UNIVERSITATEA POLITEHNICA DIN TIMIŞOARA
FACULTATEA DE ELECTRONICĂ, TELECOMUNICAŢII ŞI TEHNOLOGII INFORMAȚIONALONALE
SPECIALIZAREA ELECTRONICĂ APLICATĂ

PROIECT TEHNIC LA

DISCIPLINA
„Proiect Software”

Detecție numere înmatriculare

Întocmit,

Nastasia Elena NIȚU, EA2, sgr. 2.3

Coordonator,
Prof. Ricman Radu

TIMIŞOARA
2022

Cuprins

1. Descriere temă proiect

Proiectul presupune detectarea numerelor de înmatriculare pe baza unui fișier imagine, dar și în timp real, folosind o resursă de tip cameră video.

Folosind concepte ca și *Deep Learning*, *Transfer Learning* și *Image Processing*, proiectul realizează funcția de detecție a numerelor de înmatriculare.
Pe viitor, proiectul va constitui baza pentru un un alt proiect mai amplu, ce va folosi *OCR* și *DataBases* pentru a realiza o aplicație de detecție automată a urmaritorilor din trafic. Detecția se va face în timp real, folosind camera video conținută de un sistem *Raspberry Pi*.

De asemenea, proiectul poate constitui baza pentru alte aplicații, precum sisteme de monitorizare a traficului, a semafoarelor dintr-o localitate, monitorizare a unei parcari inteligente ș.a.

2. Mediul în care a fost realizat proiectul

PYTHON

Pentru realizarea proiectului, s-a folosit limbajul de programare *Python*, versiunea *3.10*.

Python este un limbaj de programare dinamic multi-paradigmă, creat în 1989 de programatorul olandez *Guido van Rossum*. Python este un limbaj multifuncțional folosit de exemplu de către companii ca *Google* sau *Yahoo*! pentru programarea aplicațiilor web, însă există și o serie de aplicații științifice sau de divertisment programate parțial sau în întregime în Python. Popularitatea în creștere, dar și puterea limbajului de programare Python au dus la adoptarea sa ca limbaj principal de dezvoltare de către programatori specializați și chiar și la predarea limbajului în unele medii universitare. Din aceleași motive, multe sisteme bazate pe *Unix*, inclusiv *Linux*, *BSD* și *Mac OS X* includ din start interpretatorul *CPython*.

Python pune accentul pe curățenia și simplitatea codului, iar sintaxa sa le permite dezvoltatorilor să exprime unele idei programatice într-o manieră mai clară și mai concisă decât în alte limbaje de programare, precum C. În ceea ce privește paradigma de programare, Python poate servi ca limbaj pentru software de tipul object-oriented, dar permite și programarea imperativă, funcțională sau procedurală. Sistemul de tipizare este dinamic iar administrarea memoriei decurge automat prin intermediul unui serviciu „gunoier” (*garbage collector*). Alt avantaj al limbajului este existența unei ample biblioteci standard de metode.

De ce Python pentru Deep Learning?

Python este un instrument perfect pentru *Deep Learning*, deoarece are atributul de a fi un limbaj de programare de uz general, fiind ușor de utilizat atunci când vine vorba de calculul analitic și cantitativ.

Este foarte ușor de înțeles, dinamic, având un sprijin uriaș al comunității și o gamă largă de biblioteci pentru diferite scopuri, cum ar fi *Numpy*, *Seaborn*, *Matplotlib*, *Pandas* și *Scikit-learn*.

Python oferă cod concis și ușor de citit. În timp ce algoritmii complecși și fluxurile de lucru versatile stau în spatele învățării automate și *AI*, simplitatea lui Python le permite dezvoltatorilor să scrie sisteme fiabile. Dezvoltatorii pot depune toate eforturile pentru a rezolva o problemă *ML* în loc să se concentreze pe nuanțele tehnice ale limbajului.

JUPYTER

Ca mediu de dezvoltare, am folosit *Jupyter Notebook*.

Jupyter Notebook (în trecut, *IPython Notebooks*) este un mediu computațional interactiv web pentru crearea de documente notebook.

Un document Jupyter Notebook conține o listă ordonată de celule de intrare/ieșire care pot conține cod, text (folosind *Markdown*), operații matematice, diagrame și conținut media. Sub interfață, un notebook este un document *JSON*, care se termină de obicei cu extensia „.ipynb”.

Jupyter Notebook se poate conecta la multe nuclee pentru a permite programarea în diferite limbi. Un nucleu Jupyter este un program responsabil pentru gestionarea diferitelor tipuri de solicitări (precum execuția codului) și furnizarea unui răspuns. Spre deosebire de multe alte interfețe asemănătoare notebook-urilor, în Jupyter, nucleele nu știu că sunt atașate la un anumit document și pot fi conectate la mai mulți clienți simultan. În mod implicit, Jupyter Notebook este livrat cu nucleul *IPython*.

JupyterLab este o interfață de utilizator mai nouă pentru Project Jupyter. Oferă elementele de bază ale notebook-ului Jupyter clasic (notebook, terminal, editor de text, browser de fișiere, ieșiri bogate etc.) într-o interfață de utilizator flexibilă.

3. Descrierea aplicației software

Înainte de realizarea propriu-zisă a codului, am început cu faza de setup, în care am creat directorul necesar, un virtual environment, am facut upgrade la utilitarul *PIP* și am instalat pachetul *ipykernel*. Am asociat un fișier de tip Jupyer Notebook cu environment-ul creat, am pornit jupyter și am setat kernel-ul folosit ca fiind virtual environment-ul creat.

În final, am stabilit anumite constante pentru a determina existența unui cod cât mai clar, concis și curat. De asemenea, am descărcat un *Dataset* gratuit de pe site-ul

kaggle.com, Dataset numit *Car License Plate Detection*. Acest Dataset l-am împărțit în două categorii, antrenare și testare (411 imagini pentru antrenare și 22 pentru testare).

În pasul 1 al proiectului, am clonat un repository existent, deoarece dorim folosirea unui model existent preantrenat, pe care îl adaptăm cazului nostru. Această abordare este cunoscută sub numele de *Transfer Learning*.

Urmează instalarea *Tensorflow Object Detection* și rularea unui script de verificare pentru a ne asigura că avem minimul de pachete necesare pentru rularea codului în continuare.

După descărcarea modelului preantrenat, în pasul 2 creăm un Label Map. În cazul de față, avem un singur label, numărul de înmatriculare.

În pasul 3, ne ocupăm de crearea unor fișiere de tip Tensorflow records, necesare pentru configurarea și reantrenarea modelului.

Deoarece modelul descărcat este sub forma unui config file, în pasul 4 ne ocupăm de copierea acestui fișier de configurare, iar, în pasul 5, îl reconfigurăm pentru cazul nostru de utilizare.

În pasul 6, generăm comanda de antrenare a modelului, pe care o vom folosi într-o fereastră de cmd, pentru a vizualiza mai bine progresul procesului de antrenare. Am antrenat modelul folosind un număr de 10000 de pași.

La fiecare 1000 de pași realizați, se creează un *checkpoint*. Vom folosi ultimul checkpoint creat pentru a detecta numere de înmatriculare folosind imagini (pasul 8), dar și în timp real (pasul 9), cu flux de date de la o cameră video.

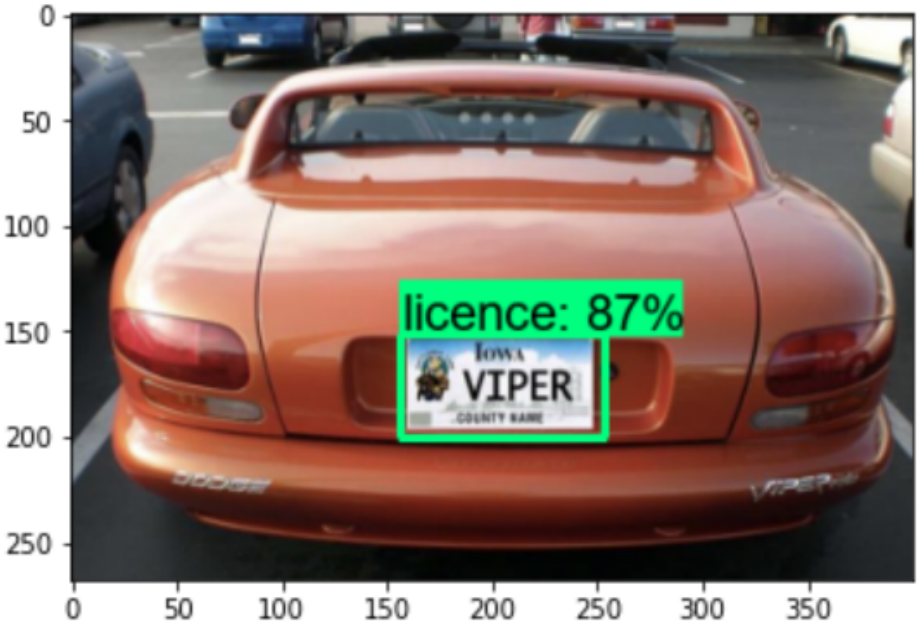
Implementarea codului a fost simplificată prin folosirea unor pachete și librării existente, precum librăriile *os* (funcții pentru interacțiunea cu sistemul de operare), *wget* (descărcare fișiere), *tensorflow* (antrenare model), *tensorflow-gpu* (accesare resurse hardware), *protobuf* (pentru utilizarea unui format stabilit de Google), *matplotlib* (realizare grafice), *Pillow* (procesare de imagine), *pyyaml* (parser), *object_detection* (detecție imagine bazată pe Tensorflow), *pandas* (analiză de date), *opencv-python* (procesare de imagine), *gin-config* (configurare potrivită pentru Machine Learning), și *numpy* (operații bazate pe array-uri).

Modelul preantrenat a fost preluat din *https://github.com/tensorflow/models*, iar baza de date a fost preluată din *https://www.kaggle.com/datasets/andrewmvd/car-plate-detection?resource=download*.

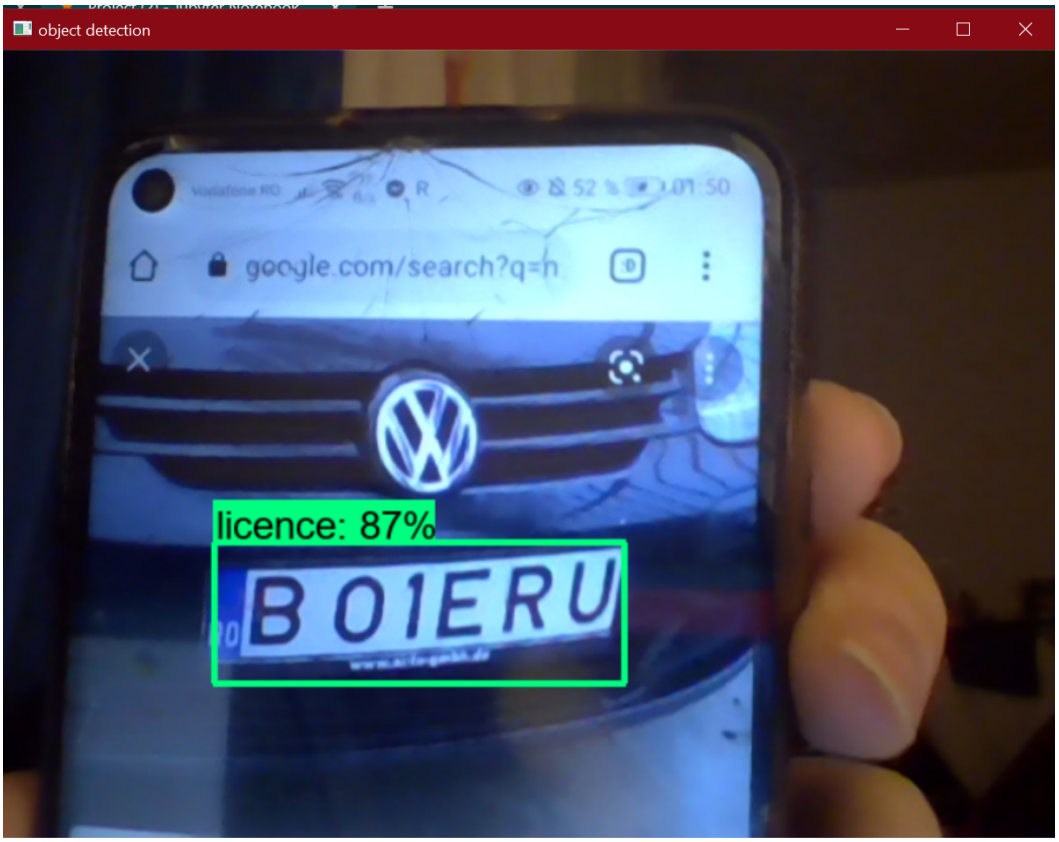
Conceptele de bază folosite au fost *Deep Learning* (concept de machine learning, ce folosește rețele neuronale cu cel puțin 3 nivele), *Transfer Learning* (concept de Machine Learning ce e folosește de aplicarea unor instrumente realizate anterior și adaptarea acestora pentru cazuri noi) și *Image Processing* (aplicarea unor operații asurpra unor imagini pentru a îmbunătăți imaginile sau pentru a obține informații pe baza acestora).

4. Rezultate rulare

a) Detecție pe baza unei imagini



b) Detecție în timp real



5. Codul sursă al aplicației

0. Setup initial

```
import os
```

```
CUSTOM_MODEL_NAME = 'my_ssd_mobnet'
PRETRAINED_MODEL_NAME = 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8'
PRETRAINED_MODEL_URL =
'http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobi
lenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz'
TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
LABEL_MAP_NAME = 'label_map.pbtxt'
```

```
paths = {
    'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),
    'SCRIPTS_PATH': os.path.join('Tensorflow','scripts'),
    'APIMODEL_PATH': os.path.join('Tensorflow','models'),
    'ANNOTATION_PATH': os.path.join('Tensorflow', 'workspace','annotations'),
    'IMAGE_PATH': os.path.join('Tensorflow', 'workspace','images'),
    'MODEL_PATH': os.path.join('Tensorflow', 'workspace','models'),
    'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow',
'workspace','pre-trained-models'),
    'CHECKPOINT_PATH': os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME),
    'OUTPUT_PATH': os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME, 'export'),
    'TFJS_PATH':os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME, 'tfjsexport'),
    'TFLITE_PATH':os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME, 'tfliteexport'),
    'PROTOC_PATH':os.path.join('Tensorflow','protoc')
}
```

```
files = {
    'PIPELINE_CONFIG':os.path.join('Tensorflow', 'workspace','models',
CUSTOM_MODEL_NAME, 'pipeline.config'),
    'TF_RECORD_SCRIPT': os.path.join(paths['SCRIPTS_PATH'],
TF_RECORD_SCRIPT_NAME),
    'LABELMAP': os.path.join(paths['ANNOTATION_PATH'], LABEL_MAP_NAME)
}
```

```
for path in paths.values():
```

```
    if not os.path.exists(path):
        if os.name == 'posix':
            !mkdir -p {path}
        if os.name == 'nt':
            !mkdir {path}
```

1. Download Modele preantrenate TensorFlow Instalare TFOD

```
# Clonam un repository existent deoarece folosim un model existent
preantrenat pe care il vom adapta cazului de fata
```

```
if os.name=='nt':
    !pip install wget
    import wget
```

```
if not os.path.exists(os.path.join(paths['APIMODEL_PATH'], 'research',
'object_detection')):
    !git clone https://github.com/tensorflow/models {paths['APIMODEL_PATH']}
```

```
# Instalare Tensorflow Object Detection
if os.name=='posix':
    !apt-get install protobuf-compiler
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto
--python_out=. && cp object_detection/packages/tf2/setup.py . && python -m
pip install .

if os.name=='nt':

url="https://github.com/protocolbuffers/protobuf/releases/download/v3.15.6/pr
otoc-3.15.6-win64.zip"
    wget.download(url)
    !move protoc-3.15.6-win64.zip {paths['PROTOC_PATH']}
    !cd {paths['PROTOC_PATH']} && tar -xf protoc-3.15.6-win64.zip
    os.environ['PATH'] += os.pathsep +
os.path.abspath(os.path.join(paths['PROTOC_PATH'], 'bin'))
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto
--python_out=. && copy object_detection\\packages\\tf2\\setup.py setup.py &&
python setup.py build && python setup.py install
    !cd Tensorflow/models/research/slim && pip install -e .
```

```
# Verificare pachete instalate
!pip list
```

```
VERIFICATION_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research',
'object_detection', 'builders', 'model_builder_tf2_test.py')
# Verificare instalare pachete necesare
!python {VERIFICATION_SCRIPT}
```

```
!pip install tensorflow==2.4.1 tensorflow-gpu==2.4.1 --upgrade
```

```
!pip uninstall protobuf matplotlib -y
!pip install protobuf matplotlib==3.2
```

```
!pip install Pillow
```

```
!pip install pyyaml
```

```
!pip list
```

```
import object_detection
```

```
# download model preantrenat
if os.name =='posix':
    !wget {PRETRAINED_MODEL_URL}
    !mv {PRETRAINED_MODEL_NAME+'.tar.gz'} {paths['PRETRAINED_MODEL_PATH']}
    !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf
{PRETRAINED_MODEL_NAME+'.tar.gz'}
if os.name == 'nt':
    wget.download(PRETRAINED_MODEL_URL)
    !move {PRETRAINED_MODEL_NAME+'.tar.gz'} {paths['PRETRAINED_MODEL_PATH']}
    !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf
{PRETRAINED_MODEL_NAME+'.tar.gz'}
```

2. Creare Label Map

```
labels = [{'name':'licence', 'id':1}]

with open(files['LABELMAP'], 'w') as f:
    for label in labels:
        f.write('item { \n')
        f.write('\tname:\'{}\'\n'.format(label['name']))
        f.write('\tid:{}\n'.format(label['id']))
        f.write('}\n')
```

3. Creare TensorFlow records

```
if not os.path.exists(files['TF_RECORD_SCRIPT']):
```

```
   !git clone https://github.com/nicknochnack/GenerateTFRecord
{paths['SCRIPTS_PATH']}
```

```
!pip install pandas
```

```
!python {files['TF_RECORD_SCRIPT']} -x {os.path.join(paths['IMAGE_PATH'],
'train')} -l {files['LABELMAP']} -o {os.path.join(paths['ANNOTATION_PATH'],
'train.record')}
!python {files['TF_RECORD_SCRIPT']} -x {os.path.join(paths['IMAGE_PATH'],
'test')} -l {files['LABELMAP']} -o {os.path.join(paths['ANNOTATION_PATH'],
'test.record')}
```

4. Copie Model Config in Training Folder

```
# modelul descarcat este sub forma unui config file ce contine toate
detaliile referitoare la model

if os.name =='posix':
   !cp {os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME,
'pipeline.config')} {os.path.join(paths['CHECKPOINT_PATH'])}
if os.name == 'nt':
   !copy {os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME,
'pipeline.config')} {os.path.join(paths['CHECKPOINT_PATH'])}
```

5. Update Config pentru folosire Transfer Learning

```
import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format
```

```
# copiem modelul si configuram caile relative si alti parametri
config = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
```

```
config
```

```
pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "r") as f:
   proto_str = f.read()
   text_format.Merge(proto_str, pipeline_config)
```

```
pipeline_config.model.ssd.num_classes = len(labels)
pipeline_config.train_config.batch_size = 4
```

```
pipeline_config.train_config.fine_tune_checkpoint =
os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME,
'checkpoint', 'ckpt-0')
pipeline_config.train_config.fine_tune_checkpoint_type = "detection"
pipeline_config.train_input_reader.label_map_path= files['LABELMAP']
pipeline_config.train_input_reader.tf_record_input_reader.input_path[:] =
[os.path.join(paths['ANNOTATION_PATH'], 'train.record')]
pipeline_config.eval_input_reader[0].label_map_path = files['LABELMAP']
pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[:] =
[os.path.join(paths['ANNOTATION_PATH'], 'test.record')]
```

```
config_text = text_format.MessageToString(pipeline_config)
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "wb") as f:
    f.write(config_text)
```

6. Antrenare model

```
TRAINING_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research',
'object_detection', 'model_main_tf2.py')
```

```
command = "python {} --model_dir={} --pipeline_config_path={}
--num_train_steps=10000".format(TRAINING_SCRIPT,
paths['CHECKPOINT_PATH'],files['PIPELINE_CONFIG'])
```

```
# printam comanda de antrenare pentru a o folosi intr-un cmd window, pentru a
vedea progresul de antrenare si alte detalii in timp real
print(command)
```

```
!pip install opencv-python
```

```
!pip install gin-config
```

```
!pip install tensorflow-addons
```

```
!{command}
```

8. Incarcare model din checkpoint

```
import os
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
from object_detection.utils import config_util
```

```
# Salvare resurse GPU
gpus = tf.config.list_physical_devices('GPU')
if gpus:
   try:
       tf.config.experimental.set_virtual_device_configuration(
           gpus[0],
[tf.config.experimental.VirtualDeviceConfiguration(memory_limit=5120)])
   except RunTimeError as e:
       print(e)
```

```
# Load pipeline config si construire model de detectie
configs =
config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
detection_model = model_builder.build(model_config=configs['model'],
is_training=False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'],
'ckpt-11')).expect_partial()

@tf.function
def detect_fn(image):
   image, shapes = detection_model.preprocess(image)
   prediction_dict = detection_model.predict(image, shapes)
   detections = detection_model.postprocess(prediction_dict, shapes)
   return detections
```

9. Detectie pe baza unei imagini

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

```
category_index =
label_map_util.create_category_index_from_labelmap(files['LABELMAP'])
```

```
IMAGE_PATH = os.path.join(paths['IMAGE_PATH'], 'Cars411.png')
```

```
img = cv2.imread(IMAGE_PATH)
image_np = np.array(img)

input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0),
dtype=tf.float32)
```

```
detections = detect_fn(input_tensor)

num_detections = int(detections.pop('num_detections'))
detections = {key: value[0, :num_detections].numpy()
              for key, value in detections.items()}
detections['num_detections'] = num_detections

# clasele de detectie sunt de tip int
detections['detection_classes'] =
detections['detection_classes'].astype(np.int64)

label_id_offset = 1
image_np_with_detections = image_np.copy()

viz_utils.visualize_boxes_and_labels_on_image_array(
            image_np_with_detections,
            detections['detection_boxes'],
            detections['detection_classes']+label_id_offset,
            detections['detection_scores'],
            category_index,
            use_normalized_coordinates=True,
            max_boxes_to_draw=5,
            min_score_thresh=.8,
            agnostic_mode=False)

plt.imshow(cv2.cvtColor(image_np_with_detections, cv2.COLOR_BGR2RGB))
plt.show()
```

```
detections.keys()
```

10. Detectie in timp real folosind un webcam

```
!pip uninstall opencv-python-headless -y
cap = cv2.VideoCapture(0)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

while cap.isOpened():
    ret, frame = cap.read()
    image_np = np.array(frame)

    input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0),
dtype=tf.float32)
    detections = detect_fn(input_tensor)

    num_detections = int(detections.pop('num_detections'))
    detections = {key: value[0, :num_detections].numpy()
```

```
                    for key, value in detections.items()}
    detections['num_detections'] = num_detections

    # detection_classes should be ints.
    detections['detection_classes'] =
detections['detection_classes'].astype(np.int64)

    label_id_offset = 1
    image_np_with_detections = image_np.copy()

    viz_utils.visualize_boxes_and_labels_on_image_array(
                image_np_with_detections,
                detections['detection_boxes'],
                detections['detection_classes']+label_id_offset,
                detections['detection_scores'],
                category_index,
                use_normalized_coordinates=True,
                max_boxes_to_draw=5,
                min_score_thresh=.8,
                agnostic_mode=False)

    try:
        text, region = ocr_it(image_np_with_detections, detections,
detection_threshold, region_threshold)
        save_results(text, region, 'realtimeresults.csv', 'Detection_Images')
    except:
        pass

    cv2.imshow('object detection',  cv2.resize(image_np_with_detections,
(800, 600)))

    if cv2.waitKey(0) & 0xFF == ord('q'):
        cap.release()
        cv2.destroyAllWindows()
        break
```

6. Bibliografie

1. https://en.wikipedia.org/wiki/Project_Jupyter, accesat la data de 28.03.2022
2. https://www.edureka.co/blog/deep-learning-with-python/, accesat la data de 28.03.2022
3. https://github.com/tensorflow/models/, accesat la data de 28.03.2022
4. https://www.kaggle.com/datasets/andrewmvd/car-plate-detection?resource=download, accesat la data de 28.03.2022
5. https://docs.python.org/3/library/os.html, accesat la data de 28.03.2022
6. https://pypi.org/project/tensorflow-gpu/, accesat la data de 28.03.2022
7. https://pypi.org/project/protobuf/, accesat la data de 28.03.2022
8. https://pandas.pydata.org/, accesat la data de 28.03.2022
9. https://pypi.org/project/gin-config/, accesat la data de 28.03.2022