0000e-04

Epoch 14/25

60/60 [==============================] - 51s 852ms/step - loss: 0.4305 - accuracy: 0.8492 - val\_loss: 0.4401 - val\_accuracy: 0.8125 - lr: 1.0000e-04

Epoch 15/25

60/60 [==============================] - 51s 856ms/step - loss: 0.3783 - accuracy: 0.8680 - val\_loss: 0.4992 - val\_accuracy: 0.8125 - lr: 1.0000e-04

Epoch 16/25

60/60 [==============================] - 52s 859ms/step - loss: 0.3842 - accuracy: 0.8628 - val\_loss: 0.4601 - val\_accuracy: 0.8156 - lr: 1.0000e-04

Epoch 17/25

60/60 [==============================] - ETA: 0s - loss: 0.3517 - accuracy: 0.8732

Epoch 17: ReduceLROnPlateau reducing learning rate to 4.999999873689376e-05.

60/60 [==============================] - 52s 859ms/step - loss: 0.3517 - accuracy: 0.8732 - val\_loss: 0.5738 - val\_accuracy: 0.7604 - lr: 1.0000e-04

Epoch 18/25

60/60 [==============================] - 52s 867ms/step - loss: 0.3275 - accuracy: 0.8859 - val\_loss: 0.4461 - val\_accuracy: 0.8156 - lr: 5.0000e-05

Epoch 19/25

60/60 [==============================] - 51s 850ms/step - loss: 0.3259 - accuracy: 0.8875 - val\_loss: 0.4524 - val\_accuracy: 0.8260 - lr: 5.0000e-05

Epoch 20/25

60/60 [==============================] - ETA: 0s - loss: 0.3352 - accuracy: 0.8841

Epoch 20: ReduceLROnPlateau reducing learning rate to 2.499999936844688e-05.

60/60 [==============================] - 51s 851ms/step - loss: 0.3352 - accuracy: 0.8841 - val\_loss: 0.4419 - val\_accuracy: 0.8240 - lr: 5.0000e-05

Epoch 21/25

60/60 [==============================] - 52s 862ms/step - loss: 0.3291 - accuracy: 0.8883 - val\_loss: 0.4201 - val\_accuracy: 0.8510 - lr: 2.5000e-05

Epoch 22/25

60/60 [==============================] - 51s 856ms/step - loss: 0.3078 - accuracy: 0.8953 - val\_loss: 0.4055 - val\_accuracy: 0.8365 - lr: 2.5000e-05

Epoch 23/25

60/60 [==============================] - 53s 882ms/step - loss: 0.3194 - accuracy: 0.8917 - val\_loss: 0.4081 - val\_accuracy: 0.8354 - lr: 2.5000e-05

Epoch 24/25

60/60 [==============================] - 51s 855ms/step - loss: 0.3325 - accuracy: 0.8870 - val\_loss: 0.4201 - val\_accuracy: 0.8562 - lr: 2.5000e-05

Epoch 25/25

60/60 [==============================] - ETA: 0s - loss: 0.3231 - accuracy: 0.8867

Epoch 25: ReduceLROnPlateau reducing learning rate to 1.249999968422344e-05.

60/60 [==============================] - 51s 855ms/step - loss: 0.3231 - accuracy: 0.8867 - val\_loss: 0.4156 - val\_accuracy: 0.8521 - lr: 2.5000e-05

In [ ]:

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs = range(1, len(acc) + 1)

sns.set\_style("whitegrid")

plt.title('Training and validation accuracy')

plt.plot(epochs, acc, 'red', label='Training acc')

plt.plot(epochs, val\_acc, 'blue', label='Validation acc')

plt.legend()

plt.figure()

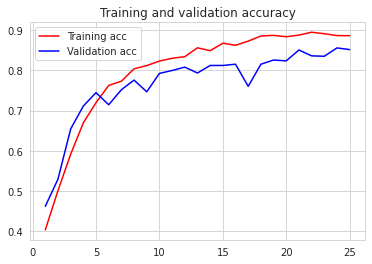
plt.title('Training and validation loss')

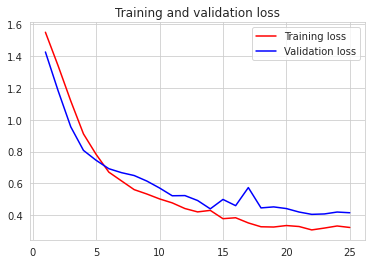
plt.plot(epochs, loss, 'red', label='Training loss')

plt.plot(epochs, val\_loss, 'blue', label='Validation loss')

plt.legend()

plt.show()





In [ ]:

Y\_pred = model3.predict(validation\_generator)

Y\_pred = model3.predict(validation\_generator)

y\_pred = np.argmax(Y\_pred, axis=1)

report1 = classification\_report(validation\_generator.classes, y\_pred, target\_names=rice\_classes, output\_dict=True)

df1 = pd.DataFrame(report1).transpose()

df1

Out[ ]:

|  | **precision** | **recall** | **f1-score** | **support** |
| --- | --- | --- | --- | --- |
| **Arborio** | 0.976378 | 0.645833 | 0.777429 | 192.000000 |
| **Basmati** | 0.941176 | 0.916667 | 0.928760 | 192.000000 |
| **Ipsala** | 0.694444 | 0.911458 | 0.788288 | 192.000000 |
| **Jasmine** | 0.925287 | 0.838542 | 0.879781 | 192.000000 |
| **Karacadag** | 0.827273 | 0.947917 | 0.883495 | 192.000000 |
| **accuracy** | 0.852083 | 0.852083 | 0.852083 | 0.852083 |
| **macro avg** | 0.872912 | 0.852083 | 0.851551 | 960.000000 |
| **weighted avg** | 0.872912 | 0.852083 | 0.851551 | 960.000000 |

The results dosen't show any improvent from the previous network, instead, it overfit and achive a val acc just of 85%.

### **2.4) VGG16**

Another pre-trained Neural Network it's used for fine tuning, VGG16.

In [ ]:

size=224

In [ ]:

size=224

channels=3

batch\_size = 64

num\_classes = 5

train\_processing = ImageDataGenerator(preprocessing\_function=preprocess\_inputVG,

validation\_split=0.2,

rotation\_range = 25,

width\_shift\_range = .2,

height\_shift\_range = .2,

horizontal\_flip = True,

zoom\_range = .2)

validation\_datagen = ImageDataGenerator(preprocessing\_function=preprocess\_inputVG,

validation\_split=0.2)

train\_generator = train\_processing.flow\_from\_directory (base\_path,

target\_size=(size, size),

batch\_size=batch\_size,

shuffle=True,

class\_mode='categorical',

subset='training',

color\_mode="rgb",

interpolation='nearest')

validation\_generator = validation\_datagen.flow\_from\_directory(base\_path,

target\_size=(size,size),

subset='validation',

color\_mode="rgb",

batch\_size=batch\_size,

class\_mode='categorical',

shuffle=False,

interpolation='nearest')

Found 3840 images belonging to 5 classes.

Found 960 images belonging to 5 classes.

In [ ]:

pretrained\_model = VGG16(

weights='imagenet',

input\_shape=(size, size, 3),

pooling='avg'

)

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16\_weights\_tf\_dim\_ordering\_tf\_kernels.h5

553467904/553467096 [==============================] - 10s 0us/step

553476096/553467096 [==============================] - 10s 0us/step

In [ ]:

print('Numero layers MobileNet: {}'.format(len(pretrained\_model.layers)))

Numero layers MobileNet: 23

In [ ]:

layer\_dict = dict([(layer.name, layer) for layer in pretrained\_model.layers])

for layer in pretrained\_model.layers:

layer.trainable = False

In [ ]:

x4 = pretrained\_model.output

x4 = Dense(2048,'relu')(x4)

x4 = Dense(512,'relu')(x4)

x4 = Dropout(.15)(x4)

x4 = Dense(128,'relu')(x4)

x4 = Dense(64,'relu')(x4)

x4 = Dropout(.15)(x4)

x4 = Dense(32,'relu')(x4)

x4 = Dense(16,'relu')(x4)

pred4 = Dense(5,'softmax')(x4)

model4 = Model(inputs=pretrained\_model.input, outputs=pred4)

In [ ]:

model4.compile(

optimizer=tf.keras.optimizers.Adam(lr=0.0001),

loss='categorical\_crossentropy',

metrics=['accuracy'])

/usr/local/lib/python3.7/dist-packages/keras/optimizer\_v2/adam.py:105: UserWarning: The `lr` argument is deprecated, use `learning\_rate` instead.

super(Adam, self).\_\_init\_\_(name, \*\*kwargs)

In [ ]:

model4.summary()

Model: "model\_3"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

input\_4 (InputLayer) [(None, 224, 224, 3)] 0

block1\_conv1 (Conv2D) (None, 224, 224, 64) 1792

block1\_conv2 (Conv2D) (None, 224, 224, 64) 36928

block1\_pool (MaxPooling2D) (None, 112, 112, 64) 0

block2\_conv1 (Conv2D) (None, 112, 112, 128) 73856

block2\_conv2 (Conv2D) (None, 112, 112, 128) 147584

block2\_pool (MaxPooling2D) (None, 56, 56, 128) 0

block3\_conv1 (Conv2D) (None, 56, 56, 256) 295168

block3\_conv2 (Conv2D) (None, 56, 56, 256) 590080

block3\_conv3 (Conv2D) (None, 56, 56, 256) 590080

block3\_pool (MaxPooling2D) (None, 28, 28, 256) 0

block4\_conv1 (Conv2D) (None, 28, 28, 512) 1180160

block4\_conv2 (Conv2D) (None, 28, 28, 512) 2359808

block4\_conv3 (Conv2D) (None, 28, 28, 512) 2359808

block4\_pool (MaxPooling2D) (None, 14, 14, 512) 0

block5\_conv1 (Conv2D) (None, 14, 14, 512) 2359808

block5\_conv2 (Conv2D) (None, 14, 14, 512) 2359808

block5\_conv3 (Conv2D) (None, 14, 14, 512) 2359808

block5\_pool (MaxPooling2D) (None, 7, 7, 512) 0

flatten (Flatten) (None, 25088) 0

fc1 (Dense) (None, 4096) 102764544

fc2 (Dense) (None, 4096) 16781312

predictions (Dense) (None, 1000) 4097000

dense\_17 (Dense) (None, 2048) 2050048

dense\_18 (Dense) (None, 512) 1049088

dropout\_5 (Dropout) (None, 512) 0

dense\_19 (Dense) (None, 128) 65664

dense\_20 (Dense) (None, 64) 8256

dropout\_6 (Dropout) (None, 64) 0

dense\_21 (Dense) (None, 32) 2080

dense\_22 (Dense) (None, 16) 528

dense\_23 (Dense) (None, 5) 85

=================================================================

Total params: 141,533,293

Trainable params: 3,175,749

Non-trainable params: 138,357,544

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In [ ]:

reduce\_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor = 'val\_loss', mode='min',patience=3,

verbose=1, factor=0.5, min\_lr=0.000001)

In [ ]:

EarlyStopping=tf.keras.callbacks.EarlyStopping(

monitor='val\_accuracy',

min\_delta=0,

patience=3,

verbose=0,

mode='auto',

baseline=None,

restore\_best\_weights=True

)

In [ ]:

history = model4.fit\_generator(

generator=train\_generator,

epochs=25,

validation\_data=validation\_generator,

callbacks=[reduce\_lr, EarlyStopping])

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:5: UserWarning: `Model.fit\_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.

"""

Epoch 1/25

60/60 [==============================] - 68s 1s/step - loss: 1.5693 - accuracy: 0.3773 - val\_loss: 1.4546 - val\_accuracy: 0.3875 - lr: 1.0000e-04

Epoch 2/25

60/60 [==============================] - 58s 963ms/step - loss: 1.3085 - accuracy: 0.4086 - val\_loss: 1.0835 - val\_accuracy: 0.5938 - lr: 1.0000e-04

Epoch 3/25

60/60 [==============================] - 59s 978ms/step - loss: 1.0516 - accuracy: 0.5482 - val\_loss: 0.8558 - val\_accuracy: 0.6948 - lr: 1.0000e-04

Epoch 4/25

60/60 [==============================] - 60s 1s/step - loss: 0.9076 - accuracy: 0.6190 - val\_loss: 0.7453 - val\_accuracy: 0.7240 - lr: 1.0000e-04

Epoch 5/25

60/60 [==============================] - 58s 962ms/step - loss: 0.8299 - accuracy: 0.6565 - val\_loss: 0.7095 - val\_accuracy: 0.7000 - lr: 1.0000e-04

Epoch 6/25

60/60 [==============================] - 58s 966ms/step - loss: 0.7720 - accuracy: 0.6740 - val\_loss: 0.6654 - val\_accuracy: 0.7302 - lr: 1.0000e-04

Epoch 7/25

60/60 [==============================] - 58s 970ms/step - loss: 0.7329 - accuracy: 0.7005 - val\_loss: 0.6081 - val\_accuracy: 0.7604 - lr: 1.0000e-04

Epoch 8/25

60/60 [==============================] - 58s 965ms/step - loss: 0.7077 - accuracy: 0.7078 - val\_loss: 0.5955 - val\_accuracy: 0.7802 - lr: 1.0000e-04

Epoch 9/25

60/60 [==============================] - 60s 991ms/step - loss: 0.6574 - accuracy: 0.7419 - val\_loss: 0.5327 - val\_accuracy: 0.7979 - lr: 1.0000e-04

Epoch 10/25

60/60 [==============================] - 57s 955ms/step - loss: 0.6159 - accuracy: 0.7557 - val\_loss: 0.5215 - val\_accuracy: 0.7844 - lr: 1.0000e-04

Epoch 11/25

60/60 [==============================] - 58s 957ms/step - loss: 0.6255 - accuracy: 0.7609 - val\_loss: 0.4852 - val\_accuracy: 0.8250 - lr: 1.0000e-04

Epoch 12/25

60/60 [==============================] - 58s 970ms/step - loss: 0.5734 - accuracy: 0.7773 - val\_loss: 0.4675 - val\_accuracy: 0.8385 - lr: 1.0000e-04

Epoch 13/25

60/60 [==============================] - 59s 975ms/step - loss: 0.5697 - accuracy: 0.7771 - val\_loss: 0.4466 - val\_accuracy: 0.8469 - lr: 1.0000e-04

Epoch 14/25

60/60 [==============================] - 59s 987ms/step - loss: 0.5299 - accuracy: 0.7940 - val\_loss: 0.4323 - val\_accuracy: 0.8427 - lr: 1.0000e-04

Epoch 15/25

60/60 [==============================] - 57s 954ms/step - loss: 0.5180 - accuracy: 0.7911 - val\_loss: 0.4385 - val\_accuracy: 0.8458 - lr: 1.0000e-04

Epoch 16/25

60/60 [==============================] - 57s 954ms/step - loss: 0.5009 - accuracy: 0.8081 - val\_loss: 0.4129 - val\_accuracy: 0.8615 - lr: 1.0000e-04

Epoch 17/25

60/60 [==============================] - 58s 960ms/step - loss: 0.4859 - accuracy: 0.8138 - val\_loss: 0.4029 - val\_accuracy: 0.8698 - lr: 1.0000e-04

Epoch 18/25

60/60 [==============================] - 58s 957ms/step - loss: 0.4685 - accuracy: 0.8221 - val\_loss: 0.4127 - val\_accuracy: 0.8427 - lr: 1.0000e-04

Epoch 19/25

60/60 [==============================] - 58s 957ms/step - loss: 0.4528 - accuracy: 0.8245 - val\_loss: 0.4001 - val\_accuracy: 0.8740 - lr: 1.0000e-04

Epoch 20/25

60/60 [==============================] - 59s 979ms/step - loss: 0.4675 - accuracy: 0.8224 - val\_loss: 0.4264 - val\_accuracy: 0.8573 - lr: 1.0000e-04

Epoch 21/25

60/60 [==============================] - 57s 951ms/step - loss: 0.4295 - accuracy: 0.8409 - val\_loss: 0.3778 - val\_accuracy: 0.8698 - lr: 1.0000e-04

Epoch 22/25

60/60 [==============================] - 57s 951ms/step - loss: 0.4273 - accuracy: 0.8398 - val\_loss: 0.3791 - val\_accuracy: 0.8583 - lr: 1.0000e-04

In [ ]:

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs = range(1, len(acc) + 1)

sns.set\_style("whitegrid")

plt.title('Training and validation accuracy')

plt.plot(epochs, acc, 'red', label='Training acc')

plt.plot(epochs, val\_acc, 'blue', label='Validation acc')

plt.legend()

plt.figure()

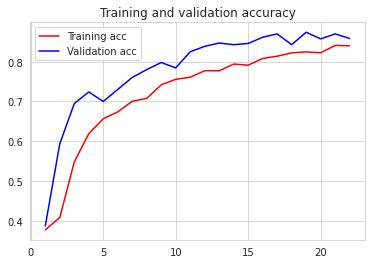
plt.title('Training and validation loss')

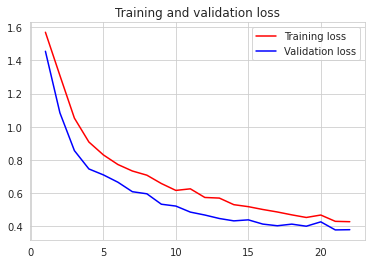
plt.plot(epochs, loss, 'red', label='Training loss')

plt.plot(epochs, val\_loss, 'blue', label='Validation loss')

plt.legend()

plt.show()





In [ ]:

Y\_pred = model4.predict(validation\_generator)

y\_pred = np.argmax(Y\_pred, axis=1)

report1 = classification\_report(validation\_generator.classes, y\_pred, target\_names=rice\_classes, output\_dict=True)

df1 = pd.DataFrame(report1).transpose()

df1

Out[ ]:

|  | **precision** | **recall** | **f1-score** | **support** |
| --- | --- | --- | --- | --- |
| **Arborio** | 0.797980 | 0.822917 | 0.810256 | 192.000000 |
| **Basmati** | 0.930108 | 0.901042 | 0.915344 | 192.000000 |
| **Ipsala** | 0.858537 | 0.916667 | 0.886650 | 192.000000 |
| **Jasmine** | 0.905556 | 0.848958 | 0.876344 | 192.000000 |
| **Karacadag** | 0.884817 | 0.880208 | 0.882507 | 192.000000 |
| **accuracy** | 0.873958 | 0.873958 | 0.873958 | 0.873958 |
| **macro avg** | 0.875399 | 0.873958 | 0.874220 | 960.000000 |
| **weighted avg** | 0.875399 | 0.873958 | 0.874220 | 960.000000 |

The results show a very good trend of the Val/loss curves, although the accuracy is not very high. This prompts us to look for improvements without changing the network.

### **2.5) VGG16 cut 1**

In this model we try to cut the pre\_trained network to an higher level.

In [ ]:

x5 = layer\_dict['block3\_pool'].output

x5 = GlobalAveragePooling2D()(x5)

x5 = Dense(2048,'relu')(x5)

x5 = Dense(512,'relu')(x5)

x5 = Dropout(.15)(x5)

x5 = Dense(128,'relu')(x5)

x5 = Dense(64,'relu')(x5)

x5 = Dropout(.15)(x5)

x5 = Dense(32,'relu')(x5)

x5 = Dense(16,'relu')(x5)

pred5 = Dense(5, activation='softmax')(x5)

In [ ]:

model5 = Model(inputs=pretrained\_model.input, outputs=pred5)

In [ ]:

model5.summary()

Model: "model"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

input\_1 (InputLayer) [(None, 224, 224, 3)] 0

block1\_conv1 (Conv2D) (None, 224, 224, 64) 1792

block1\_conv2 (Conv2D) (None, 224, 224, 64) 36928

block1\_pool (MaxPooling2D) (None, 112, 112, 64) 0

block2\_conv1 (Conv2D) (None, 112, 112, 128) 73856

block2\_conv2 (Conv2D) (None, 112, 112, 128) 147584

block2\_pool (MaxPooling2D) (None, 56, 56, 128) 0

block3\_conv1 (Conv2D) (None, 56, 56, 256) 295168

block3\_conv2 (Conv2D) (None, 56, 56, 256) 590080

block3\_conv3 (Conv2D) (None, 56, 56, 256) 590080

block3\_pool (MaxPooling2D) (None, 28, 28, 256) 0

global\_average\_pooling2d (G (None, 256) 0

lobalAveragePooling2D)

dense (Dense) (None, 2048) 526336

dense\_1 (Dense) (None, 512) 1049088

dropout (Dropout) (None, 512) 0

dense\_2 (Dense) (None, 128) 65664

dense\_3 (Dense) (None, 64) 8256

dropout\_1 (Dropout) (None, 64) 0

dense\_4 (Dense) (None, 32) 2080

dense\_5 (Dense) (None, 16) 528

dense\_6 (Dense) (None, 5) 85

=================================================================

Total params: 3,387,525

Trainable params: 1,652,037

Non-trainable params: 1,735,488

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In [ ]:

model5.compile(

optimizer=tf.keras.optimizers.Adam(lr=0.0001),

loss='categorical\_crossentropy',

metrics=['accuracy'])

/usr/local/lib/python3.7/dist-packages/keras/optimizer\_v2/adam.py:105: UserWarning: The `lr` argument is deprecated, use `learning\_rate` instead.

super(Adam, self).\_\_init\_\_(name, \*\*kwargs)

In [ ]:

history = model5.fit\_generator(

generator=train\_generator,

epochs=25,

validation\_data=validation\_generator,

callbacks=[reduce\_lr, EarlyStopping])

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:5: UserWarning: `Model.fit\_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.

"""

Epoch 1/25

60/60 [==============================] - 57s 921ms/step - loss: 8.3571 - accuracy: 0.3076 - val\_loss: 1.2231 - val\_accuracy: 0.5177 - lr: 1.0000e-04

Epoch 2/25

60/60 [==============================] - 55s 917ms/step - loss: 2.2639 - accuracy: 0.4750 - val\_loss: 0.4441 - val\_accuracy: 0.8656 - lr: 1.0000e-04

Epoch 3/25

60/60 [==============================] - 56s 927ms/step - loss: 1.3115 - accuracy: 0.5867 - val\_loss: 0.5224 - val\_accuracy: 0.7719 - lr: 1.0000e-04

Epoch 4/25

60/60 [==============================] - 55s 911ms/step - loss: 0.8853 - accuracy: 0.6747 - val\_loss: 0.3740 - val\_accuracy: 0.9021 - lr: 1.0000e-04

Epoch 5/25

60/60 [==============================] - 55s 910ms/step - loss: 0.6925 - accuracy: 0.7424 - val\_loss: 0.2728 - val\_accuracy: 0.9156 - lr: 1.0000e-04

Epoch 6/25

60/60 [==============================] - 56s 938ms/step - loss: 0.5798 - accuracy: 0.7828 - val\_loss: 0.3020 - val\_accuracy: 0.9010 - lr: 1.0000e-04

Epoch 7/25

60/60 [==============================] - 55s 910ms/step - loss: 0.5069 - accuracy: 0.8263 - val\_loss: 0.2548 - val\_accuracy: 0.9094 - lr: 1.0000e-04

Epoch 8/25

60/60 [==============================] - 55s 908ms/step - loss: 0.4470 - accuracy: 0.8354 - val\_loss: 0.1471 - val\_accuracy: 0.9688 - lr: 1.0000e-04

Epoch 9/25

60/60 [==============================] - 55s 908ms/step - loss: 0.3923 - accuracy: 0.8620 - val\_loss: 0.1514 - val\_accuracy: 0.9594 - lr: 1.0000e-04

Epoch 10/25

60/60 [==============================] - 55s 909ms/step - loss: 0.3607 - accuracy: 0.8753 - val\_loss: 0.1719 - val\_accuracy: 0.9594 - lr: 1.0000e-04

Epoch 11/25

60/60 [==============================] - ETA: 0s - loss: 0.3967 - accuracy: 0.8659

Epoch 11: ReduceLROnPlateau reducing learning rate to 4.999999873689376e-05.

60/60 [==============================] - 55s 910ms/step - loss: 0.3967 - accuracy: 0.8659 - val\_loss: 0.9894 - val\_accuracy: 0.7625 - lr: 1.0000e-04

In [ ]:

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs = range(1, len(acc) + 1)

sns.set\_style("whitegrid")

plt.title('Training and validation accuracy')

plt.plot(epochs, acc, 'red', label='Training acc')

plt.plot(epochs, val\_acc, 'blue', label='Validation acc')

plt.legend()

plt.figure()

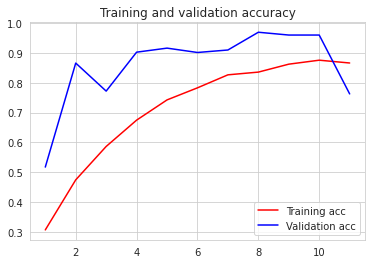
plt.title('Training and validation loss')

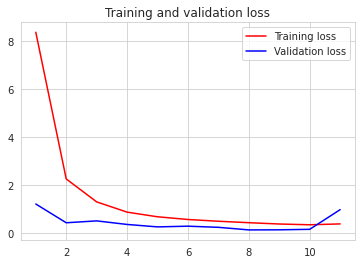
plt.plot(epochs, loss, 'red', label='Training loss')

plt.plot(epochs, val\_loss, 'blue', label='Validation loss')

plt.legend()

plt.show()





the results show a very good performance on the fisrt epochs, 96% on val acc, and a worsening on the last epoch.

### **2.6) VGG16 taglio2**

Now another cut is made on a lower level to see if there is any improvment.

In [ ]:

x6 = layer\_dict['block4\_pool'].output

x6 = GlobalAveragePooling2D()(x6)

x6 = Dense(2048,'relu')(x6)

x6 = Dense(512,'relu')(x6)

x6 = Dropout(.15)(x6)

x6 = Dense(128,'relu')(x6)

x6 = Dense(64,'relu')(x6)

x6 = Dropout(.15)(x6)

x6 = Dense(32,'relu')(x6)

x6 = Dense(16,'relu')(x6)

pred6 = Dense(5, activation='softmax')(x6)

In [ ]:

model6 = Model(inputs=pretrained\_model.input, outputs=pred6)

In [ ]:

model6.compile(

optimizer=tf.keras.optimizers.Adam(lr=0.0001),

loss='categorical\_crossentropy',

metrics=['accuracy'])

/usr/local/lib/python3.7/dist-packages/keras/optimizer\_v2/adam.py:105: UserWarning: The `lr` argument is deprecated, use `learning\_rate` instead.

super(Adam, self).\_\_init\_\_(name, \*\*kwargs)

In [ ]:

history = model6.fit\_generator(

generator=train\_generator,

epochs=25,

validation\_data=validation\_generator,

callbacks=[EarlyStopping])

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:5: UserWarning: `Model.fit\_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.

"""

Epoch 1/25

60/60 [==============================] - 58s 955ms/step - loss: 1.8115 - accuracy: 0.3409 - val\_loss: 1.0119 - val\_accuracy: 0.5792

Epoch 2/25

60/60 [==============================] - 56s 939ms/step - loss: 1.0698 - accuracy: 0.5326 - val\_loss: 0.7205 - val\_accuracy: 0.7542

Epoch 3/25

60/60 [==============================] - 56s 936ms/step - loss: 0.8094 - accuracy: 0.6604 - val\_loss: 0.5155 - val\_accuracy: 0.7594

Epoch 4/25

60/60 [==============================] - 56s 938ms/step - loss: 0.5561 - accuracy: 0.7820 - val\_loss: 0.2765 - val\_accuracy: 0.9333

Epoch 5/25

60/60 [==============================] - 58s 964ms/step - loss: 0.3854 - accuracy: 0.8682 - val\_loss: 0.2312 - val\_accuracy: 0.9271

Epoch 6/25

60/60 [==============================] - 56s 937ms/step - loss: 0.2916 - accuracy: 0.8990 - val\_loss: 0.1710 - val\_accuracy: 0.9615

Epoch 7/25

60/60 [==============================] - 56s 937ms/step - loss: 0.2101 - accuracy: 0.9305 - val\_loss: 0.1100 - val\_accuracy: 0.9760

Epoch 8/25

60/60 [==============================] - 56s 936ms/step - loss: 0.1843 - accuracy: 0.9466 - val\_loss: 0.1381 - val\_accuracy: 0.9594

Epoch 9/25

60/60 [==============================] - 56s 936ms/step - loss: 0.1384 - accuracy: 0.9539 - val\_loss: 0.0759 - val\_accuracy: 0.9708

Epoch 10/25

60/60 [==============================] - 56s 939ms/step - loss: 0.1251 - accuracy: 0.9612 - val\_loss: 0.0692 - val\_accuracy: 0.9750

In [ ]:

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs = range(1, len(acc) + 1)

sns.set\_style("whitegrid")

plt.title('Training and validation accuracy')

plt.plot(epochs, acc, 'red', label='Training acc')

plt.plot(epochs, val\_acc, 'blue', label='Validation acc')

plt.legend()

plt.figure()

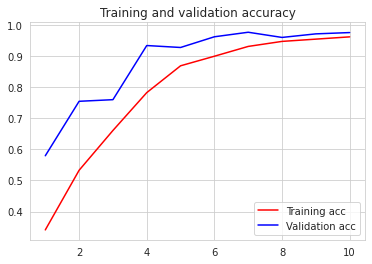
plt.title('Training and validation loss')

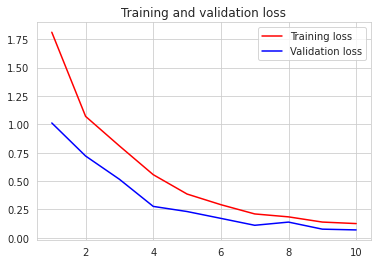
plt.plot(epochs, loss, 'red', label='Training loss')

plt.plot(epochs, val\_loss, 'blue', label='Validation loss')

plt.legend()

plt.show()





The results show a performance that is very similar to one achieved on the ResNet50.