# Deep Learning in Practice


Anis Koubaa

## The Vehicle Type Classification Project

In this notebook, we will present how to perform a classification of car brands. We will use different state of the art classifiers in Tensorflow 2.0 and Keras.

## Summary

- **Name:** Anis Koubaa
- **Date:** 20 September 2020
- **Use Case:** Vehicle Type
- **Algorithm:** MobileNetV2
- **Number of training images:** 603
- **Number of classes:** 7
- **Batch Size:** 64
- **Optimizer:** Adam
- **Learning Rate:** 0.0001
- **Loss Type:**CategoricalCrossentropy
- **Transfer Learning:** Yes | Imagenet

**Comments:** We obtained 100% on the validation accuracy on vehicle types, on validation dataset.

Let's get started.

We first need to load the requires libraries

```
# import the necessary packages
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import AveragePooling2D, GlobalAveragePooling2D, Batch
#from tensorflow.keras.applications import ResNet50
#from tensorflow.keras.applications import Xception
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
```

```
headModel = BatchNormalization()(headModel)
headModel = Dense(number_of_classes, activation="softmax")(headModel)


# place the head FC model on top of the base model (this will become
# the actual model we will train)
model = Model(inputs=baseModel.input, outputs=headModel)


# loop over all layers in the base model and freeze them so they will
# *not* be updated during the first training process
for layer in baseModel.layers:
    layer.trainable = False

model.summary()
```

```
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import argparse
import cv2
import os
import sys
import tensorflow as tf
import h5py
import numpy as np
import sys
```

```
print(tf.__version__)
```

```
2.3.0
```

Then, we mount Google Drive to be able to access the files located on it

We now specify the path the dataset located on Google Drive

```
TYPE='type'
model_type='mobilenetv2'
user='anis'
iteration='2'

first_time_training=True

PROJECT_PATH='/content/drive/My Drive/udemy-deep-learning-in-practice/03-transfer-l
print('PROJECT_PATH: ',PROJECT_PATH)
HDF5_DATASET_PATH=PROJECT_PATH+'datasets/vehicle-type-dataset-SIZE224-train-dev-tes
print('HDF5_DATASET_PATH: ', HDF5_DATASET_PATH)
ACCURACY_LOSS_OUPUT_FILE=PROJECT_PATH+'log/'+model_type+'/'+model_type+'-by-'+TYPE+
TARGET_CLASSIFICATION_MODEL=PROJECT_PATH+'trained-models/'+model_type+'/'+'vehicle-
print('TARGET_CLASSIFICATION_MODEL: ',TARGET_CLASSIFICATION_MODEL)
CHECKPOINT_PATH = PROJECT_PATH+'checkpoints/'+model_type+'/'+'by-'+TYPE+'-'+model_t
print('CHECKPOINT_PATH: ',CHECKPOINT_PATH)
LOGFILE_PATH=PROJECT_PATH+'log/'+model_type+'/'+model_type+'-by-'+TYPE+'-training-l
print('LOGFILE_PATH: ',LOGFILE_PATH)
```

```
PROJECT_PATH:  /content/drive/My Drive/udemy-deep-learning-in-practice/03-trar
HDF5_DATASET_PATH:  /content/drive/My Drive/udemy-deep-learning-in-practice/03
TARGET_CLASSIFICATION_MODEL:  /content/drive/My Drive/udemy-deep-learning-in-p
CHECKPOINT_PATH:  /content/drive/My Drive/udemy-deep-learning-in-practice/03-t
LOGFILE_PATH:  /content/drive/My Drive/udemy-deep-learning-in-practice/03-trar
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
```

```
sys.path.append(PROJECT_PATH)
import anis_koubaa_udemy_computer_vision_lib
from anis_koubaa_udemy_computer_vision_lib import *
```

# Load the Dataset

```
def load_dataset_from_hdf5_file(hdf_file_path):
  hf = h5py.File(hdf_file_path, "r")
  trainX= np.array(hf["trainX"]).astype("f8")
  ascii_train_labels = np.array(hf["trainLabels"]).astype("S65")
  trainY=np.array(hf["trainY"]).astype("int")

  devX= np.array(hf["devX"]).astype("f8")
  ascii_dev_labels = np.array(hf["devLabels"]).astype("S65")
  devY=np.array(hf["devY"]).astype("int")

  testX= np.array(hf["testX"]).astype("f8")
  ascii_test_labels = np.array(hf["testLabels"]).astype("S65")
  testY=np.array(hf["testY"]).astype("int")

  trainLabels = np.array([n.decode('unicode_escape') for n in ascii_train_labels])
  devLabels = np.array([n.decode('unicode_escape') for n in ascii_dev_labels])
  testLabels = np.array([n.decode('unicode_escape') for n in ascii_test_labels])

  print("trainX.shape: ",trainX.shape)
  print("trainY.shape: ",trainY.shape)
  print("trainLabels.shape: ",trainLabels.shape)
  print("devX.shape: ",devX.shape)
  print("devY.shape: ",devY.shape)
  print("devLabels.shape: ",devLabels.shape)
  print("testX.shape: ",testX.shape)
  print("testY.shape: ",testY.shape)
  print("testLabels.shape: ",testLabels.shape)

  return trainX, trainY, trainLabels, devX,devY,devLabels,testX,testY,testLabels
```

```
trainX, trainY, trainLabels, devX,devY,devLabels,testX,testY,testLabels=load_datase
```

```
    trainX.shape:  (603, 224, 224, 3)
    trainY.shape:  (603, 7)
    trainLabels.shape:  (603,)
    devX.shape:  (75, 224, 224, 3)
    devY.shape:  (75, 7)
```

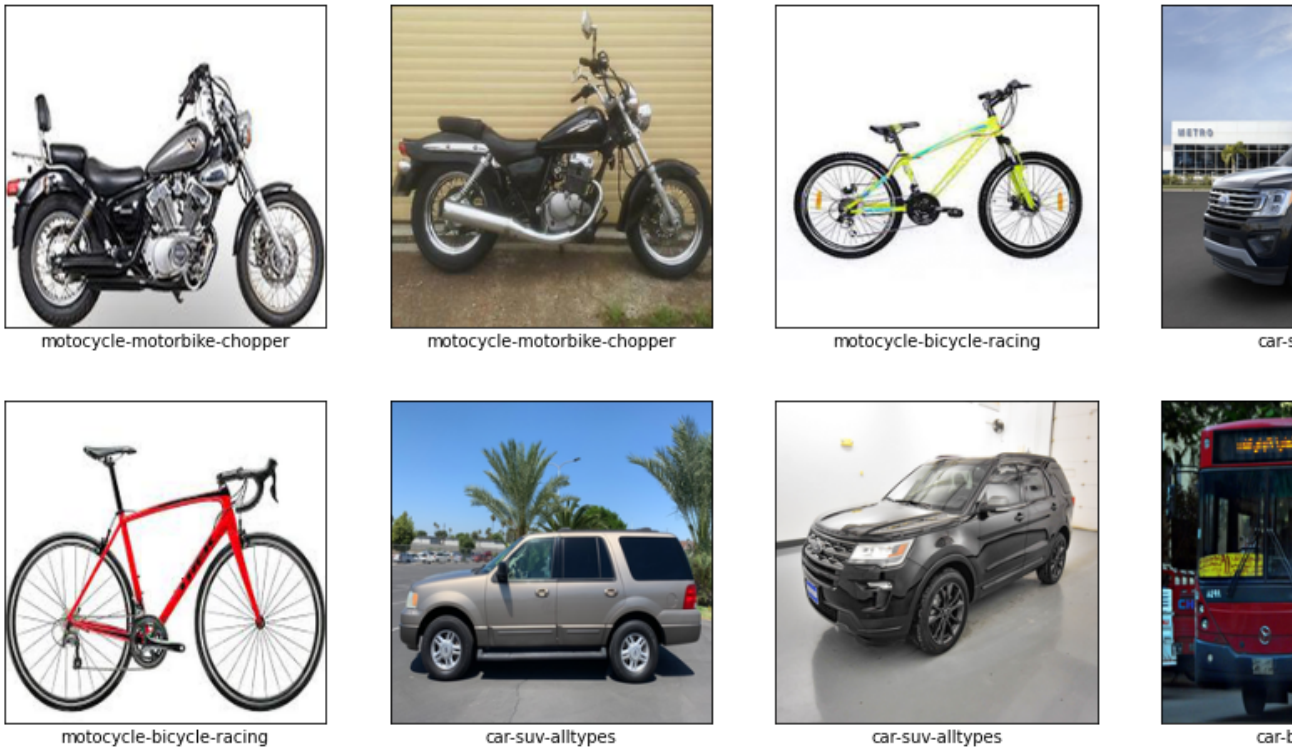
```
IMAGE_SIZE=trainX.shape[1]
print(IMAGE_SIZE)
```

```
224
```

```
number_of_classes=np.unique(trainLabels).size
```

# Dataset Visualization

```
anis_koubaa_udemy_computer_vision_lib.plot_sample_from_dataset(trainX, trainLabels,
```




```
anis_koubaa_udemy_computer_vision_lib.plot_sample_from_dataset(devX, devLabels,rows
```




```
anis_koubaa_udemy_computer_vision_lib.plot_sample_from_dataset(testX, testLabels,ro
```


# Training Configuration

## Data Augmentation

```
# initialize the training data augmentation object
trainAug = ImageDataGenerator(
  rotation_range=20)
  #fill_mode="nearest")
  #brightness_range=[0.2,1.0])
  #horizontal_flip=True)


# load the network, ensuring the head FC layer sets are left
# off
if (first_time_training==True):
    print('training for first time')
    baseModel = MobileNetV2(weights="imagenet", include_top=False, input_shape=(IMA
```

```
training for first time
```

```
    # construct the head of the model that will be placed on top of the
    # the base model
    headModel = baseModel.output
    headModel = AveragePooling2D(pool_size=(4, 4))(headModel)
    headModel = Flatten(name="flatten")(headModel)
    headModel = Dense(128, activation="relu")(headModel)
    headModel = BatchNormalization()(headModel)
    headModel = Dense(64, activation="relu")(headModel)
    headModel = Dropout(0.5)(headModel)
    headModel = BatchNormalization()(headModel)
```

| | | | |
|---|---|---|---|
| block_13_depthwise_BN (BatchNor | (None, 7, 7, 576) | 2304 | block_13_dept |
| block_13_depthwise_relu (ReLU) | (None, 7, 7, 576) | 0 | block_13_dept |
| block_13_project (Conv2D) | (None, 7, 7, 160) | 92160 | block_13_dept |
| block_13_project_BN (BatchNorma | (None, 7, 7, 160) | 640 | block_13_proj |
| block_14_expand (Conv2D) | (None, 7, 7, 960) | 153600 | block_13_proj |
| block_14_expand_BN (BatchNormal | (None, 7, 7, 960) | 3840 | block_14_expa |
| block_14_expand_relu (ReLU) | (None, 7, 7, 960) | 0 | block_14_expa |
| block_14_depthwise (DepthwiseCo | (None, 7, 7, 960) | 8640 | block_14_expa |
| block_14_depthwise_BN (BatchNor | (None, 7, 7, 960) | 3840 | block_14_dept |
| block_14_depthwise_relu (ReLU) | (None, 7, 7, 960) | 0 | block_14_dept |
| block_14_project (Conv2D) | (None, 7, 7, 160) | 153600 | block_14_dept |
| block_14_project_BN (BatchNorma | (None, 7, 7, 160) | 640 | block_14_proj |
| block_14_add (Add) | (None, 7, 7, 160) | 0 | block_13_proj<br>block_14_proj |
| block_15_expand (Conv2D) | (None, 7, 7, 960) | 153600 | block_14_add[ |
| block_15_expand_BN (BatchNormal | (None, 7, 7, 960) | 3840 | block_15_expa |
| block_15_expand_relu (ReLU) | (None, 7, 7, 960) | 0 | block_15_expa |
| block_15_depthwise (DepthwiseCo | (None, 7, 7, 960) | 8640 | block_15_expa |
| block_15_depthwise_BN (BatchNor | (None, 7, 7, 960) | 3840 | block_15_dept |
| block_15_depthwise_relu (ReLU) | (None, 7, 7, 960) | 0 | block_15_dept |
| block_15_project (Conv2D) | (None, 7, 7, 160) | 153600 | block_15_dept |
| block_15_project_BN (BatchNorma | (None, 7, 7, 160) | 640 | block_15_proj |
| block_15_add (Add) | (None, 7, 7, 160) | 0 | block_14_add[<br>block_15_proj |
| block_16_expand (Conv2D) | (None, 7, 7, 960) | 153600 | block_15_add[ |
| block_16_expand_BN (BatchNormal | (None, 7, 7, 960) | 3840 | block_16_expa |
| block_16_expand_relu (ReLU) | (None, 7, 7, 960) | 0 | block_16_expa |
| block_16_depthwise (DepthwiseCo | (None, 7, 7, 960) | 8640 | block_16_expa |
| block_16_depthwise_BN (BatchNor | (None, 7, 7, 960) | 3840 | block_16_dept |
| block_16_depthwise_relu (ReLU) | (None, 7, 7, 960) | 0 | block_16_dept |
| block_16_project (Conv2D) | (None, 7, 7, 320) | 307200 | block_16_dept |
| block_16_project_BN (BatchNorma | (None, 7, 7, 320) | 1280 | block_16_proj |

| | | | |
|---|---|---|---|
| Conv_1 (Conv2D) | (None, 7, 7, 1280) | 409600 | block_16_proj |
| Conv_1_bn (BatchNormalization) | (None, 7, 7, 1280) | 5120 | Conv_1[0][0] |
| out_relu (ReLU) | (None, 7, 7, 1280) | 0 | Conv_1_bn[0][ |
| average_pooling2d (AveragePooli | (None, 1, 1, 1280) | 0 | out_relu[0][0 |
| flatten (Flatten) | (None, 1280) | 0 | average_pooli |
| dense (Dense) | (None, 128) | 163968 | flatten[0][0] |
| batch_normalization (BatchNorma | (None, 128) | 512 | dense[0][0] |
| dense_1 (Dense) | (None, 64) | 8256 | batch_normali |
| dropout (Dropout) | (None, 64) | 0 | dense_1[0][0] |
| batch_normalization_1 (BatchNor | (None, 64) | 256 | dropout[0][0] |
| dense_2 (Dense) | (None, 7) | 455 | batch_normali |

Total params: 2,431,431
Trainable params: 173,063
Non-trainable params: 2,258,368

```
#disable this instruction if you train for the first time
```