DAFTAR PUSTAKA

Akhir, T., Umar, Y., Nrp, H., Mardi, S., Nugroho, S., Rachmadi, R. F., Teknik, D., Fakultas, K., & Elektro, T. (2020). *Deteksi Penggunaan Helm Pada Pengendara*.

Anka, A. (2020). *YOLO v4: Optimal Speed & Accuracy for object detection*. https://towardsdatascience.com/yolo-v4-optimal-speed-accuracy-for-object-detection-79896ed47b50#6ac9

B.Fisher, R., Breckon, T. P., Dawson-Howe, K., Fitzgibbon, A., Robertson, C., Trucco, E., & Williams, C. K. I. (2013). *Dictionary of Computer Vision and Image Processing, 2nd Edition*.

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*. http://arxiv.org/abs/2004.10934

Budiarjo, D. D. (2020). *IMPLEMENTASI SISTEM CERDAS PADA OTOMATISASI PENDETEKSIAN JENIS KENDARAAN DI JALAN RAYA* [Universitas Negeri Semarang]. https://repository.usm.ac.id/files/skripsi/G11A/2015/G.131.15.0065/G.131.15.0065-15-File-Komplit-20200306015622.pdf

Chethan Kumar, B., Punitha, R., & Mohana. (2020). YOLOv3 and YOLOv4: Multiple object detection for surveillance applications. *Proceedings of the 3rd International Conference on Smart Systems and Inventive Technology, ICSSIT 2020*, *Icssit*, 1316–1321. https://doi.org/10.1109/ICSSIT48917.2020.9214094

Google. (2020). *Colaboratory*. https://research.google.com/colaboratory/intl/id/faq.html

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 2261–2269. https://doi.org/10.1109/CVPR.2017.243

Islam, M. T. (2019). Traffic sign detection and recognition based on convolutional neural networks. *2019 6th IEEE International Conference on Advances in Computing, Communication and Control, ICAC3 2019*. https://doi.org/10.1109/ICAC347590.2019.9036784

Low, J. (2020). *What is Image Annotation?* Medium. https://medium.com/supahands-techblog/what-is-image-annotation-caf4107601b7. diakses

Manocha, P., Kumar, A., Khan, J. A., & Shin, H. (2019). Korean Traffic Sign Detection Using Deep Learning. *Proceedings - International SoC Design Conference 2018, ISOCC 2018*, 247–248. https://doi.org/10.1109/ISOCC.2018.8649887

Minister of Transportation. (2014). Regulation of the Minister of Transportation of the Republic of Indonesia Number Prime Minister 13 of 1993 Concerning Traffic Signs. *Regulation of the Minister of Transportation of the Republic of Indonesia Number Pm 115 of 2018*,

1–8.
Misra, D. (2020). *Mish: A Self Regularized Non-Monotonic Activation Function*. 14. https://arxiv.org/pdf/1908.08681.pdf
Morikawa, R. (2019). *24 Best Image Annotation Tools for Computer Vision*. https://lionbridge.ai/articles/image-annotation-tools-for-computer-vision/
Mulyanto, A., Borman, R. I., Prasetyawan, P., Jatmiko, W., Mursanto, P., & Sinaga, A. (2020). Indonesian Traffic Sign Recognition for Advanced Driver Assistent (ADAS) Using YOLOv4. *2020 3rd International Seminar on Research of Information Technology and Intelligent Systems, ISRITI 2020*, 520–524. https://doi.org/10.1109/ISRITI51436.2020.9315368
Nixon, M., & Aguado, A. (2019). *Feature Extraction and Image Processing for Computer Vision 4th Edition*. Academic Press.
Promlainak, S., Woraratpanya, K., Kuengwong, J., & Kuroki, Y. (2018). Thai traffic sign detection and recognition for driver assistance. *Proceeding of 2018 7th ICT International Student Project Conference, ICT-ISPC 2018*, 1–5. https://doi.org/10.1109/ICT-ISPC.2018.8523920
Qiao, K., Gu, H., Liu, J., & Liu, P. (2017). Optimization of traffic sign detection and classification based on faster R-CNN. *Proceedings - 2017 International Conference on Computer Technology, Electronics and Communication, ICCTEC 2017*, 608–611. https://doi.org/10.1109/ICCTEC.2017.00137
Rahmad, C., Rahmah, I. F., Asmara, R. A., & Adhisuwignjo, S. (2018). Indonesian traffic sign detection and recognition using color and texture feature extraction and SVM classifier. *2018 International Conference on Information and Communications Technology, ICOIACT 2018*, *2018-Janua*(c), 50–55. https://doi.org/10.1109/ICOIACT.2018.8350804
Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 779–788. https://doi.org/10.1109/CVPR.2016.91
Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 6517–6525. https://doi.org/10.1109/CVPR.2017.690
Redmon, J., & Farhadi, A. (2018). YOLO v.3. *Tech Report*, 1–6. https://pjreddie.com/media/files/papers/YOLOv3.pdf
Wang, C. (2018). Research and application of traffic sign detection and recognition based on deep learning. *Proceedings - 2018 International Conference on Robots and Intelligent System, ICRIS 2018*, 150–152. https://doi.org/10.1109/ICRIS.2018.00047
Wang, C. Y., Mark Liao, H. Y., Wu, Y. H., Chen, P. Y., Hsieh, J. W., & Yeh, I. H. (2020). CSPNet: A new backbone that can enhance

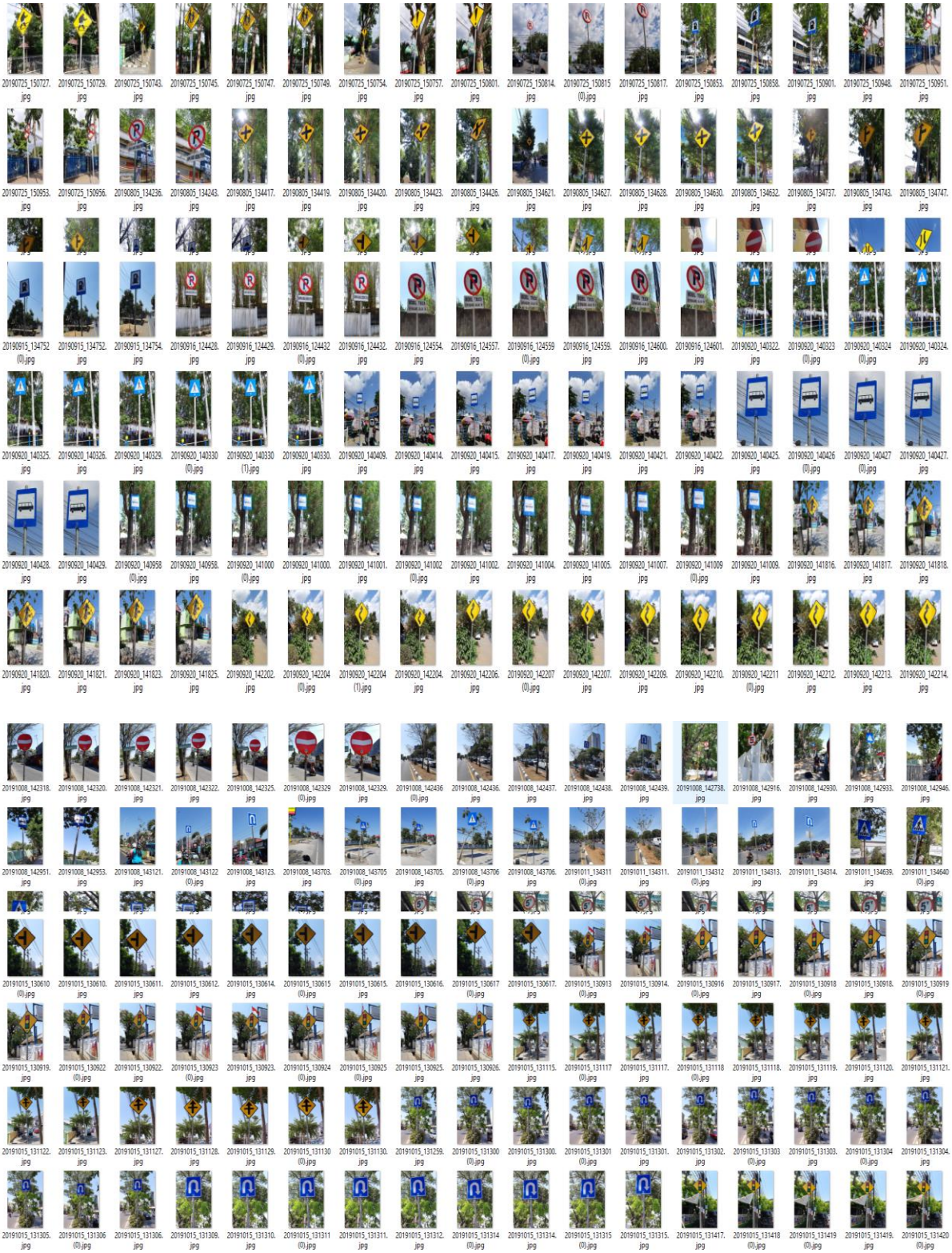
learning capability of CNN. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, *2020-June*, 1571–1580. https://doi.org/10.1109/CVPRW50498.2020.00203

Xiaobai, J. (n.d.). *A simple explanation of Yolov3 and Yolov4*. Retrieved June 24, 2021, from https://www.programmersought.com/article/13544183937/

Yu, J., Liu, H., & Zhang, H. (2019). Research on Detection and Recognition Algorithm of Road Traffic Signs. *Proceedings of the 31st Chinese Control and Decision Conference, CCDC 2019*, 1996–2001. https://doi.org/10.1109/CCDC.2019.8833426

LAMPIRAN

Data Latih



Script Training (yolov4(1).ipynb)

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

!ln -s /content/gdrive/My\ Drive/ /mydrive
!ls /mydrive

!cp /mydrive/darknet.zip ../

!unzip ../darknet.zip

%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile

!make

!cp /mydrive/yolov4/obj.zip ../

!unzip ../obj.zip -d data/

!cp /mydrive/yolov4/yolov4_custom.cfg ./cfg

!cp /mydrive/yolov4/obj.names ./data
!cp /mydrive/yolov4/obj.data  ./data

!cp /mydrive/yolov4/generate_train.py ./
!cp /mydrive/yolov4/generate_test.py ./

def download(path):
  from google.colab import files
  files.download(path)

!python generate_train.py
!python generate_test.py

!./darknet detector train data/obj.data cfg/yolov4_custom.cfg yolov4.conv.137 -
dont_show
```

Script deteksi video (YOLOv_detect_Video.ipynb)

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

!ln -s /content/gdrive/My\ Drive/ /mydrive
!ls /mydrive

!cp /mydrive/darknet.zip ../

!unzip ../darknet.zip

!ls /mydrive/yolov4

%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile

!/usr/local/cuda/bin/nvcc --version

!make

def download(path):
  from google.colab import files
  files.download(path)

!cp /mydrive/videos/4.mp4 ./

!cp /mydrive/yolov4/backup/yolov4_custom_last.weights ./

!cp /mydrive/yolov4/obj.names ./cfg
!cp /mydrive/yolov4/obj.data  ./cfg

!cp /mydrive/yolov4/obj.names ./data
!cp /mydrive/yolov4/obj.data  ./data

!cp /mydrive/yolov4/yolov4_detect_videos.cfg  ./cfg

!./darknet detector demo cfg/obj.data cfg/yolov4_detect_videos.cfg yolov4_custom_la
st.weights -dont_show 4.mp4 -i 0 -out_filename 2Khasil4.mp4

!cp -r 2Khasil4.mp4 "/content/gdrive/My Drive/"
```

Script menghitung *mean Average Precision* (mAP) (YOLOv_detect_Tutorial_measure_map (2).ipynb)

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

!ln -s /content/gdrive/My\ Drive/ /mydrive
!ls /mydrive

!cp /mydrive/darknet.zip ../

!unzip ../darknet.zip

!ls /mydrive/yolov4

%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile

!/usr/local/cuda/bin/nvcc --version

!make

def download(path):
  from google.colab import files
  files.download(path)

# define helper functions
def imShow(path):
  import cv2
  import matplotlib.pyplot as plt
  %matplotlib inline

  image = cv2.imread(path)
  height, width = image.shape[:2]
  resized_image = cv2.resize(image,(3*width, 3*height), interpolation = cv2.INTER_C
UBIC)

  fig = plt.gcf()
  fig.set_size_inches(18, 10)
  plt.axis("off")
  plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
  plt.show()
```

```
!cp /mydrive/yolov4/backup/yolov4_custom_last.weights ./

!cp /mydrive/yolov4/obj.zip ../
!unzip ../obj.zip -d data/
!cp /mydrive/yolov4/obj.names ./cfg
!cp /mydrive/yolov4/obj.data  ./cfg

!cp /mydrive/yolov4/obj.names ./data
!cp /mydrive/yolov4/obj.data  ./data

!cp /mydrive/yolov4/yolov4_detect_videos.cfg  ./cfg
!cp /mydrive/yolov4/generate_test.py ./

!python generate_test.py

!./darknet detector map data/obj.data cfg/yolov4_detect_videos.cfg yolov4_custo
m_last.weights -dont_show -ext_output < /mydrive/images.txt > result.csv
download("result.csv")
```

File obj.names

```
Peringatan Simpang Empat Prioritas (Minor)
Larangan Berhenti
Petunjuk Lokasi Putar Balik
Petunjuk Lokasi Fasilitas Pemberhentian Mobil Bus Umum
Petunjuk Lokasi Fasilitas Penyebrangan Pejalan Kaki
Larangan Parkir
Petunjuk Lokasi Balai Kesehatan, Puskesmas, Balai Pertolongan Pertama dan sejenis
Petunjuk Lokasi Masjid
Peringatan Alat Pemberi Isyarat Lalu Lintas
Peringatan Banyak Lalu Lintas Pejalan Kaki Anak Anak
Peringatan,(ditegaskan penjelasan jenis peringatan dengan papan tambahan)
Peringatan Banyak Lalu Lintas Pejalan Kaki
Larangan Memutar Balik
Petunjuk Lokasi SPBU
Larangan Masuk bagi Becak
Peringatan Persimpangan Tiga Sisi Kanan
Peringatan Persimpangan Tiga Sisi Kiri (Minor)
Larangan Masuk Bagi Kendaraan Bermotor dan Tidak Bermotor
Peringatan Jembatan, Penyempitan Bagan Jalinan Jalan Tertentu
Peringatan Persimpangan Tiga Serong Kiri (Minor)
Peringatan Persimpangan Tiga Sisi Kiri
Peringatan Persimpangan Tiga Serong Kiri
Peringatan Persimpangan Tiga Sisi Kanan (Minor)
Peringatan Tikungan Ke Kiri
Peringatan Tikungan Ke Kanan
Peringatan Persimpangan Tiga Serong Kanan (Minor)
Larangan Masuk bagi Kendaraan bermotor lebih dari 5 ton
Larangan Belok Kanan
Peringatan Simpang Empat Prioritas (Mayor)
Perintah Mengikuti ke Arah Kiri
Perintah Memasuki Jalur atau jalur yang ditunjuk
Petunjuk Lokasi Gereja
Larangan Masuk bagi Kendaraan Bermotor Roda Tunggal dengan MST sama atau lebih 8 Ton
Peringatan Banyak Tikungan dengan Tikungan Pertaama Ke Kiri
Petunjuk Lokasi Rumah Sakit
```

File obj.data

```
classes =35
train = data/train.txt
valid = data/test.txt
names = data/obj.names
backup = /mydrive/yolov4/backup/
```

Generate_train.py

```
import os

image_files = []
os.chdir(os.path.join("data", "obj"))
for filename in os.listdir(os.getcwd()):
    if filename.endswith(".jpg"):
        image_files.append("data/obj/" + filename)
os.chdir("..")
with open("train.txt", "w") as outfile:
    for image in image_files:
        outfile.write(image)
        outfile.write("\n")
    outfile.close()
os.chdir("..")
```

Generate_test.py

```
import os

image_files = []
os.chdir(os.path.join("data", "obj"))
for filename in os.listdir(os.getcwd()):
    if filename.endswith(".jpg"):
        image_files.append("data/obj/" + filename)
os.chdir("..")
with open("test.txt", "w") as outfile:
    for image in image_files:
        outfile.write(image)
        outfile.write("\n")
    outfile.close()
os.chdir("..")
```

Script *realtime* (detect_video.py)

```
import time
import tensorflow as tf
physical_devices = tf.config.experimental.list_physical_devices('GPU')
if len(physical_devices) > 0:
    tf.config.experimental.set_memory_growth(physical_devices[0], True)
from absl import app, flags, logging
from absl.flags import FLAGS
import core.utils as utils
from core.yolov4 import filter_boxes
from tensorflow.python.saved_model import tag_constants
from PIL import Image
import cv2
import numpy as np
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession

flags.DEFINE_string('framework', 'tf', '(tf, tflite, trt')
flags.DEFINE_string('weights', './checkpoints/yolov4-416',
                    'path to weights file')
flags.DEFINE_integer('size', 416, 'resize images to')
flags.DEFINE_boolean('tiny', False, 'yolo or yolo-tiny')
flags.DEFINE_string('model', 'yolov4', 'yolov3 or yolov4')
flags.DEFINE_string('video', './data/video/video.mp4', 'path to input video or set to 0 for webcam')
flags.DEFINE_string('output', None, 'path to output video')
flags.DEFINE_string('output_format', 'XVID', 'codec used in VideoWriter when saving video to file')
flags.DEFINE_float('iou', 0.45, 'iou threshold')
flags.DEFINE_float('score', 0.25, 'score threshold')
flags.DEFINE_boolean('dont_show', False, 'dont show video output')

def main(_argv):
    config = ConfigProto()
    config.gpu_options.allow_growth = True
    session = InteractiveSession(config=config)
    STRIDES, ANCHORS, NUM_CLASS, XYSCALE = utils.load_config(FLAGS)
    input_size = FLAGS.size
    video_path = FLAGS.video

    if FLAGS.framework == 'tflite':
        interpreter = tf.lite.Interpreter(model_path=FLAGS.weights)
        interpreter.allocate_tensors()
        input_details = interpreter.get_input_details()
        output_details = interpreter.get_output_details()
        print(input_details)
        print(output_details)
    else:
        saved_model_loaded = tf.saved_model.load(FLAGS.weights,tags=[tag_constants.SERVING])
        infer = saved_model_loaded.signatures['serving_default']

# begin video capture
    try:
        vid = cv2.VideoCapture(int(video_path))
    except:
        vid = cv2.VideoCapture(video_path)

    out = None

    if FLAGS.output:
        # by default VideoCapture returns float instead of int
        width = int(vid.get(cv2.CAP_PROP_FRAME_WIDTH))
        height = int(vid.get(cv2.CAP_PROP_FRAME_HEIGHT))
        fps = int(vid.get(cv2.CAP_PROP_FPS))
        codec = cv2.VideoWriter_fourcc(*FLAGS.output_format)
        out = cv2.VideoWriter(FLAGS.output, codec, fps, (width, height))

    while True:
        return_value, frame = vid.read()
```

```
if return_value:
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            image = Image.fromarray(frame)
        else:
            print('Video has ended or failed, try a different video format!')
            break

        frame_size = frame.shape[:2]
        image_data = cv2.resize(frame, (input_size, input_size))
        image_data = image_data / 255.
        image_data = image_data[np.newaxis, ...].astype(np.float32)
        start_time = time.time()

        if FLAGS.framework == 'tflite':
            interpreter.set_tensor(input_details[0]['index'], image_data)
            interpreter.invoke()
            pred = [interpreter.get_tensor(output_details[i]['index']) for i in range(len(output_details))]
            if FLAGS.model == 'yolov3' and FLAGS.tiny == True:
                boxes, pred_conf = filter_boxes(pred[1], pred[0], score_threshold=0.25,
                                                input_shape=tf.constant([input_size, input_size]))
            else:
                boxes, pred_conf = filter_boxes(pred[0], pred[1], score_threshold=0.25,
                                                input_shape=tf.constant([input_size, input_size]))
        else:
            batch_data = tf.constant(image_data)
            pred_bbox = infer(batch_data)
            for key, value in pred_bbox.items():
                boxes = value[:, :, 0:4]
                pred_conf = value[:, :, 4:]

        boxes, scores, classes, valid_detections = tf.image.combined_non_max_suppression(
            boxes=tf.reshape(boxes, (tf.shape(boxes)[0], -1, 1, 4)),
            scores=tf.reshape(
                pred_conf, (tf.shape(pred_conf)[0], -1, tf.shape(pred_conf)[-1])),
            max_output_size_per_class=50,
            max_total_size=50,
            iou_threshold=FLAGS.iou,
            score_threshold=FLAGS.score
        )
        pred_bbox = [boxes.numpy(), scores.numpy(), classes.numpy(), valid_detections.numpy()]
        image = utils.draw_bbox(frame, pred_bbox)
        fps = 1.0 / (time.time() - start_time)
        print("FPS: %.2f" % fps)
        result = np.asarray(image)
        cv2.namedWindow("result", cv2.WINDOW_AUTOSIZE)
        result = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        if not FLAGS.dont_show:
            cv2.imshow("result", result)

        if FLAGS.output:
            out.write(result)
        if cv2.waitKey(1) & 0xFF == ord('q'): break
    cv2.destroyAllWindows()

if __name__ == '__main__':
    try:
        app.run(main)
    except SystemExit:
        pass
```