KUL H02A5a Computer Vision: Group Assignment 2

In this group assignment your team will delve into some deep learning applications for computer vision. The assignment will be delivered in the same groups from *Group assignment 1* and you start from this template notebook. The notebook you submit for grading is the last notebook you submit in the Kaggle competition prior to the deadline on **Tuesday 24 May 23:59**. Closely follow these instructions for joining the competition, sharing your notebook with the TAs and making a valid notebook submission to the competition. A notebook submission not only produces a *submission.csv* file that is used to calculate your competition score, it also runs the entire notebook and saves its output as if it were a report. This way it becomes an all-in-one-place document for the TAs to review. As such, please make sure that your final submission notebook is self-contained and fully documented (e.g. provide strong arguments for the design choices that you make). Most likely, this notebook format is not appropriate to run all your experiments at submission time (e.g. the training of CNNs is a memory hungry and time consuming process; due to limited Kaggle resources). It can be a good idea to distribute your code otherwise and only summarize your findings, together with your final predictions, in the submission notebook. For example, you can substitute experiments with some text and figures that you have produced "offline" (e.g. learning curves and results on your internal validation set or even the test set for different architectures, pre-processing pipelines, etc). We advise you to first go through the PDF of this assignment entirely before you really start. Then, it can be a good idea to go through this notebook and use it as your first notebook submission to the competition. You can make use of the *Group assignment 2* forum/discussion board on Toledo if you have any questions. Good luck and have fun!

---

1. Overview

This assignment consists of *three main parts* for which we expect you to provide code and extensive documentation in the notebook:

- Image classification (Sect. 2)
- Semantic segmentation (Sect. 3)
- Adversarial attacks (Sect. 4)

In the first part, you will train an end-to-end neural network for image classification. In the second part, you will do the same for semantic segmentation. For these two tasks we expect you to put a significant effort into optimizing performance and as such competing with fellow students via the Kaggle competition. In the third part, you will try to find and exploit the weaknesses of your classification and/or segmentation network. For the latter there is no competition format, but we do expect you to put significant effort in achieving good performance on the self-posed goal for that part. Finally, we ask you to reflect and produce an overall discussion with links to the lectures and "real world" computer vision (Sect. 5). It is important to note that only a small part of the grade will reflect the actual performance of your networks. However, we do expect all things to work! In general, we will evaluate the correctness of your approach and your understanding of what you have done that you demonstrate in the descriptions and discussions in the final notebook.

1.1 Deep learning resources

If you did not yet explore this in *Group assignment 1 (Sect. 2)*, we recommend using the TensorFlow and/or Keras library for building deep learning models. You can find a nice crash course here.

In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries in
# It is defined by the kaggle/python Docker image: https://github.com/kagg
import numpy as np
np.random.seed(42)
import pandas as pd
import tensorflow as tf
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
import keras
import tensorflow as tf
from keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Conv3D,MaxPooling2D,Dense,Flat
from keras.models import Model, load_model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from keras.callbacks import EarlyStopping, ModelCheckpoint, TensorBoard, Re
import cv2
import os, sys
import matplotlib.pyplot as plt
from tensorflow_addons.metrics import HammingLoss
from tensorflow.random import set_seed
import keras.backend as K
from glob import glob
from PIL import Image
set_seed(42)

import warnings
```

1.2 PASCAL VOC 2009

For this project you will be using the PASCAL VOC 2009 dataset. This dataset consists of colour images of various scenes with different object classes (e.g. animal: *bird, cat,* ...; vehicle: *aeroplane, bicycle,* ...), totalling 20 classes.

In [2]:

```
# Loading the training data
train_df = pd.read_csv('/kaggle/input/kul-h02a5a-computer-vision-ga2-2022/t
labels = train_df.columns
train_df["img"] = [np.load('/kaggle/input/kul-h02a5a-computer-vision-ga2-2(
train_df["seg"] = [np.load('/kaggle/input/kul-h02a5a-computer-vision-ga2-2(
print("The training set contains {} examples.".format(len(train_df)))

# Show some examples
fig, axs = plt.subplots(2, 20, figsize=(10 * 20, 10 * 2))
for i, label in enumerate(labels):
    df = train_df.loc[train_df[label] == 1]
    axs[0, i].imshow(df.iloc[0]["img"], vmin=0, vmax=255)
    axs[0, i].set_title("\n".join(label for label in labels if df.iloc[0][
    axs[0, i].axis("off")
    axs[1, i].imshow(df.iloc[0]["seg"], vmin=0, vmax=20)  # with the absol
    axs[1, i].axis("off")

plt.show()

# The training dataframe contains for each image 20 columns with the groun
train_df.head(1)
```

```
The training set contains 749 examples.
```

Out[2]:

Id	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	...	horse	motorbike
0	0	0	0	0	0	0	0	0	0	1	...	0	0

1 rows × 22 columns

In [3]:
```
# Loading the test data
test_df = pd.read_csv('/kaggle/input/kul-h02a5a-computer-vision-ga2-2022/te
test_df["img"] = [np.load('/kaggle/input/kul-h02a5a-computer-vision-ga2-202
test_df["seg"] = [-1 * np.ones(img.shape[:2], dtype=np.int8) for img in tes
print("The test set contains {} examples.".format(len(test_df)))

# The test dataframe is similar to the training dataframe, but here the va
test_df.head(1)
```

```
The test set contains 750 examples.
```

Out[3]:

Id	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	...	horse	motorbike
0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	...	-1	-1

1 rows × 22 columns

1.3 Your Kaggle submission

Your filled test dataframe (during Sect. 2 and Sect. 3) must be converted to a submission.csv with two rows per example (one for classification and one for segmentation) and with only a single prediction column (the multi-class/label predictions running length encoded). You don't need to edit this section. Just make sure to call this function at the right position in this notebook.

In [4]:

```
def _rle_encode(img):
    """
    Kaggle requires RLE encoded predictions for computation of the Dice sc

    Parameters
    ----------
    img: np.ndarray - binary img array

    Returns
    -------
    rle: String - running length encoded version of img
    """
    pixels = img.flatten()
    pixels = np.concatenate([[0], pixels, [0]])
    runs = np.where(pixels[1:] != pixels[:-1])[0] + 1
    runs[1::2] -= runs[::2]
    rle = ' '.join(str(x) for x in runs)
    return rle

def generate_submission(df):
    """
    Make sure to call this function once after you completed Sect. 2 and Se

    Parameters
    ----------
    df: pd.DataFrame - filled dataframe that needs to be converted

    Returns
    -------
    submission_df: pd.DataFrame - df in submission format.
    """
    df_dict = {"Id": [], "Predicted": []}
    for idx, _ in df.iterrows():
        df_dict["Id"].append(f"{idx}_classification")
        df_dict["Predicted"].append(_rle_encode(np.array(df.loc[idx, labels
        df_dict["Id"].append(f"{idx}_segmentation")
        df_dict["Predicted"].append(_rle_encode(np.array([df.loc[idx, "seg

    submission_df = pd.DataFrame(data=df_dict, dtype=str).set_index("Id")
    submission_df.to_csv("submission.csv")
    return submission_df
```

2. Image classification

The goal here is simple: implement a classification CNN and train it to recognise all 20 classes (and/or background) using the training set and compete on the test set (by filling in the classification columns in the test dataframe).

2.1 CNN From Scratch

In this first section, we will create a model that will only be trained on the training data provided (749 training images for 20 classes). The amount of data given for training is small, thus the neural networks possibility to learn from the examples given is limited. This has two effects: Firstly, the algorithm is not perfectly trained on the training data, second, the model's ability to predict the classes on the basis of the test set is therefore also limited. Since we have few examples, our number one concern should be overfitting. Overfitting happens when a model exposed to too few examples learns patterns that do not generalise to new data, i.e. when the model starts using irrelevant features for making predictions.

In order to prevent the latter and produce good outcomes, several techniques can be applied: Firstly, data augmentation is one way to fight overfitting. By doing so, batches of tensor image data with real-time data augmentation are generated. The augmentation operations are rotation, rescaling, turning, shearing, scale changes and flips. Yet this is not sufficient since our augmented samples are still highly correlated.

Secondly, overfitting can be remedied by increasing the information a model can store and convey. By generating and calculating more features, a model can store more information, but it is also more at risk to start storing irrelevant features. As a consequence, the choice of the number of parameters in the model, i.e. the number of layers and the size of each layer is important.

The first method of creating a model will be from scratch. For this the Keras library will be used. We're however only given 749 training images for 20 classes, which is definitely not a lot and probably not enough to learn this network from scratch.

2.1.1 Data resizing and augmentation

Below, we create two functions. One function to resize all images into the required size as per model for the transfer learning (such as 224, 224 for ResNet50V2). The second function implements data augmentation: In order to overcome the issue of the small amount of training data that we have, we will use data augmentation.

Image data augmentation is a technique that can be used to expand the size of a training dataset by creating alternative versions of each image in the dataset. These alternative versions of the image are created by applying different edits to the images (e.g. zoom, brightness, flipping... etc) which not only expands the training data, but also makes the model more robust to variations in the images that it will have to classify in the test data.

In [5]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

def img_resize(img_df, img_size):
    resized_img = np.empty((len(img_df), img_size[0], img_size[1], 3))
    i = 0
    for img in img_df["img"]:
        img = tf.image.resize_with_pad(img, img_size[0], img_size[1])
        resized_img[i] = img
        i += 1
    return resized_img

def train_data_augmentation(img_df, img_size, batch):
    resized_img = img_resize(img_df, img_size)

    Aug_gen = ImageDataGenerator(
        rotation_range=20,
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

    train_data_generator = Aug_gen.flow(
            x = resized_img,
            y = img_df.iloc[:, :20].to_numpy(),
            batch_size= batch)

    return train_data_generator

def test_data_augmentation(img_df, img_size, batch):
    resized_img = img_resize(img_df, img_size)

    Aug_gen = ImageDataGenerator(rescale=1./255)

    test_data_generator = Aug_gen.flow(
            x = resized_img,
            y = None,
            batch_size= batch)

    return test_data_generator
```

2.1.2 Class distibution and weight adjustments

First, let's check the number of examples we have for each class in our training data.

In [6]:
```
hits = []
for label in train_df[labels]:
    hits.append(np.count_nonzero(train_df[label] == 1))

print(hits)
```

```
[47, 39, 55, 48, 42, 38, 63, 45, 69, 30, 48, 43, 42, 47, 207, 43, 27, 44, 4
0, 51]
```

We seem to have a relatively even distribtion for all the classes other than the element 15 (person) which has 207 examples in our training data. This might induce some bias in our model as it will lean into predicting the class 15 more than the other classes just because of the distribution of our training data.

In order to account for this bias and solve this problem, we will introduce class weights in our model. The code below will be used to calculate the weights for each class.

In [7]:
```
from sklearn.utils.class_weight import compute_class_weight, compute_sample
from sklearn.preprocessing import LabelEncoder
# Create a pd.series that represents the categorical class of each one-hot
y_classes = train_df[labels].idxmax(1, skipna=False)

# Instantiate the label encoder
lbl_enc = LabelEncoder()

# Fit the label encoder to our label series
lbl_enc.fit(list(y_classes))

# Create integer based labels Series
y_integers = lbl_enc.transform(list(y_classes))

# Create dict of labels : integer representation
labels_and_integers = dict(zip(y_classes, y_integers))
class_weights = compute_class_weight(class_weight ='balanced', classes = n
sample_weights = compute_sample_weight('balanced', y_integers)
class_weights_dict = dict(zip(lbl_enc.transform(list(lbl_enc.classes_)), c
```

2.1.3 CNN model

We are using a CNN model based on the AlexNet architecture. The model consists of:

- 5 convolutional layers: Convolutions allows the extraction of usefel informations such as edges in each step. Each convolutional layer is followed by a hidden relu activation layer. Of these 5 layers, 3 are also followed by a pooling layer (in order to decrease the size of the input and the computational costs) with batch-normalization layers between each activation adn pooling layers.

- 3 fully-connected layers: These layers connec all neurons of one layer with all neurons of a second allowing the model to keep the context of the whole image into account when predicting a class. After all, an image is only recognisable because of the combination, one edge on itself doesn't say much. Each layer is followed by a hidden relu activation layer in addition to one dropout layer to avoid overfitting.

Alexnet was constructed with images of size 227x227x3. Since we are only replicating the Alexnet architecture but not using a pretrained model, the image size is not bound to 227x227x3. In the following, we applied a shape of 150x150x.

In [8]:

```
from keras.layers import Activation, Dropout, Flatten, Dense, Conv2D, MaxPo

batch_size = 16
image_size = (150, 150)
image_shape= (image_size[0], image_size[1], 3)

CNN_model = Sequential()

# Convolutional Layers
CNN_model.add(Conv2D(64, (7, 7), input_shape= image_shape))
CNN_model.add(Activation('relu'))
CNN_model.add(BatchNormalization())
CNN_model.add(MaxPooling2D(pool_size=(3, 3)))

CNN_model.add(Conv2D(128, (5, 5)))
CNN_model.add(Activation('relu'))
CNN_model.add(BatchNormalization())
CNN_model.add(MaxPooling2D(pool_size=(3, 3)))

CNN_model.add(Conv2D(145, (3, 3)))
CNN_model.add(Activation('relu'))

CNN_model.add(Conv2D(145, (3, 3)))
CNN_model.add(Activation('relu'))

CNN_model.add(Conv2D(128, (3, 3)))
CNN_model.add(Activation('relu'))
CNN_model.add(BatchNormalization())
CNN_model.add(MaxPooling2D(pool_size=(4, 4)))

# Fully connected layers
CNN_model.add(Flatten())

CNN_model.add(Dense(64))
CNN_model.add(Activation('relu'))

CNN_model.add(Dense(64))
CNN_model.add(Activation('relu'))
CNN_model.add(Dropout(0.5))

CNN_model.add(Dense(20))
CNN_model.add(Activation('sigmoid'))

CNN_model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics= 'accuracy')

# Model summary
CNN_model.summary()
```

```
2022-05-25 18:40:17.207918: I tensorflow/stream_executor/cuda/cuda_gpu_exec
utor.cc:937] successful NUMA node read from SysFS had negative value (-1),
but there must be at least one NUMA node, so returning NUMA node zero
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 144, 144, 64)      9472
_________________________________________________________________
activation (Activation)      (None, 144, 144, 64)      0
_________________________________________________________________
batch_normalization (BatchNo (None, 144, 144, 64)      256
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 48, 48, 64)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 44, 44, 128)       204928
_________________________________________________________________
activation_1 (Activation)    (None, 44, 44, 128)       0
_________________________________________________________________
batch_normalization_1 (Batch (None, 44, 44, 128)       512
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 145)       167185
_________________________________________________________________
activation_2 (Activation)    (None, 12, 12, 145)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 145)       189370
_________________________________________________________________
activation_3 (Activation)    (None, 10, 10, 145)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 128)         167168
_________________________________________________________________
activation_4 (Activation)    (None, 8, 8, 128)         0
_________________________________________________________________
batch_normalization_2 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 2, 2, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 64)                32832
_________________________________________________________________
activation_5 (Activation)    (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                4160
_________________________________________________________________
activation_6 (Activation)    (None, 64)                0
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 20)                1300
_________________________________________________________________
activation_7 (Activation)    (None, 20)                0
=================================================================
Total params: 777,695
Trainable params: 777,055
Non-trainable params: 640
_________________________________________________________________
```

```
2022-05-25 18:40:17.332689: I tensorflow/stream_executor/cuda/cuda_gpu_exec
utor.cc:937] successful NUMA node read from SysFS had negative value (-1),
but there must be at least one NUMA node, so returning NUMA node zero
2022-05-25 18:40:17.334019: I tensorflow/stream_executor/cuda/cuda_gpu_exec
utor.cc:937] successful NUMA node read from SysFS had negative value (-1),
but there must be at least one NUMA node, so returning NUMA node zero
2022-05-25 18:40:17.336388: I tensorflow/core/platform/cpu_feature_guard.cc
:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network L
ibrary (oneDNN) to use the following CPU instructions in performance-critic
al operations:  AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
2022-05-25 18:40:17.337715: I tensorflow/stream_executor/cuda/cuda_gpu_exec
utor.cc:937] successful NUMA node read from SysFS had negative value (-1),
but there must be at least one NUMA node, so returning NUMA node zero
2022-05-25 18:40:17.338825: I tensorflow/stream_executor/cuda/cuda_gpu_exec
utor.cc:937] successful NUMA node read from SysFS had negative value (-1),
but there must be at least one NUMA node, so returning NUMA node zero
2022-05-25 18:40:17.342255: I tensorflow/stream_executor/cuda/cuda_gpu_exec
utor.cc:937] successful NUMA node read from SysFS had negative value (-1),
but there must be at least one NUMA node, so returning NUMA node zero
2022-05-25 18:40:19.842418: I tensorflow/stream_executor/cuda/cuda_gpu_exec
utor.cc:937] successful NUMA node read from SysFS had negative value (-1),
but there must be at least one NUMA node, so returning NUMA node zero
2022-05-25 18:40:19.843352: I tensorflow/stream_executor/cuda/cuda_gpu_exec
utor.cc:937] successful NUMA node read from SysFS had negative value (-1),
but there must be at least one NUMA node, so returning NUMA node zero
2022-05-25 18:40:19.844076: I tensorflow/stream_executor/cuda/cuda_gpu_exec
utor.cc:937] successful NUMA node read from SysFS had negative value (-1),
but there must be at least one NUMA node, so returning NUMA node zero
2022-05-25 18:40:19.844680: I tensorflow/core/common_runtime/gpu/gpu_device
.cc:1510] Created device /job:localhost/replica:0/task:0/device:GPU:0 with
15403 MB memory:  -> device: 0, name: Tesla P100-PCIE-16GB, pci bus id: 000
0:00:04.0, compute capability: 6.0
```

We will train our CNN model on the augmentated data (using the class weights)

In [9]:

```
train_generator = train_data_augmentation(train_df, image_size, batch_size

# Code line below if we want to train this model

CNN_model.fit(train_generator, steps_per_epoch = 749 // batch_size, class_w
```

```
2022-05-25 18:40:24.545139: I tensorflow/compiler/mlir/mlir_graph_optimizat
ion_pass.cc:185] None of the MLIR Optimization Passes are enabled (register
ed 2)
Epoch 1/150
2022-05-25 18:40:26.377242: I tensorflow/stream_executor/cuda/cuda_dnn.cc:3
69] Loaded cuDNN version 8005
46/46 [==============================] - 11s 77ms/step - loss: 0.4873 - acc
uracy: 0.0477
Epoch 2/150
46/46 [==============================] - 4s 85ms/step - loss: 0.3251 - accu
racy: 0.0628
Epoch 3/150
46/46 [==============================] - 4s 83ms/step - loss: 0.2963 - accu
racy: 0.0778
Epoch 4/150
46/46 [==============================] - 4s 81ms/step - loss: 0.2808 - accu
```

```
racy: 0.0791
Epoch 5/150
46/46 [==============================] - 4s 89ms/step - loss: 0.2736 - accu
racy: 0.0928
Epoch 6/150
46/46 [==============================] - 4s 82ms/step - loss: 0.2669 - accu
racy: 0.0928
Epoch 7/150
46/46 [==============================] - 5s 106ms/step - loss: 0.2636 - acc
uracy: 0.0996
Epoch 8/150
46/46 [==============================] - 4s 81ms/step - loss: 0.2579 - accu
racy: 0.0928
Epoch 9/150
46/46 [==============================] - 4s 93ms/step - loss: 0.2614 - accu
racy: 0.1023
Epoch 10/150
46/46 [==============================] - 4s 80ms/step - loss: 0.2585 - accu
racy: 0.0846
Epoch 11/150
46/46 [==============================] - 4s 87ms/step - loss: 0.2529 - accu
racy: 0.1091
Epoch 12/150
46/46 [==============================] - 4s 81ms/step - loss: 0.2477 - accu
racy: 0.1160
Epoch 13/150
46/46 [==============================] - 4s 79ms/step - loss: 0.2447 - accu
racy: 0.1187
Epoch 14/150
46/46 [==============================] - 5s 107ms/step - loss: 0.2455 - acc
uracy: 0.1160
Epoch 15/150
46/46 [==============================] - 4s 86ms/step - loss: 0.2435 - accu
racy: 0.1160
Epoch 16/150
46/46 [==============================] - 4s 87ms/step - loss: 0.2393 - accu
racy: 0.1132
Epoch 17/150
46/46 [==============================] - 4s 79ms/step - loss: 0.2423 - accu
racy: 0.1255
Epoch 18/150
46/46 [==============================] - 4s 79ms/step - loss: 0.2399 - accu
racy: 0.1359
Epoch 19/150
46/46 [==============================] - 4s 83ms/step - loss: 0.2359 - accu
racy: 0.1351
Epoch 20/150
46/46 [==============================] - 4s 78ms/step - loss: 0.2365 - accu
racy: 0.1310
Epoch 21/150
46/46 [==============================] - 4s 77ms/step - loss: 0.2323 - accu
racy: 0.1528
Epoch 22/150
46/46 [==============================] - 5s 103ms/step - loss: 0.2307 - acc
uracy: 0.1337
Epoch 23/150
46/46 [==============================] - 4s 85ms/step - loss: 0.2322 - accu
racy: 0.1269
Epoch 24/150
46/46 [==============================] - 4s 89ms/step - loss: 0.2274 - accu
racy: 0.1337
```

```
Epoch 25/150
46/46 [==============================] - 4s 81ms/step - loss: 0.2282 - accu
racy: 0.1569
Epoch 26/150
46/46 [==============================] - 4s 78ms/step - loss: 0.2258 - accu
racy: 0.1460
Epoch 27/150
46/46 [==============================] - 4s 84ms/step - loss: 0.2266 - accu
racy: 0.1528
Epoch 28/150
46/46 [==============================] - 4s 77ms/step - loss: 0.2248 - accu
racy: 0.1399
Epoch 29/150
46/46 [==============================] - 5s 100ms/step - loss: 0.2221 - acc
uracy: 0.1583
Epoch 30/150
46/46 [==============================] - 4s 89ms/step - loss: 0.2202 - accu
racy: 0.1528
Epoch 31/150
46/46 [==============================] - 4s 88ms/step - loss: 0.2229 - accu
racy: 0.1651
Epoch 32/150
46/46 [==============================] - 4s 79ms/step - loss: 0.2203 - accu
racy: 0.1692
Epoch 33/150
46/46 [==============================] - 4s 79ms/step - loss: 0.2165 - accu
racy: 0.1733
Epoch 34/150
46/46 [==============================] - 4s 85ms/step - loss: 0.2187 - accu
racy: 0.1651
Epoch 35/150
46/46 [==============================] - 4s 78ms/step - loss: 0.2185 - accu
racy: 0.1637
Epoch 36/150
46/46 [==============================] - 4s 81ms/step - loss: 0.2144 - accu
racy: 0.1733
Epoch 37/150
46/46 [==============================] - 4s 93ms/step - loss: 0.2119 - accu
racy: 0.1658
Epoch 38/150
46/46 [==============================] - 4s 79ms/step - loss: 0.2127 - accu
racy: 0.1719
Epoch 39/150
46/46 [==============================] - 4s 89ms/step - loss: 0.2083 - accu
racy: 0.1965
Epoch 40/150
46/46 [==============================] - 4s 77ms/step - loss: 0.2099 - accu
racy: 0.1842
Epoch 41/150
46/46 [==============================] - 4s 83ms/step - loss: 0.2109 - accu
racy: 0.1937
Epoch 42/150
46/46 [==============================] - 4s 77ms/step - loss: 0.2081 - accu
racy: 0.1910
Epoch 43/150
46/46 [==============================] - 4s 87ms/step - loss: 0.2058 - accu
racy: 0.1951
Epoch 44/150
46/46 [==============================] - 4s 93ms/step - loss: 0.2075 - accu
racy: 0.1869
Epoch 45/150
```

```
46/46 [==============================] - 4s 78ms/step - loss: 0.2032 - accu
racy: 0.1883
Epoch 46/150
46/46 [==============================] - 4s 84ms/step - loss: 0.2049 - accu
racy: 0.1883
Epoch 47/150
46/46 [==============================] - 4s 77ms/step - loss: 0.1983 - accu
racy: 0.2128
Epoch 48/150
46/46 [==============================] - 4s 85ms/step - loss: 0.2008 - accu
racy: 0.2019
Epoch 49/150
46/46 [==============================] - 4s 80ms/step - loss: 0.2017 - accu
racy: 0.2115
Epoch 50/150
46/46 [==============================] - 6s 126ms/step - loss: 0.1952 - acc
uracy: 0.2087
Epoch 51/150
46/46 [==============================] - 4s 79ms/step - loss: 0.1973 - accu
racy: 0.2469
Epoch 52/150
46/46 [==============================] - 4s 80ms/step - loss: 0.1963 - accu
racy: 0.2224
Epoch 53/150
46/46 [==============================] - 4s 80ms/step - loss: 0.2000 - accu
racy: 0.2101
Epoch 54/150
46/46 [==============================] - 4s 81ms/step - loss: 0.1990 - accu
racy: 0.2224
Epoch 55/150
46/46 [==============================] - 4s 88ms/step - loss: 0.1921 - accu
racy: 0.2060
Epoch 56/150
46/46 [==============================] - 4s 83ms/step - loss: 0.1878 - accu
racy: 0.2319
Epoch 57/150
46/46 [==============================] - 5s 101ms/step - loss: 0.1855 - acc
uracy: 0.2497
Epoch 58/150
46/46 [==============================] - 5s 101ms/step - loss: 0.1833 - acc
uracy: 0.2510
Epoch 59/150
46/46 [==============================] - 4s 81ms/step - loss: 0.1894 - accu
racy: 0.2347
Epoch 60/150
46/46 [==============================] - 4s 85ms/step - loss: 0.1801 - accu
racy: 0.2810
Epoch 61/150
46/46 [==============================] - 4s 81ms/step - loss: 0.1871 - accu
racy: 0.2715
Epoch 62/150
46/46 [==============================] - 4s 92ms/step - loss: 0.1781 - accu
racy: 0.2524
Epoch 63/150
46/46 [==============================] - 4s 85ms/step - loss: 0.1748 - accu
racy: 0.2756
Epoch 64/150
46/46 [==============================] - 5s 116ms/step - loss: 0.1821 - acc
uracy: 0.2838
Epoch 65/150
46/46 [==============================] - 4s 82ms/step - loss: 0.1824 - accu
```

```
racy: 0.2592
Epoch 66/150
46/46 [==============================] - 4s 87ms/step - loss: 0.1771 - accu
racy: 0.2606
Epoch 67/150
46/46 [==============================] - 4s 80ms/step - loss: 0.1748 - accu
racy: 0.2947
Epoch 68/150
46/46 [==============================] - 4s 80ms/step - loss: 0.1729 - accu
racy: 0.2783
Epoch 69/150
46/46 [==============================] - 4s 80ms/step - loss: 0.1666 - accu
racy: 0.3165
Epoch 70/150
46/46 [==============================] - 4s 83ms/step - loss: 0.1710 - accu
racy: 0.2947
Epoch 71/150
46/46 [==============================] - 5s 115ms/step - loss: 0.1685 - acc
uracy: 0.3070
Epoch 72/150
46/46 [==============================] - 4s 88ms/step - loss: 0.1649 - accu
racy: 0.3097
Epoch 73/150
46/46 [==============================] - 4s 79ms/step - loss: 0.1657 - accu
racy: 0.3083
Epoch 74/150
46/46 [==============================] - 4s 86ms/step - loss: 0.1676 - accu
racy: 0.3206
Epoch 75/150
46/46 [==============================] - 4s 81ms/step - loss: 0.1670 - accu
racy: 0.3111
Epoch 76/150
46/46 [==============================] - 4s 86ms/step - loss: 0.1592 - accu
racy: 0.3274
Epoch 77/150
46/46 [==============================] - 6s 132ms/step - loss: 0.1603 - acc
uracy: 0.3042
Epoch 78/150
46/46 [==============================] - 4s 82ms/step - loss: 0.1598 - accu
racy: 0.3370
Epoch 79/150
46/46 [==============================] - 4s 91ms/step - loss: 0.1584 - accu
racy: 0.3370
Epoch 80/150
46/46 [==============================] - 4s 82ms/step - loss: 0.1607 - accu
racy: 0.3192
Epoch 81/150
46/46 [==============================] - 4s 82ms/step - loss: 0.1603 - accu
racy: 0.3465
Epoch 82/150
46/46 [==============================] - 4s 91ms/step - loss: 0.1593 - accu
racy: 0.3383
Epoch 83/150
46/46 [==============================] - 4s 85ms/step - loss: 0.1504 - accu
racy: 0.3711
Epoch 84/150
46/46 [==============================] - 6s 133ms/step - loss: 0.1544 - acc
uracy: 0.3492
Epoch 85/150
46/46 [==============================] - 5s 98ms/step - loss: 0.1540 - accu
racy: 0.3656
```

```
Epoch 86/150
46/46 [==============================] - 4s 93ms/step - loss: 0.1457 - accu
racy: 0.3793
Epoch 87/150
46/46 [==============================] - 4s 80ms/step - loss: 0.1462 - accu
racy: 0.3902
Epoch 88/150
46/46 [==============================] - 4s 81ms/step - loss: 0.1481 - accu
racy: 0.3779
Epoch 89/150
46/46 [==============================] - 4s 89ms/step - loss: 0.1414 - accu
racy: 0.3872
Epoch 90/150
46/46 [==============================] - 4s 84ms/step - loss: 0.1441 - accu
racy: 0.3765
Epoch 91/150
46/46 [==============================] - 4s 90ms/step - loss: 0.1428 - accu
racy: 0.3956
Epoch 92/150
46/46 [==============================] - 6s 132ms/step - loss: 0.1387 - acc
uracy: 0.3970
Epoch 93/150
46/46 [==============================] - 4s 86ms/step - loss: 0.1391 - accu
racy: 0.4243
Epoch 94/150
46/46 [==============================] - 4s 82ms/step - loss: 0.1373 - accu
racy: 0.4229
Epoch 95/150
46/46 [==============================] - 4s 82ms/step - loss: 0.1317 - accu
racy: 0.4407
Epoch 96/150
46/46 [==============================] - 4s 92ms/step - loss: 0.1343 - accu
racy: 0.4311
Epoch 97/150
46/46 [==============================] - 4s 81ms/step - loss: 0.1326 - accu
racy: 0.4038
Epoch 98/150
46/46 [==============================] - 4s 89ms/step - loss: 0.1387 - accu
racy: 0.4175
Epoch 99/150
46/46 [==============================] - 5s 116ms/step - loss: 0.1284 - acc
uracy: 0.4229
Epoch 100/150
46/46 [==============================] - 4s 90ms/step - loss: 0.1313 - accu
racy: 0.4052
Epoch 101/150
46/46 [==============================] - 4s 94ms/step - loss: 0.1365 - accu
racy: 0.4147
Epoch 102/150
46/46 [==============================] - 4s 87ms/step - loss: 0.1488 - accu
racy: 0.3806
Epoch 103/150
46/46 [==============================] - 4s 90ms/step - loss: 0.1319 - accu
racy: 0.4502
Epoch 104/150
46/46 [==============================] - 4s 82ms/step - loss: 0.1294 - accu
racy: 0.4352
Epoch 105/150
46/46 [==============================] - 4s 87ms/step - loss: 0.1283 - accu
racy: 0.4638
Epoch 106/150
```

```
46/46 [==============================] - 5s 113ms/step - loss: 0.1308 - acc
uracy: 0.4516
Epoch 107/150
46/46 [==============================] - 4s 79ms/step - loss: 0.1226 - accu
racy: 0.4939
Epoch 108/150
46/46 [==============================] - 4s 85ms/step - loss: 0.1200 - accu
racy: 0.4925
Epoch 109/150
46/46 [==============================] - 5s 97ms/step - loss: 0.1234 - accu
racy: 0.4679
Epoch 110/150
46/46 [==============================] - 4s 92ms/step - loss: 0.1204 - accu
racy: 0.4761
Epoch 111/150
46/46 [==============================] - 4s 80ms/step - loss: 0.1206 - accu
racy: 0.4693
Epoch 112/150
46/46 [==============================] - 4s 79ms/step - loss: 0.1139 - accu
racy: 0.4843
Epoch 113/150
46/46 [==============================] - 4s 88ms/step - loss: 0.1113 - accu
racy: 0.5007
Epoch 114/150
46/46 [==============================] - 4s 91ms/step - loss: 0.1122 - accu
racy: 0.5075
Epoch 115/150
46/46 [==============================] - 4s 85ms/step - loss: 0.1138 - accu
racy: 0.5116
Epoch 116/150
46/46 [==============================] - 4s 78ms/step - loss: 0.1133 - accu
racy: 0.5116
Epoch 117/150
46/46 [==============================] - 4s 89ms/step - loss: 0.1108 - accu
racy: 0.5020
Epoch 118/150
46/46 [==============================] - 4s 79ms/step - loss: 0.1083 - accu
racy: 0.5348
Epoch 119/150
46/46 [==============================] - 4s 79ms/step - loss: 0.1104 - accu
racy: 0.5048
Epoch 120/150
46/46 [==============================] - 4s 86ms/step - loss: 0.1104 - accu
racy: 0.5048
Epoch 121/150
46/46 [==============================] - 5s 116ms/step - loss: 0.1123 - acc
uracy: 0.5130
Epoch 122/150
46/46 [==============================] - 4s 80ms/step - loss: 0.1069 - accu
racy: 0.5321
Epoch 123/150
46/46 [==============================] - 4s 94ms/step - loss: 0.1106 - accu
racy: 0.5034
Epoch 124/150
46/46 [==============================] - 4s 86ms/step - loss: 0.1072 - accu
racy: 0.5375
Epoch 125/150
46/46 [==============================] - 4s 79ms/step - loss: 0.1093 - accu
racy: 0.5607
Epoch 126/150
46/46 [==============================] - 4s 80ms/step - loss: 0.1056 - accu
```

```
racy: 0.5457
Epoch 127/150
46/46 [==============================] - 4s 92ms/step - loss: 0.1013 - accu
racy: 0.5443
Epoch 128/150
46/46 [==============================] - 4s 85ms/step - loss: 0.0969 - accu
racy: 0.5716
Epoch 129/150
46/46 [==============================] - 4s 76ms/step - loss: 0.0969 - accu
racy: 0.5880
Epoch 130/150
46/46 [==============================] - 4s 76ms/step - loss: 0.0999 - accu
racy: 0.5730
Epoch 131/150
46/46 [==============================] - 5s 99ms/step - loss: 0.1015 - accu
racy: 0.5375
Epoch 132/150
46/46 [==============================] - 4s 79ms/step - loss: 0.0963 - accu
racy: 0.5703
Epoch 133/150
46/46 [==============================] - 4s 77ms/step - loss: 0.0973 - accu
racy: 0.5607
Epoch 134/150
46/46 [==============================] - 4s 83ms/step - loss: 0.0932 - accu
racy: 0.6016
Epoch 135/150
46/46 [==============================] - 5s 115ms/step - loss: 0.0963 - acc
uracy: 0.5842
Epoch 136/150
46/46 [==============================] - 4s 84ms/step - loss: 0.0950 - accu
racy: 0.6030
Epoch 137/150
46/46 [==============================] - 4s 80ms/step - loss: 0.0925 - accu
racy: 0.6003
Epoch 138/150
46/46 [==============================] - 5s 109ms/step - loss: 0.0930 - acc
uracy: 0.5948
Epoch 139/150
46/46 [==============================] - 4s 76ms/step - loss: 0.0969 - accu
racy: 0.5648
Epoch 140/150
46/46 [==============================] - 4s 78ms/step - loss: 0.0961 - accu
racy: 0.5853
Epoch 141/150
46/46 [==============================] - 4s 83ms/step - loss: 0.0889 - accu
racy: 0.6044
Epoch 142/150
46/46 [==============================] - 4s 97ms/step - loss: 0.0865 - accu
racy: 0.6112
Epoch 143/150
46/46 [==============================] - 4s 91ms/step - loss: 0.0878 - accu
racy: 0.5962
Epoch 144/150
46/46 [==============================] - 4s 76ms/step - loss: 0.0872 - accu
racy: 0.6153
Epoch 145/150
46/46 [==============================] - 5s 102ms/step - loss: 0.0907 - acc
uracy: 0.6030
Epoch 146/150
46/46 [==============================] - 4s 76ms/step - loss: 0.0862 - accu
racy: 0.6248
```

```
Epoch 147/150
46/46 [==============================] - 4s 76ms/step - loss: 0.0789 - accu
racy: 0.6344
Epoch 148/150
46/46 [==============================] - 4s 82ms/step - loss: 0.0801 - accu
racy: 0.6276
Epoch 149/150
46/46 [==============================] - 4s 90ms/step - loss: 0.0867 - accu
racy: 0.6262
Epoch 150/150
46/46 [==============================] - 5s 109ms/step - loss: 0.0914 - acc
uracy: 0.5839
```

Out[9]: <keras.callbacks.History at 0x7f1a9a1e3690>

From the self-implemented CNN model based on AlexNet architecture we can see the following: Training is slow, the accuracy is at about 0.66 after 150 epochs, the model loss is continuously reducing. However, given the small dataset, the training gains are small and we risk overfitting on the training set.

2.2 Transfer Learning

Our CNN model trained from scratch is not performing well mainly due to the ammount of training data that we have even after we augmented the data. In order to have a better accuracy we need a larger ammount of training data which we do not have. An alternative solution is to use transfer learning, by taking a model that has already been trained on a huge collection of data and then retrain the final learning on our training data.

We experimented with three models: VGG16, ResNet50V2 and MobileNetV2. The models differ by their size and design of layers and the data they have been trained on.

The choice of the different models & datasets for transfer learning will be explained later on. Yet it has an influence on the performance of the algorithm:

If the source data for the pretrained model is similar to the target model, overfitting is a potential problem. If the source model and source data are greatly differing from each other, achieving the target model can take a longer time.

All three models can be found below. However, the one that performed best for us and the one we will be using in our predictions is MobileNetV2.

2.2.1 VGG16

In the following, the model VGG16 is used to pretrain the network on the VGG16 dataset. The VGG16 network by Karen Simonyan and Andrew Zisserman of the Visual Geometry Group Lab of Oxford University in 2014 in the paper “Very deep convolutional networks for large-scale image recognition”. The inputs of VGG16 are of fixed size of 224x224 and have RGB channels. The corresponding tensor is 440x (224,224,3) and the output is 440x26 – thus a classification value for each class. Keras provides further options to pre-process the images before feeding them which transforms RBG to BGR channels and each colour channel is zero-centered with respect to the ImageNet dataset, without scaling.

Data as prepared as follows: 1) all images were resized to the required shape of 224,224,3, all input categories were reshaped to a one-hot encoded array for each of the 440 input images. The tensors were then fed into the model. The input to the network is image of dimensions (224, 224, 3). The VGG16 network combines several layers with different channel and filter size. The following layers have then the following dimensions: 64 channels of 3x3 filter, same padding, max pool layer of stride (2, 2), two layers with convolution of 256 filter size and filter size (3, 3), max pooling layer of stride (2, 2), 2 convolution layers of filter size (3, 3) and 256 filter, followed by 2 sets of 3 convolution layer and a max pool layer. Each have 512 filters of (3, 3) size with same padding, then two convolution and max pooling layers. The filter is a 3x3 kernel in comparison to 11x11 in AlexNet. A 1x1 pixel filter is used to manipulate the number of input channels. A padding of 1-pixel (same padding) done after each convolution layer is done to maintain the spatial features of the image. To make sure the probabilities on the predicted y values add up to 1, a softmax function was implemented.

The last layers on top of the VGG16 network are then working on the PASCAL VOC dataset. The model was not trained on the PASCAL VOC data but did not provide satisfactory results.

Evaluation

The current model still suffers overfitting: the training error is low, but the validation error does not reduce. This is not very satisfactory, as it points towards the model fitting well on the training data but not on the test data, thus the models capability of generalising on the test data is poor. Similarly, the accuracy of the training set is very good, whereas the accuracy of the validation set remains low.

We can remedy this by adding more parameters to the model in the fully connected layers, such as via Dense layers. With more model parameters, the model's ability to overfit on the training set is reduced. To remedy the situation we included three more layers: 1x Flattening layer, one Dense layer of 512 with Relu activation and one Dropout of 0.5.

In [10]:

```
batch_size = 10
image_size = (224, 224)
image_shape= (image_size[0], image_size[1], 3)
train_generator_VGG16 = train_data_augmentation(train_df, image_size, batcl

from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.optimizers import SGD

# We upload the VGG16 model without the last layer (for the output)
base_VGG16_model = VGG16(weights="imagenet", include_top=False, input_tens

# We create layers to merge with the VGG16 model in order to give us an ou
head_VGG16_model = base_VGG16_model.output
head_VGG16_model = Flatten(name="flatten")(head_VGG16_model)
head_VGG16_model = Dense(512, activation="relu")(head_VGG16_model)
head_VGG16_model = Dropout(0.5)(head_VGG16_model)
head_VGG16_model = Dense(20, activation="softmax")(head_VGG16_model)


# We merge the base VGG16 model with our newly created layers
merged_VGG16_model = Model(inputs=base_VGG16_model.input, outputs=head_VGG

# We freeze the pretrained layers of the base VGG16 model
for layer in base_VGG16_model.layers:
    layer.trainable = False

# We compile the whole merged model
merged_VGG16_model.compile(loss='categorical_crossentropy', optimizer=SGD(

# Model summary
merged_VGG16_model.summary()

# Code line below if we want to train this model

#merged_VGG16_model.fit(train_generator_VGG16,steps_per_epoch=749 // batch_
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-appli
cations/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 0s 0us/step
58900480/58889256 [==============================] - 0s 0us/step
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
_________________________________________________________________
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
_________________________________________________________________
dense_3 (Dense)              (None, 512)               12845568
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 20)                10260
=================================================================
Total params: 27,570,516
Trainable params: 12,855,828
Non-trainable params: 14,714,688
_________________________________________________________________
```

The current model still suffers overfitting: the training error is low, but the validation error does not reduce. This is not very satisfactory, as it points towards the model fitting well on the training data but not on the test data. Similarly, the accuracy of the training set is very well, whereas the accuracy of the validation set remains low.

We can remedy this by adding more parameters to the model, such as via Dense layers. With more model parameters, the models ability to overfit on the training set is reduced. To remedy the situation I included three more layers: 2x Dropout of 0.2 size and a Dense layer with 200 classes and a 'relu' activation.

We also need to take into account that we have imbalanced classes, thus not all classes are equally represented in the dataset.

2.2.2 ResNet50V2

After the subsequent successful architecture (AlexNet), each new layer uses more in a deep neural network to reduce the error rate. If the number of layers is higher, a common problem in deep learning is associated with the gradient called vanishing or exploding. This causes the gradient to be 0 or too large. If the number of layers increases, the training and test error rate also increases. It becomes difficult to train bigger models, the accuracy is saturated and then degrades.

In order to solve the problem of the vanishing/exploding gradient, Resnet introduced the concept called Residual Network using socalled skip connections. The skip connection skips training from a few layers and connects directly to the output. The approach behind this network is instead of layers learn the underlying mapping, the network is allowed to fit the residual mapping.

In doing so, the training of very dee neural networks can be done, without the problems caused by vanishing/exploding gradient.
The authors of the paper experimented on 100-1000 layers on CIFAR-10 dataset. The dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. The classes are: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.

The network architecture was inspired by VGG neural networks (VGG-16, VGG-19), with the convolutional networks having 3×3 filters. ResNets do have fewer filters and lower complexity, the ResNet50 has 50 layers. The inputs of ResNet50V2 are of fixed size of 224x224. The used network is Resnet50V2 which is an improvement on the ResNet50.

In [11]:

```
batch_size = 64
image_size = (224, 224)
image_shape= (image_size[0], image_size[1], 3)
train_generator_ResNet50V2 = train_data_augmentation(train_df, image_size,

from tensorflow.keras.applications import ResNet50V2
from tensorflow.keras import layers


# We upload the ResNet50V2 model without the last layer (for the output)
base_ResNet50V2_model = ResNet50V2(weights="imagenet", include_top = False

# We create layers to merge with the ResNet50V2 model in order to give us
head_ResNet50V2_model = base_ResNet50V2_model.layers[-1].output
head_ResNet50V2_model = Flatten(name="flatten")(head_ResNet50V2_model)
head_ResNet50V2_model = Model(inputs=base_ResNet50V2_model.input, outputs=

# We freeze the pretrained layers of the base ResNet50V2 model
for layer in head_ResNet50V2_model.layers:
    layer.trainable = False

# We merge the base ResNet50V2 model with our newly created layers
merged_ResNet50V2_model = Sequential()
merged_ResNet50V2_model.add(head_ResNet50V2_model)
merged_ResNet50V2_model.add(layers.Dense(512, activation='relu', input_dim=
merged_ResNet50V2_model.add(layers.Dropout(0.3))
merged_ResNet50V2_model.add(layers.Dense(512, activation='relu'))
merged_ResNet50V2_model.add(layers.Dropout(0.3))
merged_ResNet50V2_model.add(layers.Dense(units=20, activation='sigmoid'))

# We compile the whole merged model
merged_ResNet50V2_model.compile(loss='binary_crossentropy',optimizer="sgd"

# Model summary
merged_ResNet50V2_model.summary()

# Code line below if we want to train this model

#merged_ResNet50V2_model.fit(train_generator_ResNet50V2,steps_per_epoch=74
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-appli
cations/resnet/resnet50v2_weights_tf_dim_ordering_tf_kernels_notop.h5
94674944/94668760 [==============================] - 1s 0us/step
94683136/94668760 [==============================] - 1s 0us/step
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
model_1 (Functional)         (None, 100352)            23564800
_________________________________________________________________
dense_5 (Dense)              (None, 512)               51380736
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_6 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_7 (Dense)              (None, 20)                10260
=================================================================
Total params: 75,218,452
Trainable params: 51,653,652
Non-trainable params: 23,564,800
_________________________________________________________________
```

Result: Several optimizers have been tried, yet Resnet50V2 did not perform well. The model failed to "learn" on the given pretrained model and was underfitting. The accuracy did not improve across a threshold and therefore the model would not learn well on the given training dataset, as well as fail to generalize on the test set. The result on the training set was at an accuracy of 0.05, thus no valid solution.

2.2.3 MobileNetV2

MobileNetV2 is a CNN destined for usage on mobile devices. The basis is an inverted residual structure where the residual connections are between the bottleneck layers. The intermediate expansion layer uses lightweight depthwise convolutions to filter features as a source of non-linearity. MobileNetV2 contains two types of blocks: one residual block with stride 1 and one block with stride 2 for downsizing. The first layer is a 1×1 convolution with ReLU6, followed by a depthwise convolution. The third layer is another 1×1 convolution but without any non-linearity. The architecture of MobileNetV2 has the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers.

A key aspect of both MobileNets V1 and V2 is their use of depthwise separable convolutions, which significantly reduce the number of parameters compared to networks of the same depth but with regular convolutions. The networks advantage is its small size, low-latency, low-power model that can be broadly applicable for many use cases such as image classification and object detection, but runs exceptionally well on CPUs instead of costly and resource-intensive GPUs.

In [12]:

```
batch_size = 64
image_size = (256, 256)
image_shape= (image_size[0], image_size[1], 3)
train_generator_MobileNetV2 = train_data_augmentation(train_df, image_size

from tensorflow.keras import Model
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2

# We upload the MobileNetV2 model without the last layer (for the output)
base_MobileNetV2_model = MobileNetV2(weights="imagenet", include_top=False

# We create layers to merge with the MobileNetV2 model in order to give us
head_MobileNetV2_model = base_MobileNetV2_model.output
head_MobileNetV2_model = GlobalAveragePooling2D()(head_MobileNetV2_model)
head_MobileNetV2_model = Flatten()(head_MobileNetV2_model)
#head_MobileNetV2_model = Dense(32, activation="relu")(head_MobileNetV2_mo
#head_MobileNetV2_model = Dropout(0.5)(head_MobileNetV2_model)
head_MobileNetV2_model = Dense(20, activation="softmax")(head_MobileNetV2_

# We merge the base MobileNetV2 model with our newly created layers
merged_MobileNetV2_model = Model(inputs=base_MobileNetV2_model.input, outpu

# We freeze the pretrained layers of the base MobileNetV2 model
for layer in base_MobileNetV2_model.layers:
    layer.trainable = False

# We compile the whole merged model
merged_MobileNetV2_model.compile(optimizer='rmsprop', loss='categorical_cr

# Model summary
merged_MobileNetV2_model.summary()

# Code line below if we want to train this model

merged_MobileNetV2_model.fit(train_generator_MobileNetV2,epochs=20)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-appli
cations/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_22
4_no_top.h5
9412608/9406464 [==============================] - 0s 0us/step
9420800/9406464 [==============================] - 0s 0us/step
Model: "model_2"
__________________________________________________________________________
_____
Layer (type)                    Output Shape         Param #     Connected
to
==========================================================================
=====================
input_3 (InputLayer)            [(None, None, None,  0
__________________________________________________________________________
_____
Conv1 (Conv2D)                  (None, None, None, 3 864         input_3[0]
[0]
__________________________________________________________________________
_____
bn_Conv1 (BatchNormalization)   (None, None, None, 3 128         Conv1[0][0
]
```

```
__________________________________________________________________________________________________
_____________________
Conv1_relu (ReLU)               (None, None, None, 3 0           bn_Conv1[0
][0]
__________________________________________________________________________________________________
_____________________
expanded_conv_depthwise (Depthw (None, None, None, 3 288         Conv1_relu
[0][0]
__________________________________________________________________________________________________
_____________________
expanded_conv_depthwise_BN (Bat (None, None, None, 3 128         expanded_c
onv_depthwise[0][0]
__________________________________________________________________________________________________
_____________________
expanded_conv_depthwise_relu (R (None, None, None, 3 0           expanded_c
onv_depthwise_BN[0][0]
__________________________________________________________________________________________________
_____________________
expanded_conv_project (Conv2D)  (None, None, None, 1 512         expanded_c
onv_depthwise_relu[0][0
__________________________________________________________________________________________________
_____________________
expanded_conv_project_BN (Batch (None, None, None, 1 64          expanded_c
onv_project[0][0]
__________________________________________________________________________________________________
_____________________
block_1_expand (Conv2D)         (None, None, None, 9 1536        expanded_c
onv_project_BN[0][0]
__________________________________________________________________________________________________
_____________________
block_1_expand_BN (BatchNormali (None, None, None, 9 384         block_1_ex
pand[0][0]
__________________________________________________________________________________________________
_____________________
block_1_expand_relu (ReLU)      (None, None, None, 9 0           block_1_ex
pand_BN[0][0]
__________________________________________________________________________________________________
_____________________
block_1_pad (ZeroPadding2D)     (None, None, None, 9 0           block_1_ex
pand_relu[0][0]
__________________________________________________________________________________________________
_____________________
block_1_depthwise (DepthwiseCon (None, None, None, 9 864         block_1_pa
d[0][0]
__________________________________________________________________________________________________
_____________________
block_1_depthwise_BN (BatchNorm (None, None, None, 9 384         block_1_de
pthwise[0][0]
__________________________________________________________________________________________________
_____________________
block_1_depthwise_relu (ReLU)   (None, None, None, 9 0           block_1_de
pthwise_BN[0][0]
__________________________________________________________________________________________________
_____________________
block_1_project (Conv2D)        (None, None, None, 2 2304        block_1_de
pthwise_relu[0][0]
__________________________________________________________________________________________________
_____________________
block_1_project_BN (BatchNormal (None, None, None, 2 96          block_1_pr
oject[0][0]
__________________________________________________________________________________________________
```

```
__________________________
block_2_expand (Conv2D)         (None, None, None, 1 3456        block_1_pr
oject_BN[0][0]
__________________________________________________________________________
__________________________
block_2_expand_BN (BatchNormali (None, None, None, 1 576         block_2_ex
pand[0][0]
__________________________________________________________________________
__________________________
block_2_expand_relu (ReLU)      (None, None, None, 1 0           block_2_ex
pand_BN[0][0]
__________________________________________________________________________
__________________________
block_2_depthwise (DepthwiseCon (None, None, None, 1 1296        block_2_ex
pand_relu[0][0]
__________________________________________________________________________
__________________________
block_2_depthwise_BN (BatchNorm (None, None, None, 1 576         block_2_de
pthwise[0][0]
__________________________________________________________________________
__________________________
block_2_depthwise_relu (ReLU)   (None, None, None, 1 0           block_2_de
pthwise_BN[0][0]
__________________________________________________________________________
__________________________
block_2_project (Conv2D)        (None, None, None, 2 3456        block_2_de
pthwise_relu[0][0]
__________________________________________________________________________
__________________________
block_2_project_BN (BatchNormal (None, None, None, 2 96          block_2_pr
oject[0][0]
__________________________________________________________________________
__________________________
block_2_add (Add)               (None, None, None, 2 0           block_1_pr
oject_BN[0][0]
                                                                 block_2_pr

oject_BN[0][0]
__________________________________________________________________________
__________________________
block_3_expand (Conv2D)         (None, None, None, 1 3456        block_2_ad
d[0][0]
__________________________________________________________________________
__________________________
block_3_expand_BN (BatchNormali (None, None, None, 1 576         block_3_ex
pand[0][0]
__________________________________________________________________________
__________________________
block_3_expand_relu (ReLU)      (None, None, None, 1 0           block_3_ex
pand_BN[0][0]
__________________________________________________________________________
__________________________
block_3_pad (ZeroPadding2D)     (None, None, None, 1 0           block_3_ex
pand_relu[0][0]
__________________________________________________________________________
__________________________
block_3_depthwise (DepthwiseCon (None, None, None, 1 1296        block_3_pa
d[0][0]
__________________________________________________________________________
__________________________
block_3_depthwise_BN (BatchNorm (None, None, None, 1 576         block_3_de
pthwise[0][0]
```

```
__________________________________________________________________________________________
__________________
block_3_depthwise_relu (ReLU)   (None, None, None, 1 0           block_3_de
pthwise_BN[0][0]
__________________________________________________________________________________________
__________________
block_3_project (Conv2D)        (None, None, None, 3 4608        block_3_de
pthwise_relu[0][0]
__________________________________________________________________________________________
__________________
block_3_project_BN (BatchNormal (None, None, None, 3 128         block_3_pr
oject[0][0]
__________________________________________________________________________________________
__________________
block_4_expand (Conv2D)         (None, None, None, 1 6144        block_3_pr
oject_BN[0][0]
__________________________________________________________________________________________
__________________
block_4_expand_BN (BatchNormali (None, None, None, 1 768         block_4_ex
pand[0][0]
__________________________________________________________________________________________
__________________
block_4_expand_relu (ReLU)      (None, None, None, 1 0           block_4_ex
pand_BN[0][0]
__________________________________________________________________________________________
__________________
block_4_depthwise (DepthwiseCon (None, None, None, 1 1728        block_4_ex
pand_relu[0][0]
__________________________________________________________________________________________
__________________
block_4_depthwise_BN (BatchNorm (None, None, None, 1 768         block_4_de
pthwise[0][0]
__________________________________________________________________________________________
__________________
block_4_depthwise_relu (ReLU)   (None, None, None, 1 0           block_4_de
pthwise_BN[0][0]
__________________________________________________________________________________________
__________________
block_4_project (Conv2D)        (None, None, None, 3 6144        block_4_de
pthwise_relu[0][0]
__________________________________________________________________________________________
__________________
block_4_project_BN (BatchNormal (None, None, None, 3 128         block_4_pr
oject[0][0]
__________________________________________________________________________________________
__________________
block_4_add (Add)               (None, None, None, 3 0           block_3_pr
oject_BN[0][0]
                                                                 block_4_pr
oject_BN[0][0]
__________________________________________________________________________________________
__________________
block_5_expand (Conv2D)         (None, None, None, 1 6144        block_4_ad
d[0][0]
__________________________________________________________________________________________
__________________
block_5_expand_BN (BatchNormali (None, None, None, 1 768         block_5_ex
pand[0][0]
__________________________________________________________________________________________
__________________
block_5_expand_relu (ReLU)      (None, None, None, 1 0           block_5_ex
```

```
pand_BN[0][0]
__________________________________________________________________________________________
_____________________________
block_5_depthwise (DepthwiseCon (None, None, None, 1 1728        block_5_ex
pand_relu[0][0]
__________________________________________________________________________________________
_____________________________
block_5_depthwise_BN (BatchNorm (None, None, None, 1 768         block_5_de
pthwise[0][0]
__________________________________________________________________________________________
_____________________________
block_5_depthwise_relu (ReLU)   (None, None, None, 1 0           block_5_de
pthwise_BN[0][0]
__________________________________________________________________________________________
_____________________________
block_5_project (Conv2D)        (None, None, None, 3 6144        block_5_de
pthwise_relu[0][0]
__________________________________________________________________________________________
_____________________________
block_5_project_BN (BatchNormal (None, None, None, 3 128         block_5_pr
oject[0][0]
__________________________________________________________________________________________
_____________________________
block_5_add (Add)               (None, None, None, 3 0           block_4_ad
d[0][0]
                                                                 block_5_pr
oject_BN[0][0]
__________________________________________________________________________________________
_____________________________
block_6_expand (Conv2D)         (None, None, None, 1 6144        block_5_ad
d[0][0]
__________________________________________________________________________________________
_____________________________
block_6_expand_BN (BatchNormali (None, None, None, 1 768         block_6_ex
pand[0][0]
__________________________________________________________________________________________
_____________________________
block_6_expand_relu (ReLU)      (None, None, None, 1 0           block_6_ex
pand_BN[0][0]
__________________________________________________________________________________________
_____________________________
block_6_pad (ZeroPadding2D)     (None, None, None, 1 0           block_6_ex
pand_relu[0][0]
__________________________________________________________________________________________
_____________________________
block_6_depthwise (DepthwiseCon (None, None, None, 1 1728        block_6_pa
d[0][0]
__________________________________________________________________________________________
_____________________________
block_6_depthwise_BN (BatchNorm (None, None, None, 1 768         block_6_de
pthwise[0][0]
__________________________________________________________________________________________
_____________________________
block_6_depthwise_relu (ReLU)   (None, None, None, 1 0           block_6_de
pthwise_BN[0][0]
__________________________________________________________________________________________
_____________________________
block_6_project (Conv2D)        (None, None, None, 6 12288       block_6_de
pthwise_relu[0][0]
__________________________________________________________________________________________
_____________________________
```

```
block_6_project_BN (BatchNormal (None, None, None, 6 256         block_6_pr
oject[0][0]
__________________________________________________________________________
_____________________
block_7_expand (Conv2D)         (None, None, None, 3 24576       block_6_pr
oject_BN[0][0]
__________________________________________________________________________
_____________________
block_7_expand_BN (BatchNormali (None, None, None, 3 1536        block_7_ex
pand[0][0]
__________________________________________________________________________
_____________________
block_7_expand_relu (ReLU)      (None, None, None, 3 0           block_7_ex
pand_BN[0][0]
__________________________________________________________________________
_____________________
block_7_depthwise (DepthwiseCon (None, None, None, 3 3456        block_7_ex
pand_relu[0][0]
__________________________________________________________________________
_____________________
block_7_depthwise_BN (BatchNorm (None, None, None, 3 1536        block_7_de
pthwise[0][0]
__________________________________________________________________________
_____________________
block_7_depthwise_relu (ReLU)   (None, None, None, 3 0           block_7_de
pthwise_BN[0][0]
__________________________________________________________________________
_____________________
block_7_project (Conv2D)        (None, None, None, 6 24576       block_7_de
pthwise_relu[0][0]
__________________________________________________________________________
_____________________
block_7_project_BN (BatchNormal (None, None, None, 6 256         block_7_pr
oject[0][0]
__________________________________________________________________________
_____________________
block_7_add (Add)               (None, None, None, 6 0           block_6_pr
oject_BN[0][0]
                                                                 block_7_pr
oject_BN[0][0]
__________________________________________________________________________
_____________________
block_8_expand (Conv2D)         (None, None, None, 3 24576       block_7_ad
d[0][0]
__________________________________________________________________________
_____________________
block_8_expand_BN (BatchNormali (None, None, None, 3 1536        block_8_ex
pand[0][0]
__________________________________________________________________________
_____________________
block_8_expand_relu (ReLU)      (None, None, None, 3 0           block_8_ex
pand_BN[0][0]
__________________________________________________________________________
_____________________
block_8_depthwise (DepthwiseCon (None, None, None, 3 3456        block_8_ex
pand_relu[0][0]
__________________________________________________________________________
_____________________
block_8_depthwise_BN (BatchNorm (None, None, None, 3 1536        block_8_de
pthwise[0][0]
__________________________________________________________________________
```

```
__________________________
block_8_depthwise_relu (ReLU)   (None, None, None, 3 0           block_8_de
pthwise_BN[0][0]
__________________________________________________________________________
__________________________
block_8_project (Conv2D)        (None, None, None, 6 24576       block_8_de
pthwise_relu[0][0]
__________________________________________________________________________
__________________________
block_8_project_BN (BatchNormal (None, None, None, 6 256         block_8_pr
oject[0][0]
__________________________________________________________________________
__________________________
block_8_add (Add)               (None, None, None, 6 0           block_7_ad
d[0][0]
                                                                 block_8_pr
oject_BN[0][0]
__________________________________________________________________________
__________________________
block_9_expand (Conv2D)         (None, None, None, 3 24576       block_8_ad
d[0][0]
__________________________________________________________________________
__________________________
block_9_expand_BN (BatchNormali (None, None, None, 3 1536        block_9_ex
pand[0][0]
__________________________________________________________________________
__________________________
block_9_expand_relu (ReLU)      (None, None, None, 3 0           block_9_ex
pand_BN[0][0]
__________________________________________________________________________
__________________________
block_9_depthwise (DepthwiseCon (None, None, None, 3 3456        block_9_ex
pand_relu[0][0]
__________________________________________________________________________
__________________________
block_9_depthwise_BN (BatchNorm (None, None, None, 3 1536        block_9_de
pthwise[0][0]
__________________________________________________________________________
__________________________
block_9_depthwise_relu (ReLU)   (None, None, None, 3 0           block_9_de
pthwise_BN[0][0]
__________________________________________________________________________
__________________________
block_9_project (Conv2D)        (None, None, None, 6 24576       block_9_de
pthwise_relu[0][0]
__________________________________________________________________________
__________________________
block_9_project_BN (BatchNormal (None, None, None, 6 256         block_9_pr
oject[0][0]
__________________________________________________________________________
__________________________
block_9_add (Add)               (None, None, None, 6 0           block_8_ad
d[0][0]
                                                                 block_9_pr
oject_BN[0][0]
__________________________________________________________________________
__________________________
block_10_expand (Conv2D)        (None, None, None, 3 24576       block_9_ad
d[0][0]
__________________________________________________________________________
__________________________
```

```
block_10_expand_BN (BatchNormal (None, None, None, 3 1536        block_10_e
xpand[0][0]
__________________________________________________________________________
_____________________
block_10_expand_relu (ReLU)     (None, None, None, 3 0           block_10_e
xpand_BN[0][0]
__________________________________________________________________________
_____________________
block_10_depthwise (DepthwiseCo (None, None, None, 3 3456        block_10_e
xpand_relu[0][0]
__________________________________________________________________________
_____________________
block_10_depthwise_BN (BatchNor (None, None, None, 3 1536        block_10_d
epthwise[0][0]
__________________________________________________________________________
_____________________
block_10_depthwise_relu (ReLU)  (None, None, None, 3 0           block_10_d
epthwise_BN[0][0]
__________________________________________________________________________
_____________________
block_10_project (Conv2D)       (None, None, None, 9 36864       block_10_d
epthwise_relu[0][0]
__________________________________________________________________________
_____________________
block_10_project_BN (BatchNorma (None, None, None, 9 384         block_10_p
roject[0][0]
__________________________________________________________________________
_____________________
block_11_expand (Conv2D)        (None, None, None, 5 55296       block_10_p
roject_BN[0][0]
__________________________________________________________________________
_____________________
block_11_expand_BN (BatchNormal (None, None, None, 5 2304        block_11_e
xpand[0][0]
__________________________________________________________________________
_____________________
block_11_expand_relu (ReLU)     (None, None, None, 5 0           block_11_e
xpand_BN[0][0]
__________________________________________________________________________
_____________________
block_11_depthwise (DepthwiseCo (None, None, None, 5 5184        block_11_e
xpand_relu[0][0]
__________________________________________________________________________
_____________________
block_11_depthwise_BN (BatchNor (None, None, None, 5 2304        block_11_d
epthwise[0][0]
__________________________________________________________________________
_____________________
block_11_depthwise_relu (ReLU)  (None, None, None, 5 0           block_11_d
epthwise_BN[0][0]
__________________________________________________________________________
_____________________
block_11_project (Conv2D)       (None, None, None, 9 55296       block_11_d
epthwise_relu[0][0]
__________________________________________________________________________
_____________________
block_11_project_BN (BatchNorma (None, None, None, 9 384         block_11_p
roject[0][0]
__________________________________________________________________________
_____________________
block_11_add (Add)              (None, None, None, 9 0           block_10_p
```

```
roject_BN[0][0]
                                                                    block_11_p
roject_BN[0][0]
__________________________________________________________________________________________
__________________________
block_12_expand (Conv2D)        (None, None, None, 5 55296       block_11_a
dd[0][0]
__________________________________________________________________________________________
__________________________
block_12_expand_BN (BatchNormal (None, None, None, 5 2304        block_12_e
xpand[0][0]
__________________________________________________________________________________________
__________________________
block_12_expand_relu (ReLU)     (None, None, None, 5 0           block_12_e
xpand_BN[0][0]
__________________________________________________________________________________________
__________________________
block_12_depthwise (DepthwiseCo (None, None, None, 5 5184        block_12_e
xpand_relu[0][0]
__________________________________________________________________________________________
__________________________
block_12_depthwise_BN (BatchNor (None, None, None, 5 2304        block_12_d
epthwise[0][0]
__________________________________________________________________________________________
__________________________
block_12_depthwise_relu (ReLU)  (None, None, None, 5 0           block_12_d
epthwise_BN[0][0]
__________________________________________________________________________________________
__________________________
block_12_project (Conv2D)       (None, None, None, 9 55296       block_12_d
epthwise_relu[0][0]
__________________________________________________________________________________________
__________________________
block_12_project_BN (BatchNorma (None, None, None, 9 384         block_12_p
roject[0][0]
__________________________________________________________________________________________
__________________________
block_12_add (Add)              (None, None, None, 9 0           block_11_a
dd[0][0]
                                                                    block_12_p
roject_BN[0][0]
__________________________________________________________________________________________
__________________________
block_13_expand (Conv2D)        (None, None, None, 5 55296       block_12_a
dd[0][0]
__________________________________________________________________________________________
__________________________
block_13_expand_BN (BatchNormal (None, None, None, 5 2304        block_13_e
xpand[0][0]
__________________________________________________________________________________________
__________________________
block_13_expand_relu (ReLU)     (None, None, None, 5 0           block_13_e
xpand_BN[0][0]
__________________________________________________________________________________________
__________________________
block_13_pad (ZeroPadding2D)    (None, None, None, 5 0           block_13_e
xpand_relu[0][0]
__________________________________________________________________________________________
__________________________
block_13_depthwise (DepthwiseCo (None, None, None, 5 5184        block_13_p
ad[0][0]
```

```
__________________________________________________________________________________________________
__________________________
block_13_depthwise_BN (BatchNor (None, None, None, 5 2304        block_13_d
epthwise[0][0]
__________________________________________________________________________________________________
__________________________
block_13_depthwise_relu (ReLU)  (None, None, None, 5 0           block_13_d
epthwise_BN[0][0]
__________________________________________________________________________________________________
__________________________
block_13_project (Conv2D)       (None, None, None, 1 92160       block_13_d
epthwise_relu[0][0]
__________________________________________________________________________________________________
__________________________
block_13_project_BN (BatchNorma (None, None, None, 1 640         block_13_p
roject[0][0]
__________________________________________________________________________________________________
__________________________
block_14_expand (Conv2D)        (None, None, None, 9 153600      block_13_p
roject_BN[0][0]
__________________________________________________________________________________________________
__________________________
block_14_expand_BN (BatchNormal (None, None, None, 9 3840        block_14_e
xpand[0][0]
__________________________________________________________________________________________________
__________________________
block_14_expand_relu (ReLU)     (None, None, None, 9 0           block_14_e
xpand_BN[0][0]
__________________________________________________________________________________________________
__________________________
block_14_depthwise (DepthwiseCo (None, None, None, 9 8640        block_14_e
xpand_relu[0][0]
__________________________________________________________________________________________________
__________________________
block_14_depthwise_BN (BatchNor (None, None, None, 9 3840        block_14_d
epthwise[0][0]
__________________________________________________________________________________________________
__________________________
block_14_depthwise_relu (ReLU)  (None, None, None, 9 0           block_14_d
epthwise_BN[0][0]
__________________________________________________________________________________________________
__________________________
block_14_project (Conv2D)       (None, None, None, 1 153600      block_14_d
epthwise_relu[0][0]
__________________________________________________________________________________________________
__________________________
block_14_project_BN (BatchNorma (None, None, None, 1 640         block_14_p
roject[0][0]
__________________________________________________________________________________________________
__________________________
block_14_add (Add)              (None, None, None, 1 0           block_13_p
roject_BN[0][0]
                                                                 block_14_p
roject_BN[0][0]
__________________________________________________________________________________________________
__________________________
block_15_expand (Conv2D)        (None, None, None, 9 153600      block_14_a
dd[0][0]
__________________________________________________________________________________________________
__________________________
block_15_expand_BN (BatchNormal (None, None, None, 9 3840        block_15_e
```

```
xpand[0][0]
__________________________________________________________________________________________
__________________________
block_15_expand_relu (ReLU)     (None, None, None, 9 0           block_15_e
xpand_BN[0][0]
__________________________________________________________________________________________
__________________________
block_15_depthwise (DepthwiseCo (None, None, None, 9 8640        block_15_e
xpand_relu[0][0]
__________________________________________________________________________________________
__________________________
block_15_depthwise_BN (BatchNor (None, None, None, 9 3840        block_15_d
epthwise[0][0]
__________________________________________________________________________________________
__________________________
block_15_depthwise_relu (ReLU)  (None, None, None, 9 0           block_15_d
epthwise_BN[0][0]
__________________________________________________________________________________________
__________________________
block_15_project (Conv2D)       (None, None, None, 1 153600      block_15_d
epthwise_relu[0][0]
__________________________________________________________________________________________
__________________________
block_15_project_BN (BatchNorma (None, None, None, 1 640         block_15_p
roject[0][0]
__________________________________________________________________________________________
__________________________
block_15_add (Add)              (None, None, None, 1 0           block_14_a
dd[0][0]
                                                                 block_15_p
roject_BN[0][0]
__________________________________________________________________________________________
__________________________
block_16_expand (Conv2D)        (None, None, None, 9 153600      block_15_a
dd[0][0]
__________________________________________________________________________________________
__________________________
block_16_expand_BN (BatchNormal (None, None, None, 9 3840        block_16_e
xpand[0][0]
__________________________________________________________________________________________
__________________________
block_16_expand_relu (ReLU)     (None, None, None, 9 0           block_16_e
xpand_BN[0][0]
__________________________________________________________________________________________
__________________________
block_16_depthwise (DepthwiseCo (None, None, None, 9 8640        block_16_e
xpand_relu[0][0]
__________________________________________________________________________________________
__________________________
block_16_depthwise_BN (BatchNor (None, None, None, 9 3840        block_16_d
epthwise[0][0]
__________________________________________________________________________________________
__________________________
block_16_depthwise_relu (ReLU)  (None, None, None, 9 0           block_16_d
epthwise_BN[0][0]
__________________________________________________________________________________________
__________________________
block_16_project (Conv2D)       (None, None, None, 3 307200      block_16_d
epthwise_relu[0][0]
__________________________________________________________________________________________
__________________________
```

```
block_16_project_BN (BatchNorma (None, None, None, 3 1280        block_16_p
roject[0][0]
__________________________________________________________________________
_____________________
Conv_1 (Conv2D)                 (None, None, None, 1 409600      block_16_p
roject_BN[0][0]
__________________________________________________________________________
_____________________
Conv_1_bn (BatchNormalization)  (None, None, None, 1 5120        Conv_1[0][
0]
__________________________________________________________________________
_____________________
out_relu (ReLU)                 (None, None, None, 1 0           Conv_1_bn[
0][0]
__________________________________________________________________________
_____________________
global_average_pooling2d (Globa (None, 1280)         0           out_relu[0
][0]
__________________________________________________________________________
_____________________
flatten_1 (Flatten)             (None, 1280)         0           global_ave
rage_pooling2d[0][0]
__________________________________________________________________________
_____________________
dense_8 (Dense)                 (None, 20)           25620       flatten_1[
0][0]
==========================================================================
======================
Total params: 2,283,604
Trainable params: 25,620
Non-trainable params: 2,257,984
__________________________________________________________________________
_____________________
Epoch 1/20
12/12 [==============================] - 12s 819ms/step - loss: 3.9209 - bi
nary_accuracy: 0.9287
Epoch 2/20
12/12 [==============================] - 11s 882ms/step - loss: 3.0205 - bi
nary_accuracy: 0.9309
Epoch 3/20
12/12 [==============================] - 12s 1s/step - loss: 2.6172 - binar
y_accuracy: 0.9354
Epoch 4/20
12/12 [==============================] - 10s 825ms/step - loss: 2.3686 - bi
nary_accuracy: 0.9400
Epoch 5/20
12/12 [==============================] - 10s 870ms/step - loss: 2.1647 - bi
nary_accuracy: 0.9454
Epoch 6/20
12/12 [==============================] - 12s 997ms/step - loss: 2.0776 - bi
nary_accuracy: 0.9495
Epoch 7/20
12/12 [==============================] - 10s 811ms/step - loss: 1.9811 - bi
nary_accuracy: 0.9516
Epoch 8/20
12/12 [==============================] - 10s 858ms/step - loss: 1.9388 - bi
nary_accuracy: 0.9545
Epoch 9/20
12/12 [==============================] - 12s 998ms/step - loss: 1.8808 - bi
nary_accuracy: 0.9573
Epoch 10/20
```

```
12/12 [==============================] - 10s 789ms/step - loss: 1.8461 - bi
nary_accuracy: 0.9575
Epoch 11/20
12/12 [==============================] - 10s 860ms/step - loss: 1.8298 - bi
nary_accuracy: 0.9585
Epoch 12/20
12/12 [==============================] - 10s 820ms/step - loss: 1.8375 - bi
nary_accuracy: 0.9599
Epoch 13/20
12/12 [==============================] - 12s 939ms/step - loss: 1.7994 - bi
nary_accuracy: 0.9599
Epoch 14/20
12/12 [==============================] - 10s 875ms/step - loss: 1.7710 - bi
nary_accuracy: 0.9623
Epoch 15/20
12/12 [==============================] - 10s 817ms/step - loss: 1.8097 - bi
nary_accuracy: 0.9630
Epoch 16/20
12/12 [==============================] - 12s 995ms/step - loss: 1.7437 - bi
nary_accuracy: 0.9645
Epoch 17/20
12/12 [==============================] - 10s 863ms/step - loss: 1.7841 - bi
nary_accuracy: 0.9637
Epoch 18/20
12/12 [==============================] - 12s 1s/step - loss: 1.7371 - binar
y_accuracy: 0.9644
Epoch 19/20
12/12 [==============================] - 10s 811ms/step - loss: 1.8213 - bi
nary_accuracy: 0.9640
Epoch 20/20
12/12 [==============================] - 11s 827ms/step - loss: 1.7753 - bi
nary_accuracy: 0.9666
```

Out[12]: `<keras.callbacks.History at 0x7f1a902b53d0>`

Result: We can see here that the model learns somewhat. The accuracy increases yet the loss does not decrease within 10 epochs. Generalization of the MobileNet on the test set of Pascal Voc produced the best performance across the models.

2.2.4 Fine Tuning

One last step is fine-tuning, which is means unfreezing the entire model or parts of the model obtained and re-training it on the dataset with a small learning rate. The goal is to achieve improvements, by incrementally adapting the pretrained features to the new data.

After the initial training, we tried to further improve the performamce of our transfer learning model by unfreezing the layers of the pretrained model and perform another training session (see the code below). In doing so, the weights in the lastlayers of the pretrained model (that are unfrozen) are updated given the new data. Usually, iterative unfreezing starting at the last layer is performed. However, this did not lead to any increased accuracy.

As reported, MobileNetV2 produced best results on the test set so far - therefore the fine tuning for better accuracy results will be done with MobileNetV2 as a base.

In [13]:
```
len_model = len(base_MobileNetV2_model.layers)
print("Number of layers in the base model: ", len_model)
```

```
Number of layers in the base model:  154
```

In [14]:
```
from tensorflow.keras.optimizers import SGD, RMSprop

# first set: base_MobileNetV2_model.layers.trainable = True

# Fine-tune from this layer onwards (usually start at the last layer minus
fine_tune_at = len_model-2

# We unfreeze the layers
for layer in base_MobileNetV2_model.layers[:fine_tune_at]:
  layer.trainable = False

merged_model = merged_MobileNetV2_model

merged_model.compile(optimizer=RMSprop(learning_rate=0.00001, momentum=0.9

# Model summary
merged_model.summary()

#Code line below if we want to retrain this model

merged_model.fit(train_generator_MobileNetV2, epochs=5)
```

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
Connected to
==================================================================
==========
input_3 (InputLayer)         [(None, None, None,  0
_________________________________________________________________
```

```
Conv1 (Conv2D)                  (None, None, None, 3 864         input_3[0]
[0]
__________________________________________________________________________
_____________________
bn_Conv1 (BatchNormalization)   (None, None, None, 3 128         Conv1[0][0
]
__________________________________________________________________________
_____________________
Conv1_relu (ReLU)               (None, None, None, 3 0           bn_Conv1[0
][0]
__________________________________________________________________________
_____________________
expanded_conv_depthwise (Depthw (None, None, None, 3 288         Conv1_relu
[0][0]
__________________________________________________________________________
_____________________
expanded_conv_depthwise_BN (Bat (None, None, None, 3 128         expanded_c
onv_depthwise[0][0]
__________________________________________________________________________
_____________________
expanded_conv_depthwise_relu (R (None, None, None, 3 0           expanded_c
onv_depthwise_BN[0][0]
__________________________________________________________________________
_____________________
expanded_conv_project (Conv2D)  (None, None, None, 1 512         expanded_c
onv_depthwise_relu[0][0
__________________________________________________________________________
_____________________
expanded_conv_project_BN (Batch (None, None, None, 1 64          expanded_c
onv_project[0][0]
__________________________________________________________________________
_____________________
block_1_expand (Conv2D)         (None, None, None, 9 1536        expanded_c
onv_project_BN[0][0]
__________________________________________________________________________
_____________________
block_1_expand_BN (BatchNormali (None, None, None, 9 384         block_1_ex
pand[0][0]
__________________________________________________________________________
_____________________
block_1_expand_relu (ReLU)      (None, None, None, 9 0           block_1_ex
pand_BN[0][0]
__________________________________________________________________________
_____________________
block_1_pad (ZeroPadding2D)     (None, None, None, 9 0           block_1_ex
pand_relu[0][0]
__________________________________________________________________________
_____________________
block_1_depthwise (DepthwiseCon (None, None, None, 9 864         block_1_pa
d[0][0]
__________________________________________________________________________
_____________________
block_1_depthwise_BN (BatchNorm (None, None, None, 9 384         block_1_de
pthwise[0][0]
__________________________________________________________________________
_____________________
block_1_depthwise_relu (ReLU)   (None, None, None, 9 0           block_1_de
pthwise_BN[0][0]
__________________________________________________________________________
_____________________
block_1_project (Conv2D)        (None, None, None, 2 2304        block_1_de
```

```
pthwise_relu[0][0]
__________________________________________________________________________________________
_____________________________
block_1_project_BN (BatchNormal (None, None, None, 2 96          block_1_pr
oject[0][0]
__________________________________________________________________________________________
_____________________________
block_2_expand (Conv2D)         (None, None, None, 1 3456        block_1_pr
oject_BN[0][0]
__________________________________________________________________________________________
_____________________________
block_2_expand_BN (BatchNormali (None, None, None, 1 576         block_2_ex
pand[0][0]
__________________________________________________________________________________________
_____________________________
block_2_expand_relu (ReLU)      (None, None, None, 1 0           block_2_ex
pand_BN[0][0]
__________________________________________________________________________________________
_____________________________
block_2_depthwise (DepthwiseCon (None, None, None, 1 1296        block_2_ex
pand_relu[0][0]
__________________________________________________________________________________________
_____________________________
block_2_depthwise_BN (BatchNorm (None, None, None, 1 576         block_2_de
pthwise[0][0]
__________________________________________________________________________________________
_____________________________
block_2_depthwise_relu (ReLU)   (None, None, None, 1 0           block_2_de
pthwise_BN[0][0]
__________________________________________________________________________________________
_____________________________
block_2_project (Conv2D)        (None, None, None, 2 3456        block_2_de
pthwise_relu[0][0]
__________________________________________________________________________________________
_____________________________
block_2_project_BN (BatchNormal (None, None, None, 2 96          block_2_pr
oject[0][0]
__________________________________________________________________________________________
_____________________________
block_2_add (Add)               (None, None, None, 2 0           block_1_pr
oject_BN[0][0]
                                                                 block_2_pr
oject_BN[0][0]
__________________________________________________________________________________________
_____________________________
block_3_expand (Conv2D)         (None, None, None, 1 3456        block_2_ad
d[0][0]
__________________________________________________________________________________________
_____________________________
block_3_expand_BN (BatchNormali (None, None, None, 1 576         block_3_ex
pand[0][0]
__________________________________________________________________________________________
_____________________________
block_3_expand_relu (ReLU)      (None, None, None, 1 0           block_3_ex
pand_BN[0][0]
__________________________________________________________________________________________
_____________________________
block_3_pad (ZeroPadding2D)     (None, None, None, 1 0           block_3_ex
pand_relu[0][0]
__________________________________________________________________________________________
_____________________________
```

```
block_3_depthwise (DepthwiseCon (None, None, None, 1 1296        block_3_pa
d[0][0]
__________________________________________________________________________
_____________________
block_3_depthwise_BN (BatchNorm (None, None, None, 1 576         block_3_de
pthwise[0][0]
__________________________________________________________________________
_____________________
block_3_depthwise_relu (ReLU)   (None, None, None, 1 0           block_3_de
pthwise_BN[0][0]
__________________________________________________________________________
_____________________
block_3_project (Conv2D)        (None, None, None, 3 4608        block_3_de
pthwise_relu[0][0]
__________________________________________________________________________
_____________________
block_3_project_BN (BatchNormal (None, None, None, 3 128         block_3_pr
oject[0][0]
__________________________________________________________________________
_____________________
block_4_expand (Conv2D)         (None, None, None, 1 6144        block_3_pr
oject_BN[0][0]
__________________________________________________________________________
_____________________
block_4_expand_BN (BatchNormali (None, None, None, 1 768         block_4_ex
pand[0][0]
__________________________________________________________________________
_____________________
block_4_expand_relu (ReLU)      (None, None, None, 1 0           block_4_ex
pand_BN[0][0]
__________________________________________________________________________
_____________________
block_4_depthwise (DepthwiseCon (None, None, None, 1 1728        block_4_ex
pand_relu[0][0]
__________________________________________________________________________
_____________________
block_4_depthwise_BN (BatchNorm (None, None, None, 1 768         block_4_de
pthwise[0][0]
__________________________________________________________________________
_____________________
block_4_depthwise_relu (ReLU)   (None, None, None, 1 0           block_4_de
pthwise_BN[0][0]
__________________________________________________________________________
_____________________
block_4_project (Conv2D)        (None, None, None, 3 6144        block_4_de
pthwise_relu[0][0]
__________________________________________________________________________
_____________________
block_4_project_BN (BatchNormal (None, None, None, 3 128         block_4_pr
oject[0][0]
__________________________________________________________________________
_____________________
block_4_add (Add)               (None, None, None, 3 0           block_3_pr
oject_BN[0][0]
                                                                 block_4_pr
oject_BN[0][0]
__________________________________________________________________________
_____________________
block_5_expand (Conv2D)         (None, None, None, 1 6144        block_4_ad
d[0][0]
__________________________________________________________________________
```

```
__________________________
block_5_expand_BN (BatchNormali (None, None, None, 1 768         block_5_ex
pand[0][0]
__________________________________________________________________________
__________________________
block_5_expand_relu (ReLU)      (None, None, None, 1 0           block_5_ex
pand_BN[0][0]
__________________________________________________________________________
__________________________
block_5_depthwise (DepthwiseCon (None, None, None, 1 1728        block_5_ex
pand_relu[0][0]
__________________________________________________________________________
__________________________
block_5_depthwise_BN (BatchNorm (None, None, None, 1 768         block_5_de
pthwise[0][0]
__________________________________________________________________________
__________________________
block_5_depthwise_relu (ReLU)   (None, None, None, 1 0           block_5_de
pthwise_BN[0][0]
__________________________________________________________________________
__________________________
block_5_project (Conv2D)        (None, None, None, 3 6144        block_5_de
pthwise_relu[0][0]
__________________________________________________________________________
__________________________
block_5_project_BN (BatchNormal (None, None, None, 3 128         block_5_pr
oject[0][0]
__________________________________________________________________________
__________________________
block_5_add (Add)               (None, None, None, 3 0           block_4_ad
d[0][0]
                                                                 block_5_pr
oject_BN[0][0]
__________________________________________________________________________
__________________________
block_6_expand (Conv2D)         (None, None, None, 1 6144        block_5_ad
d[0][0]
__________________________________________________________________________
__________________________
block_6_expand_BN (BatchNormali (None, None, None, 1 768         block_6_ex
pand[0][0]
__________________________________________________________________________
__________________________
block_6_expand_relu (ReLU)      (None, None, None, 1 0           block_6_ex
pand_BN[0][0]
__________________________________________________________________________
__________________________
block_6_pad (ZeroPadding2D)     (None, None, None, 1 0           block_6_ex
pand_relu[0][0]
__________________________________________________________________________
__________________________
block_6_depthwise (DepthwiseCon (None, None, None, 1 1728        block_6_pa
d[0][0]
__________________________________________________________________________
__________________________
block_6_depthwise_BN (BatchNorm (None, None, None, 1 768         block_6_de
pthwise[0][0]
__________________________________________________________________________
__________________________
block_6_depthwise_relu (ReLU)   (None, None, None, 1 0           block_6_de
pthwise_BN[0][0]
```

```
__________________________________________________________________________________________________
__________________________
block_6_project (Conv2D)        (None, None, None, 6 12288       block_6_de
pthwise_relu[0][0]
__________________________________________________________________________________________________
__________________________
block_6_project_BN (BatchNormal (None, None, None, 6 256         block_6_pr
oject[0][0]
__________________________________________________________________________________________________
__________________________
block_7_expand (Conv2D)         (None, None, None, 3 24576       block_6_pr
oject_BN[0][0]
__________________________________________________________________________________________________
__________________________
block_7_expand_BN (BatchNormali (None, None, None, 3 1536        block_7_ex
pand[0][0]
__________________________________________________________________________________________________
__________________________
block_7_expand_relu (ReLU)      (None, None, None, 3 0           block_7_ex
pand_BN[0][0]
__________________________________________________________________________________________________
__________________________
block_7_depthwise (DepthwiseCon (None, None, None, 3 3456        block_7_ex
pand_relu[0][0]
__________________________________________________________________________________________________
__________________________
block_7_depthwise_BN (BatchNorm (None, None, None, 3 1536        block_7_de
pthwise[0][0]
__________________________________________________________________________________________________
__________________________
block_7_depthwise_relu (ReLU)   (None, None, None, 3 0           block_7_de
pthwise_BN[0][0]
__________________________________________________________________________________________________
__________________________
block_7_project (Conv2D)        (None, None, None, 6 24576       block_7_de
pthwise_relu[0][0]
__________________________________________________________________________________________________
__________________________
block_7_project_BN (BatchNormal (None, None, None, 6 256         block_7_pr
oject[0][0]
__________________________________________________________________________________________________
__________________________
block_7_add (Add)               (None, None, None, 6 0           block_6_pr
oject_BN[0][0]
                                                                 block_7_pr

oject_BN[0][0]
__________________________________________________________________________________________________
__________________________
block_8_expand (Conv2D)         (None, None, None, 3 24576       block_7_ad
d[0][0]
__________________________________________________________________________________________________
__________________________
block_8_expand_BN (BatchNormali (None, None, None, 3 1536        block_8_ex
pand[0][0]
__________________________________________________________________________________________________
__________________________
block_8_expand_relu (ReLU)      (None, None, None, 3 0           block_8_ex
pand_BN[0][0]
__________________________________________________________________________________________________
__________________________
block_8_depthwise (DepthwiseCon (None, None, None, 3 3456        block_8_ex
```

```
pand_relu[0][0]
__________________________________________________________________________________________________
__________________________
block_8_depthwise_BN (BatchNorm (None, None, None, 3 1536        block_8_de
pthwise[0][0]
__________________________________________________________________________________________________
__________________________
block_8_depthwise_relu (ReLU)   (None, None, None, 3 0           block_8_de
pthwise_BN[0][0]
__________________________________________________________________________________________________
__________________________
block_8_project (Conv2D)        (None, None, None, 6 24576       block_8_de
pthwise_relu[0][0]
__________________________________________________________________________________________________
__________________________
block_8_project_BN (BatchNormal (None, None, None, 6 256         block_8_pr
oject[0][0]
__________________________________________________________________________________________________
__________________________
block_8_add (Add)               (None, None, None, 6 0           block_7_ad
d[0][0]
                                                                 block_8_pr
oject_BN[0][0]
__________________________________________________________________________________________________
__________________________
block_9_expand (Conv2D)         (None, None, None, 3 24576       block_8_ad
d[0][0]
__________________________________________________________________________________________________
__________________________
block_9_expand_BN (BatchNormali (None, None, None, 3 1536        block_9_ex
pand[0][0]
__________________________________________________________________________________________________
__________________________
block_9_expand_relu (ReLU)      (None, None, None, 3 0           block_9_ex
pand_BN[0][0]
__________________________________________________________________________________________________
__________________________
block_9_depthwise (DepthwiseCon (None, None, None, 3 3456        block_9_ex
pand_relu[0][0]
__________________________________________________________________________________________________
__________________________
block_9_depthwise_BN (BatchNorm (None, None, None, 3 1536        block_9_de
pthwise[0][0]
__________________________________________________________________________________________________
__________________________
block_9_depthwise_relu (ReLU)   (None, None, None, 3 0           block_9_de
pthwise_BN[0][0]
__________________________________________________________________________________________________
__________________________
block_9_project (Conv2D)        (None, None, None, 6 24576       block_9_de
pthwise_relu[0][0]
__________________________________________________________________________________________________
__________________________
block_9_project_BN (BatchNormal (None, None, None, 6 256         block_9_pr
oject[0][0]
__________________________________________________________________________________________________
__________________________
block_9_add (Add)               (None, None, None, 6 0           block_8_ad
d[0][0]
                                                                 block_9_pr
oject_BN[0][0]
```

```
____________________________________________________________________________________________
____________________
block_10_expand (Conv2D)        (None, None, None, 3 24576       block_9_ad
d[0][0]
____________________________________________________________________________________________
____________________
block_10_expand_BN (BatchNormal (None, None, None, 3 1536        block_10_e
xpand[0][0]
____________________________________________________________________________________________
____________________
block_10_expand_relu (ReLU)     (None, None, None, 3 0           block_10_e
xpand_BN[0][0]
____________________________________________________________________________________________
____________________
block_10_depthwise (DepthwiseCo (None, None, None, 3 3456        block_10_e
xpand_relu[0][0]
____________________________________________________________________________________________
____________________
block_10_depthwise_BN (BatchNor (None, None, None, 3 1536        block_10_d
epthwise[0][0]
____________________________________________________________________________________________
____________________
block_10_depthwise_relu (ReLU)  (None, None, None, 3 0           block_10_d
epthwise_BN[0][0]
____________________________________________________________________________________________
____________________
block_10_project (Conv2D)       (None, None, None, 9 36864       block_10_d
epthwise_relu[0][0]
____________________________________________________________________________________________
____________________
block_10_project_BN (BatchNorma (None, None, None, 9 384         block_10_p
roject[0][0]
____________________________________________________________________________________________
____________________
block_11_expand (Conv2D)        (None, None, None, 5 55296       block_10_p
roject_BN[0][0]
____________________________________________________________________________________________
____________________
block_11_expand_BN (BatchNormal (None, None, None, 5 2304        block_11_e
xpand[0][0]
____________________________________________________________________________________________
____________________
block_11_expand_relu (ReLU)     (None, None, None, 5 0           block_11_e
xpand_BN[0][0]
____________________________________________________________________________________________
____________________
block_11_depthwise (DepthwiseCo (None, None, None, 5 5184        block_11_e
xpand_relu[0][0]
____________________________________________________________________________________________
____________________
block_11_depthwise_BN (BatchNor (None, None, None, 5 2304        block_11_d
epthwise[0][0]
____________________________________________________________________________________________
____________________
block_11_depthwise_relu (ReLU)  (None, None, None, 5 0           block_11_d
epthwise_BN[0][0]
____________________________________________________________________________________________
____________________
block_11_project (Conv2D)       (None, None, None, 9 55296       block_11_d
epthwise_relu[0][0]
____________________________________________________________________________________________
```

```
__________________________
block_11_project_BN (BatchNorma (None, None, None, 9 384         block_11_p
roject[0][0]
__________________________________________________________________________
__________________________
block_11_add (Add)              (None, None, None, 9 0           block_10_p
roject_BN[0][0]
                                                                 block_11_p
roject_BN[0][0]
__________________________________________________________________________
__________________________
block_12_expand (Conv2D)        (None, None, None, 5 55296       block_11_a
dd[0][0]
__________________________________________________________________________
__________________________
block_12_expand_BN (BatchNormal (None, None, None, 5 2304        block_12_e
xpand[0][0]
__________________________________________________________________________
__________________________
block_12_expand_relu (ReLU)     (None, None, None, 5 0           block_12_e
xpand_BN[0][0]
__________________________________________________________________________
__________________________
block_12_depthwise (DepthwiseCo (None, None, None, 5 5184        block_12_e
xpand_relu[0][0]
__________________________________________________________________________
__________________________
block_12_depthwise_BN (BatchNor (None, None, None, 5 2304        block_12_d
epthwise[0][0]
__________________________________________________________________________
__________________________
block_12_depthwise_relu (ReLU)  (None, None, None, 5 0           block_12_d
epthwise_BN[0][0]
__________________________________________________________________________
__________________________
block_12_project (Conv2D)       (None, None, None, 9 55296       block_12_d
epthwise_relu[0][0]
__________________________________________________________________________
__________________________
block_12_project_BN (BatchNorma (None, None, None, 9 384         block_12_p
roject[0][0]
__________________________________________________________________________
__________________________
block_12_add (Add)              (None, None, None, 9 0           block_11_a
dd[0][0]
                                                                 block_12_p
roject_BN[0][0]
__________________________________________________________________________
__________________________
block_13_expand (Conv2D)        (None, None, None, 5 55296       block_12_a
dd[0][0]
__________________________________________________________________________
__________________________
block_13_expand_BN (BatchNormal (None, None, None, 5 2304        block_13_e
xpand[0][0]
__________________________________________________________________________
__________________________
block_13_expand_relu (ReLU)     (None, None, None, 5 0           block_13_e
xpand_BN[0][0]
__________________________________________________________________________
__________________________
```

```
block_13_pad (ZeroPadding2D)    (None, None, None, 5 0           block_13_e
xpand_relu[0][0]
__________________________________________________________________________
_____________________
block_13_depthwise (DepthwiseCo (None, None, None, 5 5184        block_13_p
ad[0][0]
__________________________________________________________________________
_____________________
block_13_depthwise_BN (BatchNor (None, None, None, 5 2304        block_13_d
epthwise[0][0]
__________________________________________________________________________
_____________________
block_13_depthwise_relu (ReLU)  (None, None, None, 5 0           block_13_d
epthwise_BN[0][0]
__________________________________________________________________________
_____________________
block_13_project (Conv2D)       (None, None, None, 1 92160       block_13_d
epthwise_relu[0][0]
__________________________________________________________________________
_____________________
block_13_project_BN (BatchNorma (None, None, None, 1 640         block_13_p
roject[0][0]
__________________________________________________________________________
_____________________
block_14_expand (Conv2D)        (None, None, None, 9 153600      block_13_p
roject_BN[0][0]
__________________________________________________________________________
_____________________
block_14_expand_BN (BatchNormal (None, None, None, 9 3840        block_14_e
xpand[0][0]
__________________________________________________________________________
_____________________
block_14_expand_relu (ReLU)     (None, None, None, 9 0           block_14_e
xpand_BN[0][0]
__________________________________________________________________________
_____________________
block_14_depthwise (DepthwiseCo (None, None, None, 9 8640        block_14_e
xpand_relu[0][0]
__________________________________________________________________________
_____________________
block_14_depthwise_BN (BatchNor (None, None, None, 9 3840        block_14_d
epthwise[0][0]
__________________________________________________________________________
_____________________
block_14_depthwise_relu (ReLU)  (None, None, None, 9 0           block_14_d
epthwise_BN[0][0]
__________________________________________________________________________
_____________________
block_14_project (Conv2D)       (None, None, None, 1 153600      block_14_d
epthwise_relu[0][0]
__________________________________________________________________________
_____________________
block_14_project_BN (BatchNorma (None, None, None, 1 640         block_14_p
roject[0][0]
__________________________________________________________________________
_____________________
block_14_add (Add)              (None, None, None, 1 0           block_13_p
roject_BN[0][0]
                                                                 block_14_p
roject_BN[0][0]
__________________________________________________________________________
_____________________
```

```
__________________________
block_15_expand (Conv2D)        (None, None, None, 9 153600      block_14_a
dd[0][0]
__________________________________________________________________________
__________________________
block_15_expand_BN (BatchNormal (None, None, None, 9 3840        block_15_e
xpand[0][0]
__________________________________________________________________________
__________________________
block_15_expand_relu (ReLU)     (None, None, None, 9 0           block_15_e
xpand_BN[0][0]
__________________________________________________________________________
__________________________
block_15_depthwise (DepthwiseCo (None, None, None, 9 8640        block_15_e
xpand_relu[0][0]
__________________________________________________________________________
__________________________
block_15_depthwise_BN (BatchNor (None, None, None, 9 3840        block_15_d
epthwise[0][0]
__________________________________________________________________________
__________________________
block_15_depthwise_relu (ReLU)  (None, None, None, 9 0           block_15_d
epthwise_BN[0][0]
__________________________________________________________________________
__________________________
block_15_project (Conv2D)       (None, None, None, 1 153600      block_15_d
epthwise_relu[0][0]
__________________________________________________________________________
__________________________
block_15_project_BN (BatchNorma (None, None, None, 1 640         block_15_p
roject[0][0]
__________________________________________________________________________
__________________________
block_15_add (Add)              (None, None, None, 1 0           block_14_a
dd[0][0]
                                                                 block_15_p
roject_BN[0][0]
__________________________________________________________________________
__________________________
block_16_expand (Conv2D)        (None, None, None, 9 153600      block_15_a
dd[0][0]
__________________________________________________________________________
__________________________
block_16_expand_BN (BatchNormal (None, None, None, 9 3840        block_16_e
xpand[0][0]
__________________________________________________________________________
__________________________
block_16_expand_relu (ReLU)     (None, None, None, 9 0           block_16_e
xpand_BN[0][0]
__________________________________________________________________________
__________________________
block_16_depthwise (DepthwiseCo (None, None, None, 9 8640        block_16_e
xpand_relu[0][0]
__________________________________________________________________________
__________________________
block_16_depthwise_BN (BatchNor (None, None, None, 9 3840        block_16_d
epthwise[0][0]
__________________________________________________________________________
__________________________
block_16_depthwise_relu (ReLU)  (None, None, None, 9 0           block_16_d
epthwise_BN[0][0]
```

```
__________________________________________________________________________________________
_____________________
block_16_project (Conv2D)       (None, None, None, 3 307200      block_16_d
epthwise_relu[0][0]
__________________________________________________________________________________________
_____________________
block_16_project_BN (BatchNorma (None, None, None, 3 1280        block_16_p
roject[0][0]
__________________________________________________________________________________________
_____________________
Conv_1 (Conv2D)                 (None, None, None, 1 409600      block_16_p
roject_BN[0][0]
__________________________________________________________________________________________
_____________________
Conv_1_bn (BatchNormalization)  (None, None, None, 1 5120        Conv_1[0][
0]
__________________________________________________________________________________________
_____________________
out_relu (ReLU)                 (None, None, None, 1 0           Conv_1_bn[
0][0]
__________________________________________________________________________________________
_____________________
global_average_pooling2d (Globa (None, 1280)         0           out_relu[0
][0]
__________________________________________________________________________________________
_____________________
flatten_1 (Flatten)             (None, 1280)         0           global_ave
rage_pooling2d[0][0]
__________________________________________________________________________________________
_____________________
dense_8 (Dense)                 (None, 20)           25620       flatten_1[
0][0]
==========================================================================================
======================
Total params: 2,283,604
Trainable params: 25,620
Non-trainable params: 2,257,984
__________________________________________________________________________________________
_____________________
Epoch 1/5
12/12 [==============================] - 12s 827ms/step - loss: 1.7498 - bi
nary_accuracy: 0.9662
Epoch 2/5
12/12 [==============================] - 10s 833ms/step - loss: 1.7459 - bi
nary_accuracy: 0.9662
Epoch 3/5
12/12 [==============================] - 12s 1s/step - loss: 1.7403 - binar
y_accuracy: 0.9664
Epoch 4/5
12/12 [==============================] - 10s 827ms/step - loss: 1.7568 - bi
nary_accuracy: 0.9672
Epoch 5/5
12/12 [==============================] - 10s 884ms/step - loss: 1.7579 - bi
nary_accuracy: 0.9665
```

Out[14]: <keras.callbacks.History at 0x7f1a713dca10>

Once we finished training our model, we use the following code to make our classifications on the test data. We are using the merged_MobileNetV2_model which was made by adding output layers to the frozen layers of the base of the MobileNetV2. We are not using the retrained model with the unfrozen layers because it did not improve the performance.

In [15]:
```
#image_size = (256, 256)

test_img = img_resize(test_df, image_size)

threshold = 0.15

images = []
im = test_img/255
indexes = np.arange(20)
predictions = merged_MobileNetV2_model.predict(im)
for i in range(len(test_img)):
    match = False
    for ind in indexes:
        if(predictions[i, ind] > threshold):
            test_df.at[i,labels[ind]] = 1
            match = True
        else:
            test_df.at[i,labels[ind]] = 0
    if match:
        match = False
    else:
        test_df.at[i, labels[np.argmax(predictions[i])]] = 1
```

3. Semantic segmentation

The goal here is to implement a segmentation CNN that labels every pixel in the image as belonging to one of the 20 classes (and/or background). Use the training set to train your CNN and compete on the test set (by filling in the segmentation column in the test dataframe).

In [16]:
```
!mkdir -p images_to_train/x images_to_train/y
```

In [17]:
```
for i, j in enumerate(zip(train_df["img"].to_numpy(), train_df["seg"].to_nu
    x, y = j
    Image.fromarray(x).save(f"images_to_train/x/x{i}.png","PNG")
    Image.fromarray(y).save(f"images_to_train/y/y{i}.png","PNG")
```

We need to create a Dataset using Tensorflow for the images that we will use to train.

In [18]:
```
#IMAGE_SIZE = 512
IMAGE_SIZE = 224
BATCH_SIZE = 4
NUM_CLASSES = 21
DATA_DIR = "./images_to_train/"
NUM_TRAIN_IMAGES = 675 # change if we decide to use the data agumentation
NUM_VAL_IMAGES = 74 # change if we decide to use the data agumentation

train_images = sorted(glob(os.path.join(DATA_DIR, "x/*")))[:NUM_TRAIN_IMAGI
train_masks = sorted(glob(os.path.join(DATA_DIR, "y/*")))[:NUM_TRAIN_IMAGES
val_images = sorted(glob(os.path.join(DATA_DIR, "x/*")))[
    NUM_TRAIN_IMAGES : NUM_VAL_IMAGES + NUM_TRAIN_IMAGES
]
val_masks = sorted(glob(os.path.join(DATA_DIR, "y/*")))[
    NUM_TRAIN_IMAGES : NUM_VAL_IMAGES + NUM_TRAIN_IMAGES
]
```

In [19]:
```
def read_image(image_path, mask=False):
    image = tf.io.read_file(image_path)
    if mask:
        image = tf.image.decode_png(image, channels=1)
        image.set_shape([None, None, 1])
        image = tf.image.resize(images=image, size=[IMAGE_SIZE, IMAGE_SIZE
    else:
        image = tf.image.decode_png(image, channels=3)
        image.set_shape([None, None, 3])
        image = tf.image.resize(images=image, size=[IMAGE_SIZE, IMAGE_SIZE
        image = image / 127.5 - 1
    return image


def load_data(image_list, mask_list):
    image = read_image(image_list)
    mask = read_image(mask_list, mask=True)
    return image, mask


def data_generator(image_list, mask_list):
    dataset = tf.data.Dataset.from_tensor_slices((image_list, mask_list))
    dataset = dataset.map(load_data, num_parallel_calls=tf.data.AUTOTUNE)
    dataset = dataset.batch(BATCH_SIZE, drop_remainder=True)
    return dataset



train_dataset = data_generator(train_images, train_masks)
val_dataset = data_generator(val_images, val_masks)

print("Train Dataset:", train_dataset)
print("Val Dataset:", val_dataset)
```

```
Train Dataset: <BatchDataset shapes: ((4, 224, 224, 3), (4, 224, 224, 1)),
types: (tf.float32, tf.float32)>
Val Dataset: <BatchDataset shapes: ((4, 224, 224, 3), (4, 224, 224, 1)), ty
pes: (tf.float32, tf.float32)>
```

In [20]:

```
from tensorflow.keras import layers
from tensorflow import keras
```

DeeplabV3+

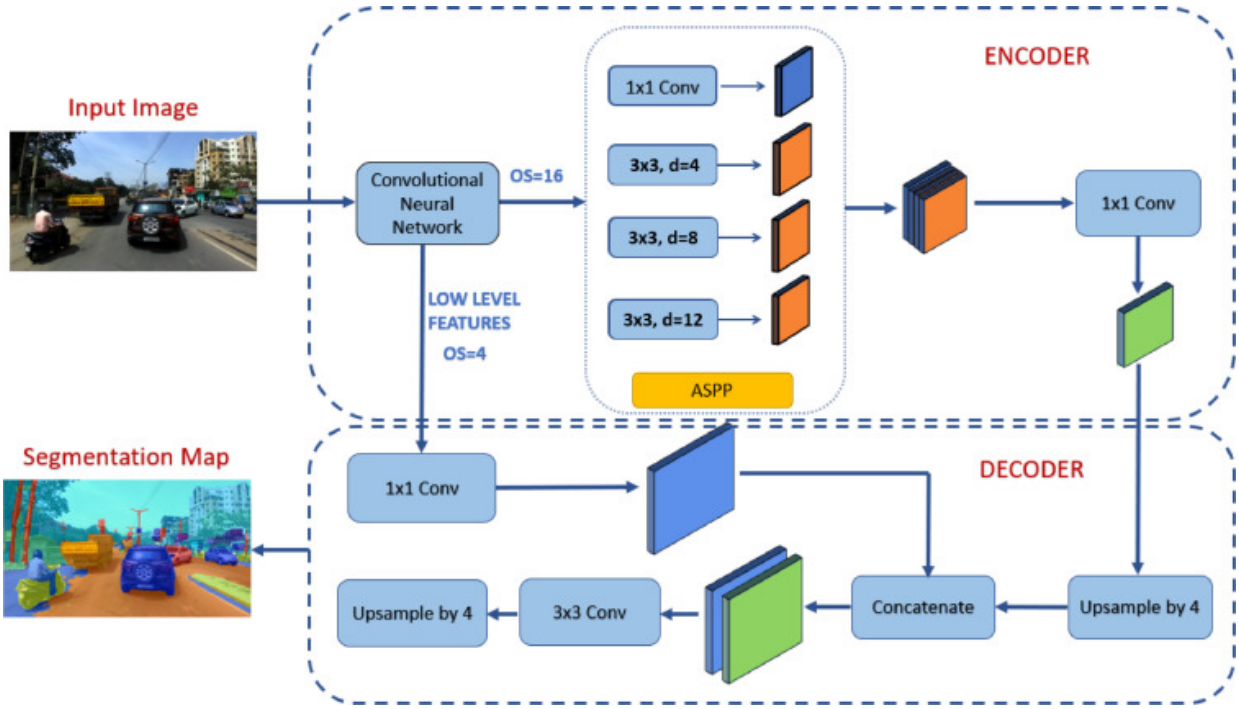
DeepLabv3+ is a model for semantic segmentation, where the goal is to assign semantic labels to every pixel in the input image. It has an encoding phase and a decoding phase, in the encoding phase it extracts the essential information of the image using CNN, while the decoding phase reconstructs the output with the information obtained in the encoder part. The decoder part was added to better segment along object boundaries, it is easy to see that DeepLabv3+ is a large model needing a great amount of processing.
https://keras.io/examples/vision/deeplabv3_plus/

Building...

Now we are ready to build our model!

The model that we are using here has a encoder-decoder structure, where in the encoder part the processing of contextual information happens by applying the dilated CNN at multiple scales and the decoder refine object boundaries of the segmentation.

Image from: https://www.sciencedirect.com/science/article/pii/S0167865520302750



In [21]:
```
def convolution_block(
    block_input,
    num_filters=256,
    kernel_size=3,
    dilation_rate=1,
    padding="same",
    use_bias=False,
):
    x = layers.Conv2D(
        num_filters,
        kernel_size=kernel_size,
        dilation_rate=dilation_rate,
        padding="same",
        use_bias=use_bias,
        kernel_initializer=keras.initializers.HeNormal(),
    )(block_input)
    x = layers.BatchNormalization()(x)
    return tf.nn.relu(x)


def DilatedSpatialPyramidPooling(dspp_input):
    dims = dspp_input.shape
    x = layers.AveragePooling2D(pool_size=(dims[-3], dims[-2]))(dspp_input
    x = convolution_block(x, kernel_size=1, use_bias=True)
    out_pool = layers.UpSampling2D(
        size=(dims[-3] // x.shape[1], dims[-2] // x.shape[2]), interpolati
    )(x)

    out_1 = convolution_block(dspp_input, kernel_size=1, dilation_rate=1)
    out_6 = convolution_block(dspp_input, kernel_size=3, dilation_rate=6)
    out_12 = convolution_block(dspp_input, kernel_size=3, dilation_rate=12
    out_18 = convolution_block(dspp_input, kernel_size=3, dilation_rate=18

    x = layers.Concatenate(axis=-1)([out_pool, out_1, out_6, out_12, out_1
    output = convolution_block(x, kernel_size=1)
    return output
```

We will use a MobileNet model pretrained on ImageNet

In [22]:
```
model_input = keras.Input(shape=(224, 224, 3))
mob_net = keras.applications.MobileNet(
        weights="imagenet", include_top=False, input_tensor=model_input
    )
mob_net.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-appli
cations/mobilenet/mobilenet_1_0_224_tf_no_top.h5
17227776/17225924 [==============================] - 0s 0us/step
17235968/17225924 [==============================] - 0s 0us/step
Model: "mobilenet_1.00_224"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
```

conv1 (Conv2D)	(None, 112, 112, 32)	864
conv1_bn (BatchNormalization	(None, 112, 112, 32)	128
conv1_relu (ReLU)	(None, 112, 112, 32)	0
conv_dw_1 (DepthwiseConv2D)	(None, 112, 112, 32)	288
conv_dw_1_bn (BatchNormaliza	(None, 112, 112, 32)	128
conv_dw_1_relu (ReLU)	(None, 112, 112, 32)	0
conv_pw_1 (Conv2D)	(None, 112, 112, 64)	2048
conv_pw_1_bn (BatchNormaliza	(None, 112, 112, 64)	256
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 (BatchNormaliza	(None, 56, 56, 64)	256
conv_dw_2_relu (ReLU)	(None, 56, 56, 64)	0
conv_pw_2 (Conv2D)	(None, 56, 56, 128)	8192
conv_pw_2_bn (BatchNormaliza	(None, 56, 56, 128)	512
conv_pw_2_relu (ReLU)	(None, 56, 56, 128)	0
conv_dw_3 (DepthwiseConv2D)	(None, 56, 56, 128)	1152
conv_dw_3_bn (BatchNormaliza	(None, 56, 56, 128)	512
conv_dw_3_relu (ReLU)	(None, 56, 56, 128)	0
conv_pw_3 (Conv2D)	(None, 56, 56, 128)	16384
conv_pw_3_bn (BatchNormaliza	(None, 56, 56, 128)	512
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 (BatchNormaliza	(None, 28, 28, 128)	512
conv_dw_4_relu (ReLU)	(None, 28, 28, 128)	0
conv_pw_4 (Conv2D)	(None, 28, 28, 256)	32768
conv_pw_4_bn (BatchNormaliza	(None, 28, 28, 256)	1024
conv_pw_4_relu (ReLU)	(None, 28, 28, 256)	0
conv_dw_5 (DepthwiseConv2D)	(None, 28, 28, 256)	2304
conv_dw_5_bn (BatchNormaliza	(None, 28, 28, 256)	1024

conv_dw_5_relu (ReLU)	(None, 28, 28, 256)	0
conv_pw_5 (Conv2D)	(None, 28, 28, 256)	65536
conv_pw_5_bn (BatchNormaliza	(None, 28, 28, 256)	1024
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 (BatchNormaliza	(None, 14, 14, 256)	1024
conv_dw_6_relu (ReLU)	(None, 14, 14, 256)	0
conv_pw_6 (Conv2D)	(None, 14, 14, 512)	131072
conv_pw_6_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_pw_6_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_7 (DepthwiseConv2D)	(None, 14, 14, 512)	4608
conv_dw_7_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_dw_7_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_7 (Conv2D)	(None, 14, 14, 512)	262144
conv_pw_7_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_pw_7_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_8 (DepthwiseConv2D)	(None, 14, 14, 512)	4608
conv_dw_8_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_dw_8_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_8 (Conv2D)	(None, 14, 14, 512)	262144
conv_pw_8_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_pw_8_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_9 (DepthwiseConv2D)	(None, 14, 14, 512)	4608
conv_dw_9_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_dw_9_relu (ReLU)	(None, 14, 14, 512)	0
conv_pw_9 (Conv2D)	(None, 14, 14, 512)	262144
conv_pw_9_bn (BatchNormaliza	(None, 14, 14, 512)	2048
conv_pw_9_relu (ReLU)	(None, 14, 14, 512)	0
conv_dw_10 (DepthwiseConv2D)	(None, 14, 14, 512)	4608

```
conv_dw_10_bn (BatchNormaliz (None, 14, 14, 512)       2048
_________________________________________________________________
conv_dw_10_relu (ReLU)       (None, 14, 14, 512)       0
_________________________________________________________________
conv_pw_10 (Conv2D)          (None, 14, 14, 512)       262144
_________________________________________________________________
conv_pw_10_bn (BatchNormaliz (None, 14, 14, 512)       2048
_________________________________________________________________
conv_pw_10_relu (ReLU)       (None, 14, 14, 512)       0
_________________________________________________________________
conv_dw_11 (DepthwiseConv2D) (None, 14, 14, 512)       4608
_________________________________________________________________
conv_dw_11_bn (BatchNormaliz (None, 14, 14, 512)       2048
_________________________________________________________________
conv_dw_11_relu (ReLU)       (None, 14, 14, 512)       0
_________________________________________________________________
conv_pw_11 (Conv2D)          (None, 14, 14, 512)       262144
_________________________________________________________________
conv_pw_11_bn (BatchNormaliz (None, 14, 14, 512)       2048
_________________________________________________________________
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 (BatchNormaliz (None, 7, 7, 512)         2048
_________________________________________________________________
conv_dw_12_relu (ReLU)       (None, 7, 7, 512)         0
_________________________________________________________________
conv_pw_12 (Conv2D)          (None, 7, 7, 1024)        524288
_________________________________________________________________
conv_pw_12_bn (BatchNormaliz (None, 7, 7, 1024)        4096
_________________________________________________________________
conv_pw_12_relu (ReLU)       (None, 7, 7, 1024)        0
_________________________________________________________________
conv_dw_13 (DepthwiseConv2D) (None, 7, 7, 1024)        9216
_________________________________________________________________
conv_dw_13_bn (BatchNormaliz (None, 7, 7, 1024)        4096
_________________________________________________________________
conv_dw_13_relu (ReLU)       (None, 7, 7, 1024)        0
_________________________________________________________________
conv_pw_13 (Conv2D)          (None, 7, 7, 1024)        1048576
_________________________________________________________________
conv_pw_13_bn (BatchNormaliz (None, 7, 7, 1024)        4096
_________________________________________________________________
conv_pw_13_relu (ReLU)       (None, 7, 7, 1024)        0
=================================================================
Total params: 3,228,864
Trainable params: 3,206,976
Non-trainable params: 21,888
_________________________________________________________________
```

We will also use the features (low-level) from conv_pw_11_bn to concatenate with the encoder features.

In [23]:

```
def DeeplabV3Plus(image_size, num_classes):
    model_input = keras.Input(shape=(image_size, image_size, 3))
    mob_net = keras.applications.MobileNet(
        weights="imagenet", include_top=False, input_tensor=model_input
    )
    x = mob_net.get_layer("conv_pw_11_bn").output

    x = DilatedSpatialPyramidPooling(x)

    input_a = layers.UpSampling2D(
        size=(image_size // 4 // x.shape[1], image_size // 4 // x.shape[2]
        interpolation="bilinear",
    )(x)
    input_b = mob_net.get_layer("conv_pw_3_bn").output
    input_b = convolution_block(input_b, num_filters=48, kernel_size=1)

    x = layers.Concatenate(axis=-1)([input_a, input_b])
    x = convolution_block(x)
    x = convolution_block(x)
    x = layers.UpSampling2D(
        size=(image_size // x.shape[1], image_size // x.shape[2]),
        interpolation="bilinear",
    )(x)
    model_output = layers.Conv2D(num_classes, kernel_size=(1, 1), padding='
    return keras.Model(inputs=model_input, outputs=model_output)


model = DeeplabV3Plus(image_size=IMAGE_SIZE, num_classes=NUM_CLASSES)
```

Ready for the finetuning! Let's go!

For the optimizer we use Adam and for the loss Sparse Categorical Crossentropy

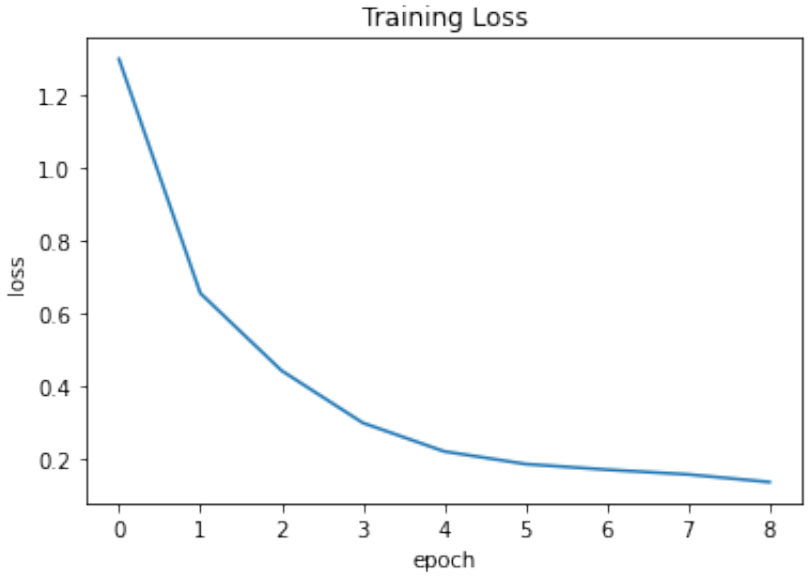
The checkpointer will save the best model Also was applied the learning rate reduction when the model stops learning for consecutives epochs. If the model continues not improving, there will be an early stopping. We also add plots to see how the model is performing.

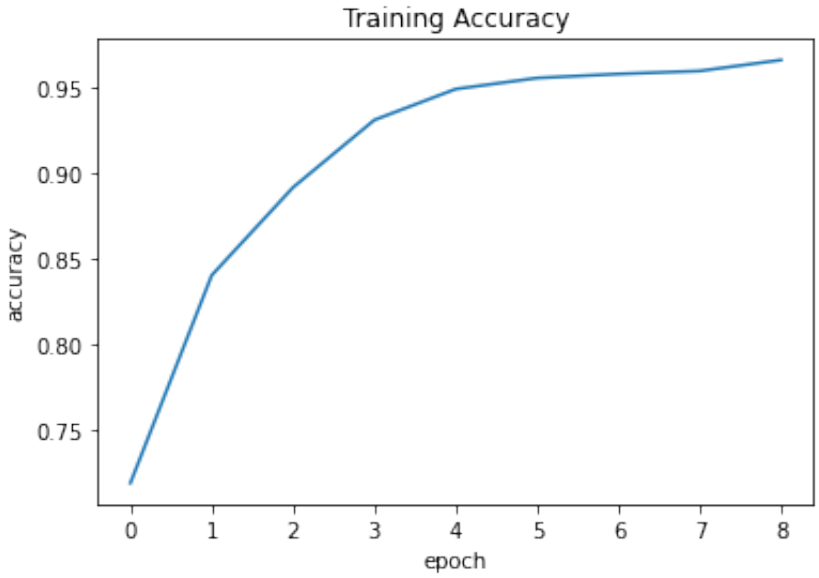
In [24]:

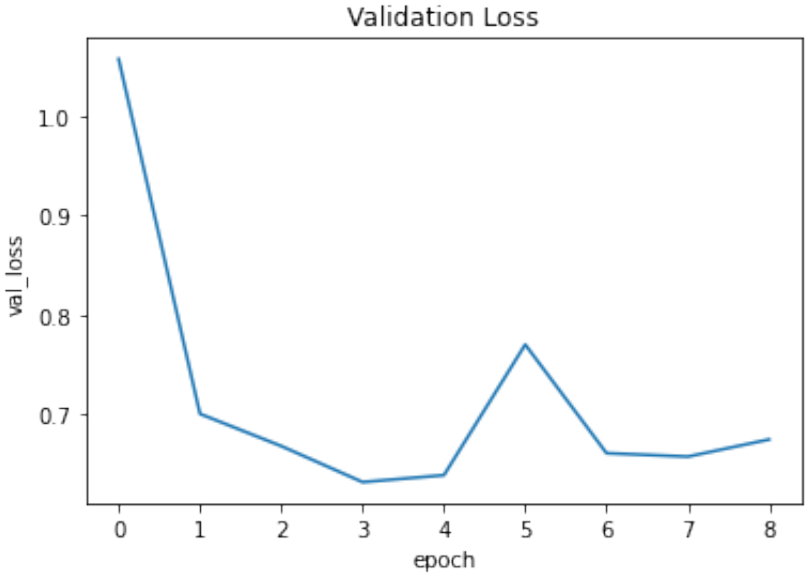
```
earlystopper = EarlyStopping(patience=5, verbose=1)
checkpointer = ModelCheckpoint("seg_model_mobilenet.h5" , verbose=1, save_
learning_rate_reduction = ReduceLROnPlateau(monitor='val_loss',
                                            patience=3,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=0.000001)

loss = keras.losses.SparseCategoricalCrossentropy(from_logits=True)
model.compile(
    optimizer=keras.optimizers.Adam(learning_rate=0.0001),
    loss=loss,
    metrics=["accuracy"]
)

history = model.fit(train_dataset, validation_data=val_dataset, epochs=40,
                    callbacks=[earlystopper,checkpointer,learning_rate_redu

plt.plot(history.history["loss"])
plt.title("Training Loss")
plt.ylabel("loss")
plt.xlabel("epoch")
plt.show()

plt.plot(history.history["accuracy"])
plt.title("Training Accuracy")
plt.ylabel("accuracy")
plt.xlabel("epoch")
plt.show()

plt.plot(history.history["val_loss"])
plt.title("Validation Loss")
plt.ylabel("val_loss")
plt.xlabel("epoch")
plt.show()

plt.plot(history.history["val_accuracy"])
plt.title("Validation Accuracy")
plt.ylabel("val_accuracy")
plt.xlabel("epoch")
plt.show()
```

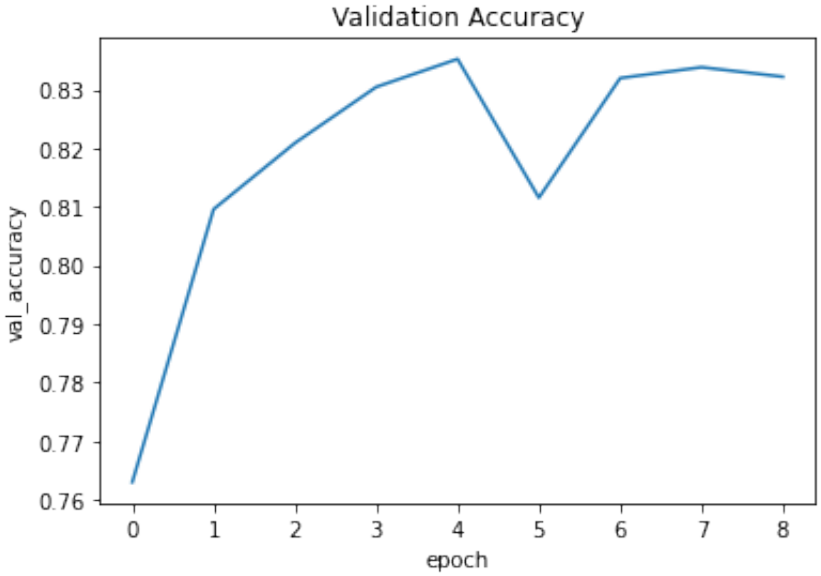
```
Epoch 1/40
168/168 [==============================] - 16s 74ms/step - loss: 1.2985 - a
ccuracy: 0.7191 - val_loss: 1.0576 - val_accuracy: 0.7630

Epoch 00001: val_loss improved from inf to 1.05762, saving model to seg_mod
el_mobilenet.h5
Epoch 2/40
168/168 [==============================] - 13s 78ms/step - loss: 0.6542 - a
ccuracy: 0.8403 - val_loss: 0.7002 - val_accuracy: 0.8096

Epoch 00002: val_loss improved from 1.05762 to 0.70022, saving model to seg
_model_mobilenet.h5
Epoch 3/40
168/168 [==============================] - 12s 71ms/step - loss: 0.4411 - a
```

```
ccuracy: 0.8916 - val_loss: 0.6678 - val_accuracy: 0.8209

Epoch 00003: val_loss improved from 0.70022 to 0.66776, saving model to seg
_model_mobilenet.h5
Epoch 4/40
168/168 [==============================] - 13s 76ms/step - loss: 0.2973 - a
ccuracy: 0.9310 - val_loss: 0.6315 - val_accuracy: 0.8304

Epoch 00004: val_loss improved from 0.66776 to 0.63148, saving model to seg
_model_mobilenet.h5
Epoch 5/40
168/168 [==============================] - 12s 70ms/step - loss: 0.2190 - a
ccuracy: 0.9490 - val_loss: 0.6384 - val_accuracy: 0.8352

Epoch 00005: val_loss did not improve from 0.63148
Epoch 6/40
168/168 [==============================] - 13s 78ms/step - loss: 0.1842 - a
ccuracy: 0.9554 - val_loss: 0.7700 - val_accuracy: 0.8115

Epoch 00006: val_loss did not improve from 0.63148
Epoch 7/40
168/168 [==============================] - 12s 70ms/step - loss: 0.1687 - a
ccuracy: 0.9578 - val_loss: 0.6606 - val_accuracy: 0.8320

Epoch 00007: val_loss did not improve from 0.63148

Epoch 00007: ReduceLROnPlateau reducing learning rate to 4.999999873689376e
-05.
Epoch 8/40
168/168 [==============================] - 12s 72ms/step - loss: 0.1556 - a
ccuracy: 0.9595 - val_loss: 0.6571 - val_accuracy: 0.8338

Epoch 00008: val_loss did not improve from 0.63148
Epoch 9/40
168/168 [==============================] - 13s 78ms/step - loss: 0.1346 - a
ccuracy: 0.9659 - val_loss: 0.6744 - val_accuracy: 0.8322

Epoch 00009: val_loss did not improve from 0.63148
Epoch 00009: early stopping
```









Now, we are loading the saved model that was defined in the checkpointer.

In [25]:
```
m = load_model("seg_model_mobilenet.h5")
```

In [26]:
```
def infer(model, image_tensor):
    predictions = model.predict(np.expand_dims((image_tensor), axis=0))
    predictions = np.squeeze(predictions)
    predictions = np.argmax(predictions, axis=2)
    return predictions

def plot_predictions(images_list, colormap, model):
    for image_file in images_list:
        image_tensor = read_image(image_file)
        prediction_mask = infer(image_tensor=image_tensor, model=model)
        return prediction_mask
```

In [27]:
```
!mkdir -p test_images
```

In [28]:
```
def resize_img(imgs, width, height, interpolation=cv2.INTER_LINEAR):
    resized_imgs = []
    for mask in imgs:
        im_resize = cv2.resize(mask,
                               (width, height),
                               interpolation=interpolation)
        resized_imgs.append(im_resize)
    return resized_imgs
```

Now,we are applying the segmentation in the test images and saving the results.

In [29]:
```
final_images_seg = []
for i, j in enumerate(test_df["img"].to_numpy()):
    shape = j.shape[:2][::-1]
    im_name = f"test_images/x{i}.png"
    Image.fromarray(j).save(im_name,"PNG")
    img_infer = infer(m, read_image(im_name))
    final_images_seg.append(resize_img([img_infer], *shape, interpolation=
```

In [30]:
```
test_df["seg"] = final_images_seg
```

In [31]:
```
test_df.head()
```

Out[31]:

Id	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	...	horse	motorbike

0	0	0	0	0	1	0	0	0	1	0	...	0	0
1	0	0	0	0	0	0	0	0	1	0	...	0	0
2	0	0	0	0	0	0	0	0	1	0	...	0	0
3	0	1	0	0	0	0	0	0	0	0	...	0	0

4	0	0	0	0	0	0	0	0	1	0	...	0	0

5 rows × 22 columns

Submit to competition

We have encountered some issues submitting the csv file to the competition after saving the notebook. After checking Kaggle forums and trying a few ways to fix this issue, we found out that Kaggle cannot find the submission file if the notebook has more than 500 output files. The best workaround is to produce fewer output files on the notebook version that will be used to make a submission.

Our output files are just training images used in the segmentation section and a saved segmentation model. The following few lines delete those files so that we can be able to submit the output csv file.

In [32]:
```
import shutil
import os

shutil.rmtree('images_to_train')
shutil.rmtree('test_images')
os.remove('seg_model_mobilenet.h5')
```

The following line saves our predictions as a csv file to be submitted for the competition

In [33]:
```
generate_submission(test_df)
```

Out[33]:

Id	Predicted
0_classification	5 1 9 1 11 1
0_segmentation	2717727 4 2718222 12 2718722 12 2719220 16 271...
1_classification	9 1 16 1 18 1
1_segmentation	1636077 3 1636577 3 1637075 7 1637575 7 163807...
2_classification	9 1 18 1
...	...
747_segmentation	414138 4 414200 2 414635 7 415135 7 415629 13 ...
748_classification	11 1 15 1
748_segmentation	2696198 7 2696693 16 2697193 16 2697693 16 269...
749_classification	9 1 16 1 18 1
749_segmentation	

1500 rows × 1 columns

4. Adversarial attack

For this part, your goal is to fool your classification and/or segmentation CNN, using an *adversarial attack*. More specifically, the goal is build a CNN to perturb test images in a way that (i) they look unperturbed to humans; but (ii) the CNN classifies/segments these images in line with the perturbations.

We have taken the test data image from the “Cat” class, which has been correctly classified by the model with a high confidence interval. Then, noise perturbation of different values is added to each pixel based on its contribution to the loss value. Loss calculation is done while model parameters are constant. We use the infinity norm perturbation, which takes the largest absolute value of an element that doesn’t exceed precision constraint epsilon. Simply, it restricts the maximum bound on the change in activation function of the neural network model.

N = E.sign(w), where N is the perturbation, w is the weights and E is the precision constant.

We produce multiple images with different precision constant values. Based on experimentation, we have observed that original images that have already been classified with high accuracy require higher noise perturbations to fool the network (they are more robust).

As the perturbation is increased, the image becomes less recognizable to the human eye.

These attacks are realistic as noise perturbation occurs constantly in a real world setting, due to sensor and camera calibrations and sensitivity, and image resolution quality. Furthermore, a real may always be different from the images used to train the models. This makes the attack very realistic.

The attack conditions to be met are the same as we mimicked in the code. Another example that also produces some “noise” in the image is when a picture is taken from an original object, printed and then photographed again. We usually distinguish some “blur”/”noise” due to the losses in taking the image, printing and retaking an image and this could be a similar approach to fooling a system.

These attacks can also be instigated by adversaries for malicious goals. There are examples where the Google AI [6], a state of the art system was fooled. There is also other proof that shows how stickers and different angles impede correct classification of traffic signs for example [7]. As a consequence, each image recognition system needs to be handled with caution.

In [34]:
```
import tensorflow as tf
import matplotlib as mpl
import matplotlib.pyplot as plt

mpl.rcParams['figure.figsize'] = (8, 8)
mpl.rcParams['axes.grid'] = False
```

In [35]:
```
pretrained_model = merged_MobileNetV2_model
pretrained_model.trainable = False

# ImageNet labels
decode_predictions = tf.keras.applications.mobilenet_v2.decode_predictions
```

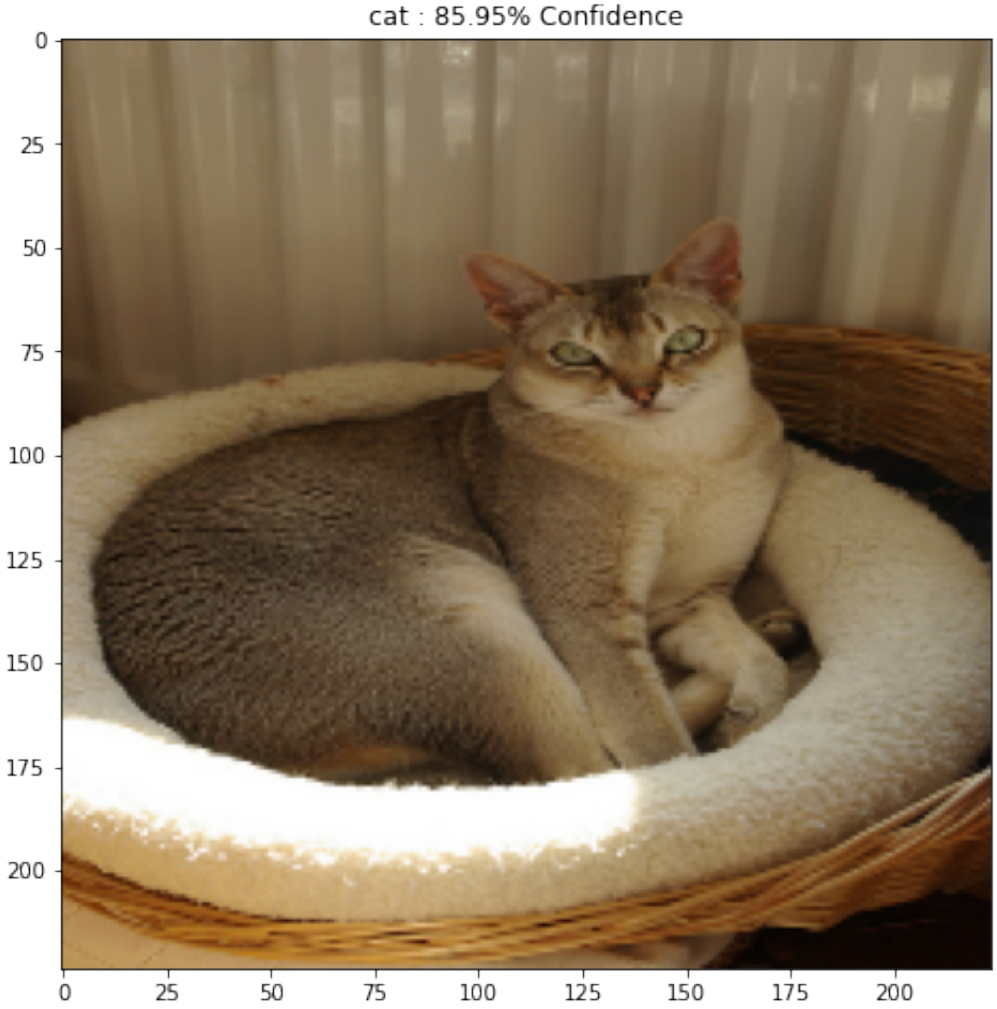
In [36]:
```
def preprocess(image):
    image = tf.cast(image, tf.float32)
    image = tf.image.resize(image, (224, 224))
    image = tf.keras.applications.mobilenet_v2.preprocess_input(image)
    image = image[None, ...]
    return image

def get_imagenet_label(probs):
    testNP = probs
    zeros = np.zeros(980)
    test_image_probs = np.append(testNP,zeros)
    yy = np.expand_dims(test_image_probs,0)

    return decode_predictions(yy, top=1)[0][0]#, resultingProb]
```

In [37]:
```
advIdx = 120
originalImage = test_df["img"][advIdx]
image = preprocess(originalImage)
```

In [38]:
```
image_probs = pretrained_model.predict(image)
```

In [39]:
```
plt.figure()
plt.imshow(image[0] * 0.5 + 0.5)  # To change [-1, 1] to [0,1]
_, image_class, class_confidence = get_imagenet_label(image_probs)
labelPred = np.argmax(predictions[advIdx])
labelTitle = test_df.columns[labelPred]
plt.title('{} : {:.2f}% Confidence'.format(labelTitle, class_confidence*10(
plt.show()
```

```
Downloading data from https://storage.googleapis.com/download.tensorflow.or
g/data/imagenet_class_index.json
40960/35363 [==================================] - 0s 0us/step
49152/35363 [==========================================] - 0s 0us/step
```

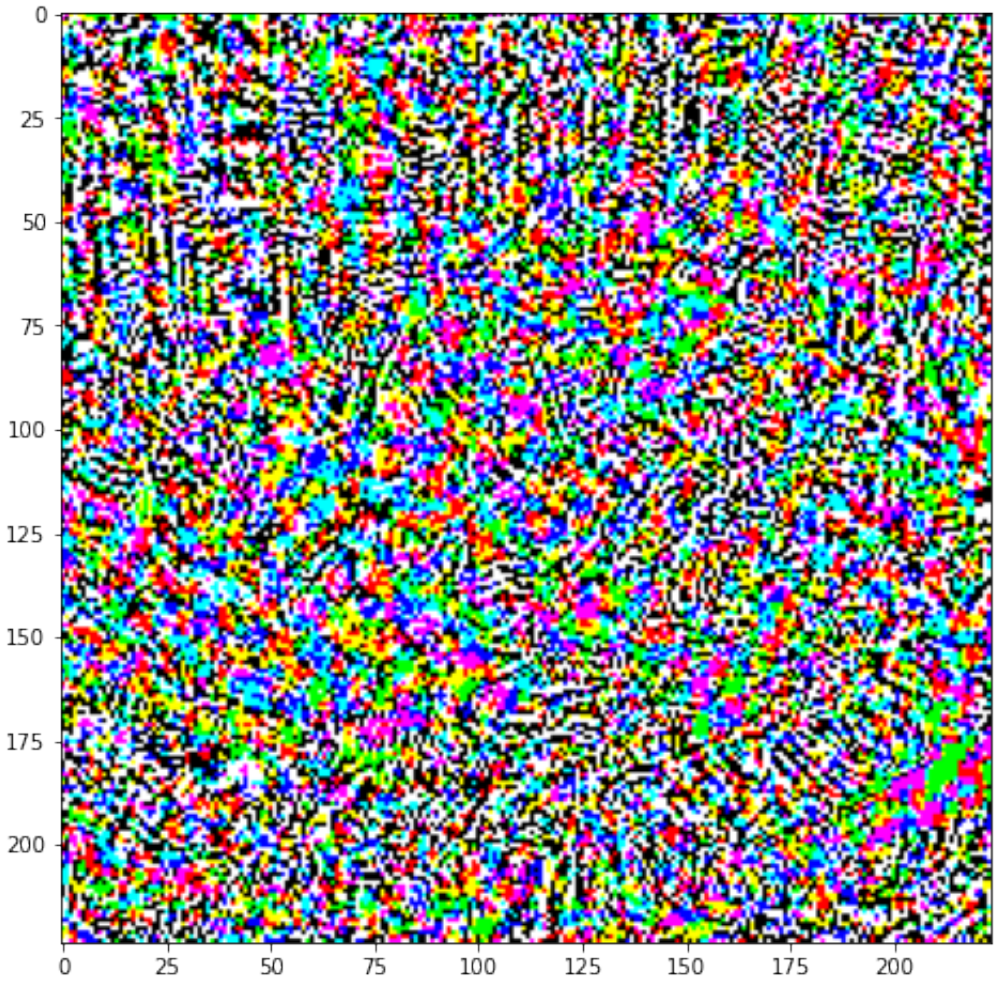


In [40]:

```
loss_object = tf.keras.losses.CategoricalCrossentropy()

def create_adversarial_pattern(input_image, input_label):
    with tf.GradientTape() as tape:
        tape.watch(input_image)
        prediction = pretrained_model(input_image)
        loss = loss_object(input_label, prediction)
    # Get the gradients of the loss w.r.t to the input image.
    gradient = tape.gradient(loss, input_image)
    # Get the sign of the gradients to create the perturbation
    signed_grad = tf.sign(gradient)
    return signed_grad
```

In [41]:

```
label = tf.one_hot(advIdx, image_probs.shape[-1])
label = tf.reshape(label, (1, image_probs.shape[-1]))

perturbations = create_adversarial_pattern(image, label)
plt.imshow(perturbations[0] * 0.5 + 0.5);  # To change [-1, 1] to [0,1]
```

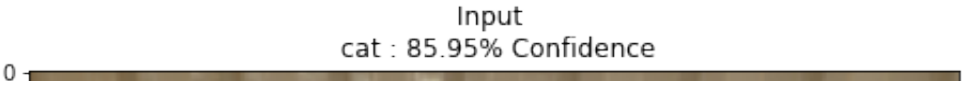


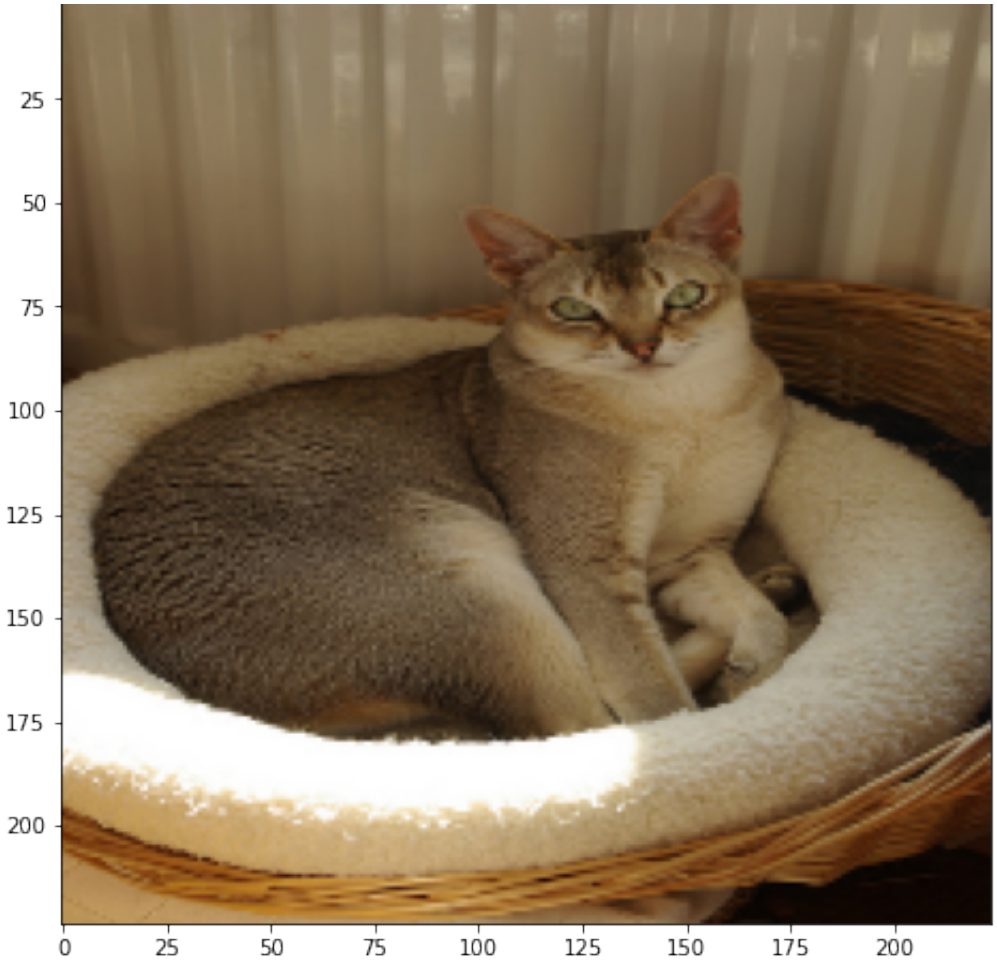
In [42]:

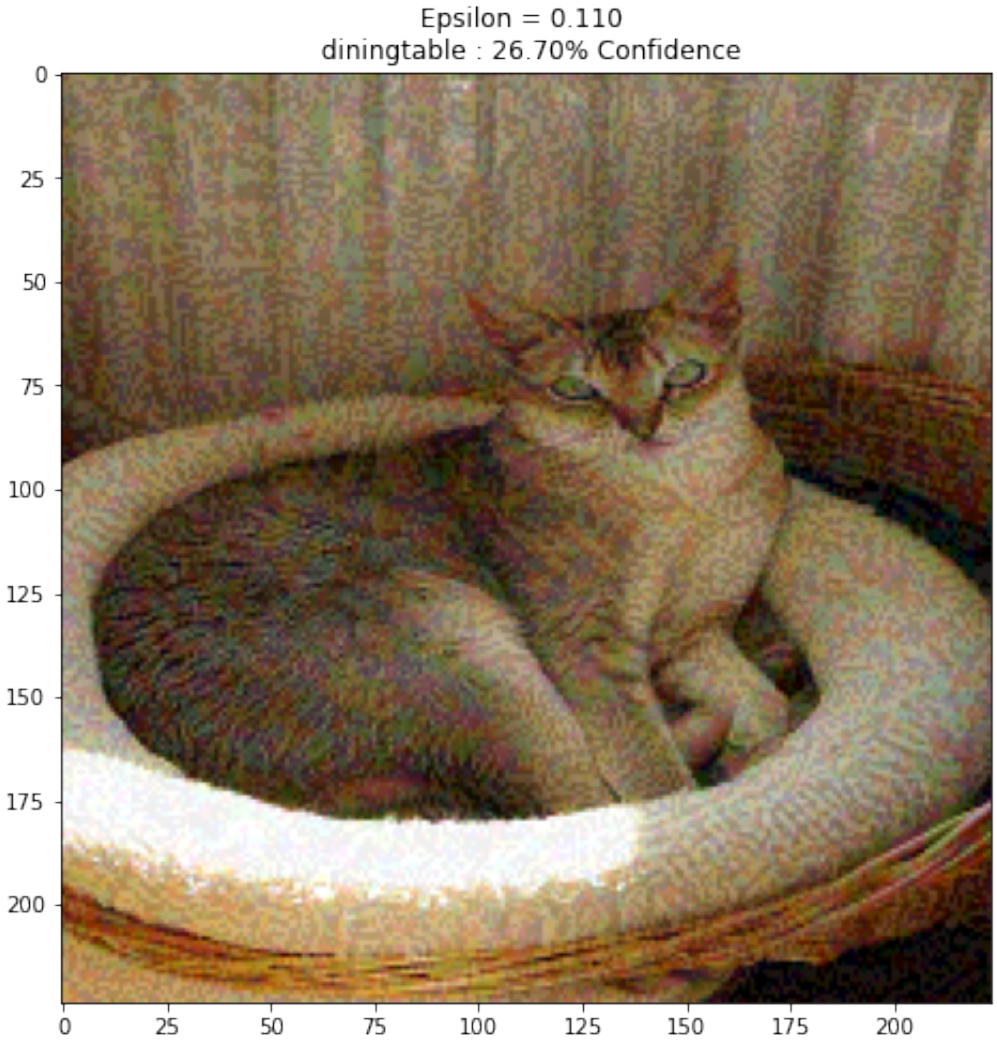
```
def display_images(image, description, labelTitle_Final):
    _, label, confidence, = get_imagenet_label(pretrained_model.predict(ima
    plt.figure()
    plt.imshow(image[0]*0.5+0.5)
    #plt.title('{} \n {} : {:.2f}% Confidence'.format(description,label, c
    predictedLabel = labelTitle_Final
    plt.title('{} \n {} : {:.2f}% Confidence'.format(description,predictedl
    plt.show()
```

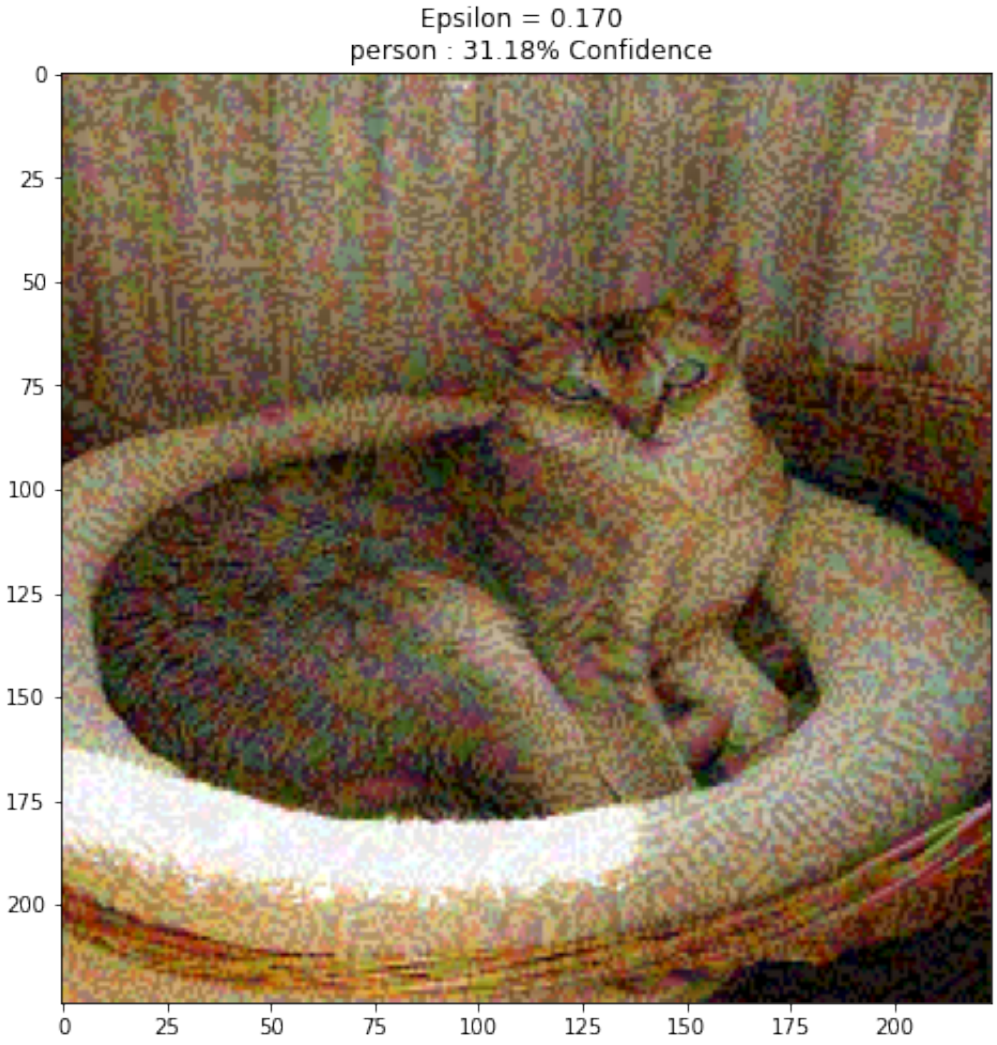
In [43]:

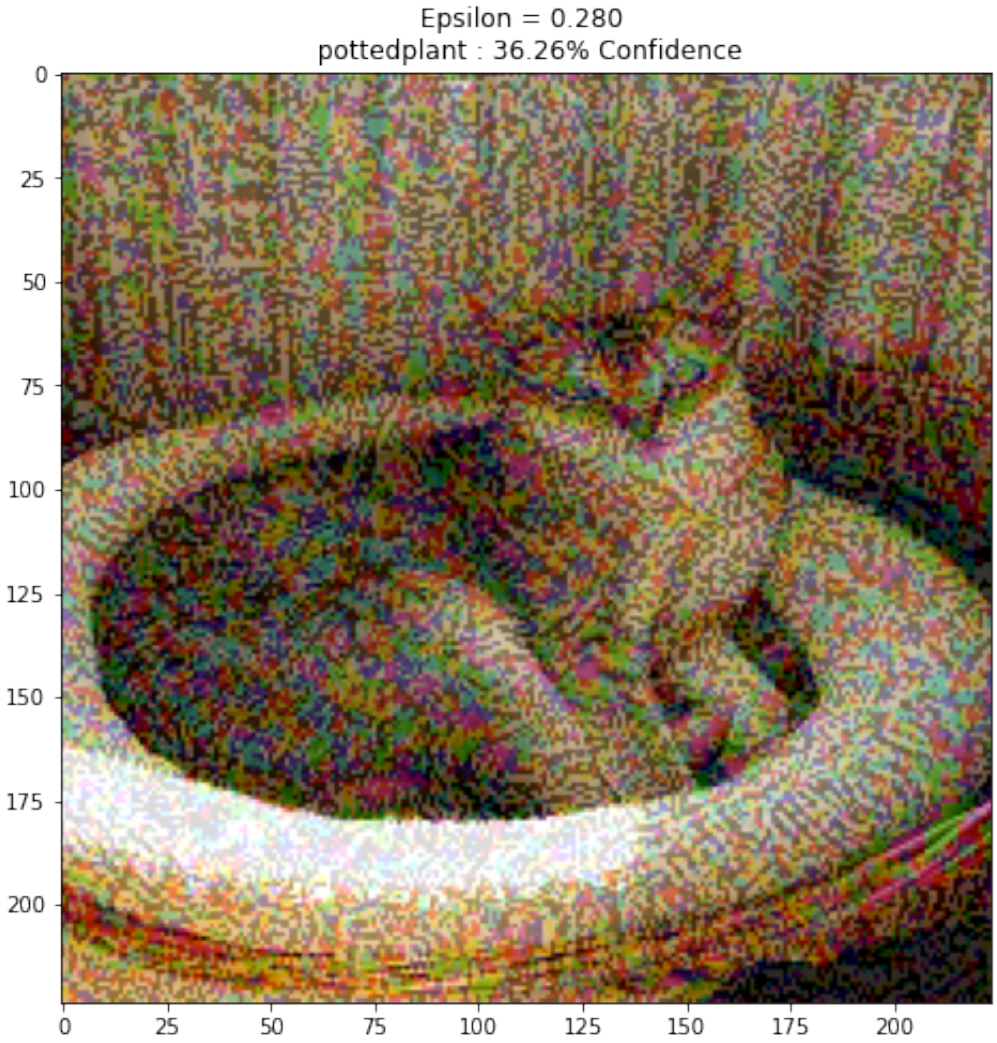
```
#epsilons = [0, 0.01, 0.1, 0.55]
epsilons = [0, 0.11, 0.17, 0.28]
descriptions = [('Epsilon = {:0.3f}'.format(eps) if eps else 'Input')
                for eps in epsilons]

for i, eps in enumerate(epsilons):
    adv_x = image + eps*perturbations
    adv_x = tf.clip_by_value(adv_x, -1, 1)
    newLabelProb = pretrained_model.predict(adv_x)
    labelPred = np.argmax(newLabelProb)
    labelTitle_Final = test_df.columns[labelPred]
    display_images(adv_x, descriptions[i],labelTitle_Final)
```




25
50
75
100
125
150
175
200
0
25
50
75
100
125
150
175
200


Epsilon = 0.110
diningtable : 26.70% Confidence
0
25
50
75
100
125
150
175
200
0
25
50
75
100
125
150
175
200


Epsilon = 0.170
person : 31.18% Confidence
0
25
50
75
100
125
150
175
200
0
25
50
75
100
125
150
175
200



5. Discussion

Classification

Classification of the image data has been done with a self-built CNN (with AlexNet Architecture), as well as transfer learning with the pretrained models of VGG16, ResNet50V2 and MobileNetV2. Among the four models used, the self-built CNN only improves accuracy very slowly. Resnet50V2 was underfitting and VGG16 suffers from overfitting - both models did not provide satisfactory results. MobilenetV2 provided the best results in terms of accuracy and loss and generalizes best on the test data.

This diverging behaviour is either a result of data preprocessing and augmentation or also the similarity of the base images, the different models have been pre-trained on. However, with more fitting of the pretrained models, Finally we picked MobileNetV2 as the final model. Fine tuning was attempted with unfreezing the last layer of the pretrained MobilenedV2 sequentially. Thus we first unfroze the last layer, then the second last, etc. and tested for improvements in the accuracy score. Whereas we looked at different accuracy measures such as hamming loss, accuracy, binary accuracy and categorical

accuracy.

Data augmentation

Data augmentation has been applied to expand the size of a training dataset by creating alternative versions of each image in the dataset. These alternative versions of the image are created by applying different edits to the images (e.g. zoom, brighteness, flipping... etc) does not only expand the training data but also makes the model more robust to variations in the images that it will have to classify in the test data.

Which semantic segmentation technique and why?

DeepLabV3+, several studies [1][2][3] have used this segmentation model with different versions of the PASCAL dataset. A comparisons between models were found[4], where we can see the great performance of DeepLabV3+ using the Pascal 2012.

Determination of threshold value proved to be difficult

The definition of what could be the optimal threshold was taken into account and prove to be difficult to be established: Some images do not have something to classify Some images has just one thing to be classified while others has several classes to be predicted Another issue was that sometimes the probability to have something in the image was really low and still a correct prediction, while others images have several high probabilities. Several thresholds functions were tested, the better predictions came from argmax

Could have done a second segmentation which is promising to produce better results?

Yes, we could have done a second segmentation. In the late stage we found a paper [5] that used a combination of two segmentations and got a better score using DeepLabV3+ in PASCAL VOC 2012. The authors got the class index score map using DeepLabV3+ and applied the superpixels in the input image using quick shift. They explain that it is hard for DeepLabV3+ produce a semantic segmentation with accurate boundaries since it is heavy to train and then the DCNN should adopt grouping and convolution to have parameters reduction and also the generated cascade features blur the boundaries. The result was promising in most cases, but still has some difficulties when the boundaries are similar with the background.

Reflections on Adversarial Attack

In the case of our model, it is especially susceptible to noise perturbations for images that are not predicted with a very high confidence score. Some of the solutions to increase robustness include:

Hiding the model gradients from adversaries, although an alternative black-box attack can overcome this defense. Distilling knowledge from large networks to smaller models, which contains the scope of attack. Making the image features less rich, so that the predicted probabilities are encoded with fewer values. This may reduce model accuracy performance. Using an additional model prior to testing, which filters between regular and adversarial examples.

We would like to specifically attempt the defense (3) by using PCA techniques to train on a reduced feature set data, instead of directly using the raw images and (4) use a model of continual learning (LSTM-RNN) that evaluates how the gradients change between images domains that would be otherwise similar.

Additionally, since our models appear easily fooled for images that include less rich features, such as edges and variety of colours, it would be interesting to tune parameters individually for such features.

[1] Zeng, Haibo, Siqi Peng, and Dongxiang Li. "Deeplabv3+ semantic segmentation model based on feature cross attention mechanism." Journal of Physics: Conference Series. Vol. 1678. No. 1. IOP Publishing, 2020.

[2] GitHub. 2022. GitHub - VainF/DeepLabV3Plus-Pytorch: DeepLabv3 and DeepLabv3+ with pretrained weights for Pascal VOC & Cityscapes. [online] Available at: https://github.com/VainF/DeepLabV3Plus-Pytorch.

[3] Papers.nips.cc. 2022. [online] Available at: https://papers.nips.cc/paper/2019/file/a67c8c9a961b4182688768dd9ba015fe-AuthorFeedback.pdf.

[4] Paperswithcode.com. 2022. Papers with Code - PASCAL VOC 2012 test Benchmark (Semantic Segmentation). [online] Available at: https://paperswithcode.com/sota/semantic-segmentation-on-pascal-voc-2012?p=wider-or-deeper-revisiting-the-resnet-model [Accessed 24 May 2022].

[5] Zhang, Sanxing & Ma, Zhenhuan & Zhang, Gang & Lei, Tao & Zhang, Rui & Cui, Yi. (2020). Semantic Image Segmentation with Deep Convolutional Neural Networks and Quick Shift. Symmetry. 12. 427. 10.3390/sym12030427.

[6] K. Eykholt et al., "Robust Physical-World Attacks on Deep Learning Visual Classification," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 1625-1634, doi: 10.1109/CVPR.2018.00175.

[7] A. Athalye et al., ″Synthesizing Robust Adversarial Examples″, Proceedings of the 35th International Conference on Machine Learning, {ICML} 2018,doi: 10.48550/arxiv.1707.07397