LAMPIRAN

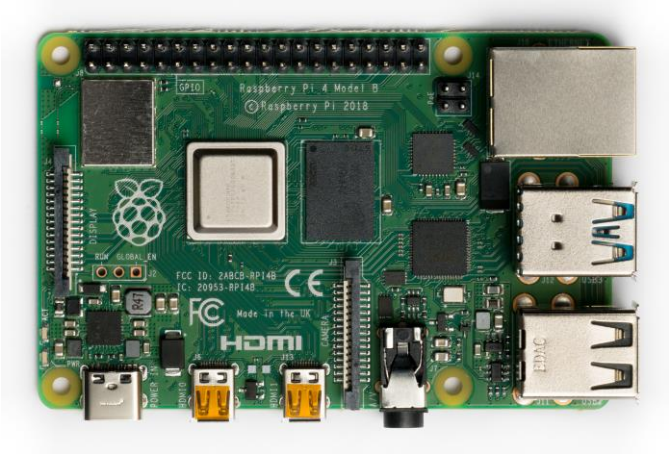
Lampiran 1 Skrip untuk *bounding box clustering*

```
# coding: utf-8
# This script is modified from https://github.com/lars76/kmeans-anchor-box

from __future__ import division, print_function

import numpy as np

def iou(box, clusters):
    """
    Calculates the Intersection over Union (IoU) between a box and k clusters.
    param:
        box: tuple or array, shifted to the origin (i. e. width and height)
        clusters: numpy array of shape (k, 2) where k is the number of clusters
    return:
        numpy array of shape (k, 0) where k is the number of clusters
    """
    x = np.minimum(clusters[:, 0], box[0])
    y = np.minimum(clusters[:, 1], box[1])
    if np.count_nonzero(x == 0) > 0 or np.count_nonzero(y == 0) > 0:
        raise ValueError("Box has no area")

    intersection = x * y
    box_area = box[0] * box[1]
    cluster_area = clusters[:, 0] * clusters[:, 1]

    iou_ = np.true_divide(intersection, box_area + cluster_area - intersection + 1e-10)
    # iou_ = intersection / (box_area + cluster_area - intersection + 1e-10)

    return iou_


def avg_iou(box, clusters):
    """
    Calculates the average Intersection over Union (IoU) between a numpy array of box and k clusters.
    param:
        box: numpy array of shape (r, 2), where r is the number of rows
        clusters: numpy array of shape (k, 2) where k is the number of clusters
    return:
        average IoU as a single float
    """
    return np.mean([np.max(iou(box[i], clusters)) for i in range(box.shape[0])])


def translate_box(box):
```

```
    """
    Translates all the box to the origin.
    param:
        box: numpy array of shape (r, 4)
    return:
    numpy array of shape (r, 2)
    """
    new_box = box.copy()
    for row in range(new_box.shape[0]):
        new_box[row][2] = np.abs(new_box[row][2] -
new_box[row][0])
        new_box[row][3] = np.abs(new_box[row][3] -
new_box[row][1])
    return np.delete(new_box, [0, 1], axis=1)


def kmeans(box, k, dist=np.median):
    """
    Calculates k-means clustering with the Intersection over
Union (IoU) metric.
    param:
        box: numpy array of shape (r, 2), where r is the number
of rows
        k: number of clusters
        dist: distance function
    return:
        numpy array of shape (k, 2)
    """
    rows = box.shape[0]

    distances = np.empty((rows, k))
    last_clusters = np.zeros((rows,))

    np.random.seed()

    # the Forgy method will file if the whole array contains the
same rows
    clusters = box[np.random.choice(rows, k, replace=False)]

    while True:
        for row in range(rows):
            distances[row] = 1 - iou(box[row], clusters)

        nearest_clusters = np.argmin(distances, axis=1)

        if (last_clusters == nearest_clusters).all():
            break

        for cluster in range(k):
            clusters[cluster] = dist(box[nearest_clusters ==
cluster], axis=0)

        last_clusters = nearest_clusters

    return clusters

def parse_anno(annotation_path, target_size=None):
    anno = open(annotation_path, 'r')
    result = []
```

```
    for line in anno:
        s = line.strip().split(' ')
        img_w = int(s[2])
        img_h = int(s[3])
        s = s[4:]
        box_cnt = len(s) // 5
        for i in range(box_cnt):
            x_min, y_min, x_max, y_max = float(s[i*5+1]),
float(s[i*5+2]), float(s[i*5+3]), float(s[i*5+4])
            width = x_max - x_min
            height = y_max - y_min
            assert width > 0
            assert height > 0
            # use letterbox resize, i.e. keep the original
aspect ratio
            # get k-means anchors on the resized target image
size
            if target_size is not None:
                resize_ratio = min(target_size[0] / img_w,
target_size[1] / img_h)
                width *= resize_ratio
                height *= resize_ratio
                result.append([width, height])
            # get k-means anchors on the original image size
            else:
                result.append([width, height])
    result = np.asarray(result)
    return result

def get_kmeans(anno, cluster_num=9):

    anchors = kmeans(anno, cluster_num)
    ave_iou = avg_iou(anno, anchors)

    anchors = anchors.astype('int').tolist()

    anchors = sorted(anchors, key=lambda x: x[0] * x[1])

    return anchors, ave_iou

if __name__ == '__main__':
    # target resize format: [width, height]
    target_size = [416, 416]
    annotation_path = "train.txt"
    anno_result = parse_anno(annotation_path, target_size =
target_size)
    anchors, ave_iou = get_kmeans(anno_result, 9)

    anchor_string = ''
    for anchor in anchors:
        anchor_string += '{},{}, '.format(anchor[0], anchor[1])
    anchor_string = anchor_string[:-2]

    print('anchors are:')
    print(anchor_string)
    print('the average iou is:')
    print(ave_iou)
```

Lampiran 2 Raspberry Pi 4 Model B



Lampiran 3 Kamera Raspberry Pi Rev 1.3

Lampiran 4 *Setup* alat akuisisi data tampak atas

Lampiran 5 *Setup* alat akuisisi data tampak keseluruhan

Lampiran 6 Pengajuan penggunaan layanan HPC LIPI

Berikut ini adalah hal-hal terkait pengajuan penggunaan fasilitas HPC LIPI untuk pelatihan model *deep learning* menggunakan Tensorflow.

1. Bagi pengguna nonsivitas LIPI, diharuskan untuk mendaftar terlebih dahulu agar dapat menggunakan ELSA (E-Layanan Sains). Situs ELSA dapat diakses pada https://elsa.lipi.go.id/. Adapun pedoman pendaftaran, masuk, reset *password*, dan keluar dari ELSA dapat diakses pada https://elsatur.lipi.go.id/public/uploads/layanan_file/284/sop_layanan_284.pdf.
2. Setelah selesai membuat akun dan berhasil masuk ELSA, kemudian pada menu **LAIN-LAIN → LIPI HPC** pilih layanan **Machine Learning dengan Tensorflow**, atau secara langsung dapat mengakses halaman https://elsa.lipi.go.id/layanan/index/Machine%20Learning%20dengan%20TensorFlow/284. Setelah halaman tersebut terbuka, kemudian klik **AJUKAN LAYANAN**.
3. Pada halaman pengajuan layanan, isi formulir yang tersedia dengan isian yang sesuai. Isian formulir tersebut meliputi:
   a. **Tanggal pelaksanaan layanan** diisi dengan tanggal penggunaan layanan dimulai sampai dengan tanggal penggunaan layanan berakhir. Rentang waktu penggunaan maksimal 6 bulan. Pengguna juga dapat melihat jadwal yang telah terisi dengan cara klik tombol **Lihat Jadwal** di samping kanan kolom isian.
   b. **Judul proposal** diisi dengan judul penelitian yang akan dilaksanakan, misalnya "Analisis Perbandingan Kinerja Algoritme *Object Detection* Berbasis *Deep Learning* pada Perangkat Komputasi Terbatas (Studi Kasus: Deteksi Kelainan Daun Tanaman Melon)".
   c. **Abstrak** diisi dengan abstrak penelitian yang akan dilaksanakan. Contoh abstrak adalah sebagai berikut.

> "*Object detection* merupakan salah satu metode dalam *computer vision* untuk mengidentifikasi kelas dan lokasi objek pada suatu gambar. Dibandingkan dengan

pendekatan tradisional, metode *object detection* menggunakan pendekatan *deep learning* memiliki performa yang lebih baik secara signifikan terutama dalam hal akurasi dan presisi. Namun, metode dengan pendekatan *deep learning* ini menghasilkan waktu komputasi yang tinggi dan pemakaian memori yang lebih besar. Kedua hal tersebut menjadi tantangan dalam implementasi algoritma *deep learning* pada perangkat yang memiliki kemampuan komputasi terbatas seperti Raspberry Pi. Raspberry Pi merupakan perangkat komputer yang murah (*low-cost*) dengan kapasitas memori dan kemampuan pemrosesan yang terbatas. Aspek akurasi dan efisiensi komputasi menjadi *trade-off* dalam penerapan algoritma *deep learning* pada perangkat komputasi terbatas. Oleh karena itu, penelitian ini bertujuan untuk melakukan analisis perbandingan terhadap beberapa algoritma *object detection* berbasis *deep learning* pada salah satu perangkat dengan kemampuan komputasi terbatas yaitu Raspberry Pi. Adapun objek yang dideteksi adalah penyakit pada daun tanaman melon. Melon (*Cucumis melo* L.) adalah salah satu tanaman yang sangat sensitif terhadap hama dan penyakit. Deteksi dini pada penyakit tanaman melon dapat menjadi upaya untuk mengurangi kerugian terhadap hasil produksi dan ekonomi pada sektor pertanian. Oleh karena itu, deteksi penyakit pada daun tanaman melon menjadi studi kasus dalam penelitian ini".

d. **Kata kunci** diisi dengan kata kunci pada penelitian yang akan dilaksanakan, misalnya "*Object detection*, *deep learning*, perangkat komputasi terbatas, Raspberry Pi, melon, *Cucumis melo* L."
e. **Tujuan** diisi dengan tujuan penelitian, misalnya,

"1. Mengimplementasikan algoritme *object detection* berbasis *deep learning* pada Raspberry Pi untuk mendeteksi penyakit daun tanamana melon.
2. Melakukan analisis perbandingan terhadap kinerja algoritme *object detection* berbasis *deep learning* dalam implementasinya pada perangkat komputasi terbatas".

f. **Metodologi** diisi dengan metode penelitian yang akan dikerjakan. Contoh metode penelitian adalah sebagai berikut.

"Dataset yang digunakan meliputi data citra sekumpulan daun tanaman melon. Proses pengambilan citra dilakukan dengan memotret daun dan kumpulan daun tanaman melon menggunakan kamera Raspberry Pi. Dataset yang diperoleh digunakan untuk proses *object detection* dengan

algoritme deep learning Faster R-CNN, YOLOv3, dan SSD. Pada penelitian ini, kelas daun yang dideteksi dibagi menjadi dua kategori, yaitu abnormal dan normal. Model diharapkan dapat mendeteksi keberadaan objek daun pada suatu citra dan menentukan kelas daun yang sesuai (abnormal atau normal). Setelah model *deep learning* diperoleh melalui proses pelatihan, kemudian dilakukan evaluasi model dan analisis kinerja masing-masing algoritme (Faster R-CNN, YOLOv3, dan SSD) pada Raspberry Pi. Secara umum, tahapan penelitian terdiri dari akuisisi data, anotasi data, pelatihan model *object detection*, evaluasi model (menghitung mAP), implementasi model pada Raspberry Pi, dan analisis perbandingan kinerja dari algoritme-algoritme *object detection* yang digunakan."

g. **Daftar anggota** diisi dengan nama peneliti yang terlibat, yakni dapat diisi dengan nama mahasiswa beserta ketua dan anggota komisi pembimbing.
h. **Kapasistas penyimpanan (GB)** dapat diisi "100" atau sesuai kebutuhan.
i. **Jumlah Core CPU** dapat diisi "28" atau sesuai kebutuhan.
j. **Jumlah GPU** dapat diisi "1" atau sesuai kebutuhan.
k. **Perangkat lunak** diisi perangkat lunak yang dibutuhkan saat penelitian, misalnya "Python3.7, Anaconda3" atau sesuai kebutuhan.
l. **Sumber dana** diisi dengan sumber dana penelitian. Jika penelitian didanai oleh lembaga tertentu, maka isi dengan lembaga pendana tersebut. Jika tidak ada pendana dari luar, atau dengan kata lain menggunakan biaya pribadi, maka dapat diisi dengan "Pribadi".
m. **Catatan** dapat diisi dengan catatan bagi pihak ELSA jika diperlukan.
n. **Keluaran** dapat diisi dengan jumlah keluaran pada kolom sebelah kiri (misalnya diisi "1"), dan jenis keluaran pada kolom sebelah kanan (misalnya dipilih "Tesis").

4. Setelah formulir selesai diisi, selanjutnya klik **AJUKAN**.
5. Pengajuan layanan akan ditinjau dalam beberapa saat/hari. Untuk melihat status layanan, pada menu di situs ELSA LIPI pilih **PROFIL → TRANSAKSI**.

Lampiran 7 Bekerja pada *node* GPU secara interaktif di HPC LIPI

1. Untuk mengakses HPC LIPI, pada *command line* (cmd) komputer/laptop jalankan perintah **`ssh username@masuk.hpc.lipi.go.id`**, atau menggunakan aplikasi SSH *client* seperti PuTTY, SmarTTY, dan sebagainya. Saat melakukan akses pertama kali, pengguna akan diminta mengganti *password*.
2. Untuk bekerja pada *node* GPU secara interaktif, berikut adalah langkah-langkahnya.
   a. Buka aplikasi *text editor*, kemudian tulis skrip berikut ini.

```
#!/bin/bash

#PBS -N your_job_name
#PBS -I
#PBS -q gpu
#PBS -l nodes=1:ppn=72
#PBS -l walltime=6:00:00
#PBS -m ae
#PBS -M your@email.address

cd $PBS_O_WORKDIR
```

b. Simpan skrip tersebut dengan nama **interactive-gpu.pbs**.
c. Untuk menjalankannya, dapat dilakukan dengan menjalankan perintah **`qsub interactive-gpu.pbs`**. Jika terdapat *node* yang sedang *idle*, maka akan *landing* pada *node* tersebut.
d. Untuk melihat *job* PBS yang aktif, dapat dilakukan dengan cara menjalankan perintah **`qstat`**.
e. Untuk menghapus *job* PBS yang aktif, dapat dilakukan dengan cara menjalankan perintah **`qdel <job_id>`**, misalnya **`qdel 1234`**.

3. Untuk membuat *environment* sendiri menggunakan Anaconda3 (tidak menggunakan perangkat lunak bawaan HPC), dapat mengikuti langkah-langkah berikut.
   a. Ketik **`qsub interactive-gpu.pbs`**.
   b. Setelah mendapatkan *node*, selanjutnya unduh Anaconda3 dengan menjalankan perintah **`wget https://repo.anaconda.com/archive/Anaconda3-2020.02-Linux-x86_64.sh`** (bisa juga menggunakan versi terbaru jika tersedia).
   c. Lakukan instalasi perangkat lunak dengan menjalankan perintah **`sh Anaconda3-2020.02-Linux-x86_64.sh`**. Kemudian ikut langkah-langkahnya.
   d. Masukkan Anaconda3 pada *path* sistem dengan cara menjalankan perintah **`export PATH=~/anaconda3/bin:$PATH`**.
   e. Instal Tensorflow GPU dengan menjalankan perintah **`conda install tensorflow_gpu`**.

Lampiran 8 Instalasi *package* dan TensorFlow *Object Detection* API

1. Instalasi *package*
   Setelah **tensorflow_gpu** berhasil diinstal, di antara *package* yang dibutuhkan untuk proses *object detection* adalah **pillow**, **lxml**, **matplotlib**, dan **opencv-python**. Jika terdapat *package* yang belum terinstal, dapat dilakukan instalasi dengan menjalankan perintah **`conda install <nama_package>`**, misalnya **`conda install lxml`**. Cara memeriksa daftar *package* yang sudah terinstall dapat dilakukan dengan menjalankan perintah **`conda list`**.
2. Pengunduhan Tensorflow Model Garden untuk *object detection*

Perlu diperhatikan bahwa penelitian ini menggunakan Tensorflow 1.x. Penggunaan Tensorflow 2.x untuk *object detection* bisa jadi memiliki langkah-langkah atau tata cara yang berbeda.

a. Terlebih dahulu buat *folder* baru dengan nama "Tensorflow". Kemudian unduh Tensorflow Model Garden dan simpan hasil unduhan di dalam *folder* "Tensorflow". Model yang digunakan dalam penelitian ini adalah model r1.13.0. Model dapat diunduh melalui tautan https://codeload.github.com/tensorflow/models/zip/r1.13.0. Di HPC, Proses ini dapat dilakukan dengan menjalankan perintah berikut.

```
> mkdir Tensorflow
> cd Tensorflow
> mkdir models
> cd models
> wget -O models.zip
https://codeload.github.com/tensorflow/models/zip/r1.1
3.0
> unzip models.zip
```

3. Instalasi *Protocol Buffers* (Protobuf)
   Terlebih dahulu buat *folder* baru dengan nama "protoc_315" di dalam *folder* **models/research**. Kemudian di dalam *folder* **protoc_315**, unduh Protobuf versi 3.1.5 (atau sesuai kebutuhan) melalui tautan https://github.com/protocolbuffers/protobuf/releases/download/v3.15.0/protoc-3.15.0-linux-x86_64.zip. Kemudian lakukan instalasi Protobuf.
   Proses ini dan instalasi dapat dilakukan dengan menjalankan perintah berikut di dalam direktori **Tensorflow/models/research**.

```
> mkdir protoc_315
> cd protoc_315
> wget -O protobuf.zip
https://github.com/protocolbuffers/protobuf/releases/down
load/v3.15.0/protoc-3.15.0-linux-x86_64.zip
> unzip protobuf.zip
```

   Setelah Protobuf berhasil diunduh, ubah direktori (**cd**) ke **Tensorflow/models/research**, kemudian jalankan perintah berikut.

```
./protoc_315/bin/protoc object_detection/protos/*.proto -
-python_out=.
```

4. Instalasi TensorFlow *Object Detection* API
   a. Ubah direktori (**cd**) ke **Tensorflow/models/research** lalu ketik perintah **`python setup.py build`** kemudian **`python setup.py install`**.
   b. Ubah direktori (**cd**) ke **Tensorflow/models/research/slim** lalu ketik perintah **`python setup.py build`** kemudian **`python setup.py install`**.

5. Pengujian instalasi
   Untuk menguji apakah instalasi Tensorflow untuk *object detection* telah berhasil, dapat dilakukan dengan menjalankan perintah berikut pada direktori **Tensorflow/models/research**.

```
python object_detection/builders/model_builder_test.py
```

   Jika telah berhasil diinstal, maka akan ada pesan **OK** atau sejenisnya di akhir pesan yang muncul.

Lampiran 9 Pelatihan model *object detection*

Perlu diperhatikan bahwa proses anotasi citra tidak dikerjakan di HPC, melainkan di komputer/laptop pribadi. Oleh karena itu, karena keterbatasan ruang dan waktu langkah-langkah proses anotasi tidak dijelaskan di sini. Adapun tahapan pelatihan model *object detection* menggunakan TensorFlow *Object Detection* API di HPC LIPI adalah sebagai berikut.

1. Konfigurasi *pipeline* pelatihan
   Konfigurasi *pipeline* pelatihan dilakukan pada *file* *.config. Adapun langkah-langkahnya adalah sebagai berikut.
   a. Salin salah satu *file* *.config yang terdapat pada direktori **Tensorflow/models/research/object_detection/samples/configs** sesuai dengan model *object detection* yang akan digunakan. Misalnya, jika akan digunakan model SSD MobileNetV2, maka *file* yang disalin adalah **ssd_mobilenet_v2_coco.config**.
   b. Letakkan *file* *.config tersebut pada direktori **Tensorflow/workspace/melon_leaf/training**.
   c. Buka salinan *file* *.config tersebut, kemudian lakukan beberapa konfigurasi seperti berikut.
      1) Baris 9. Nilai disesuaikan dengan jumlah kelas pada dataset. Misalkan jumlah kelas adalah 2, maka ditulis,

```
num_classes: 2
```

      2) Baris 33-40. Nilai disesuaikan dengan parameter SSD seperti jumlah *layer*, skala minimum, skala maksimum, dan rasio aspek. Lakukan perubahan jika diperlukan.
      3) Baris 45-46. Nilai disesuaikan dengan ukuran citra yang akan dimasukkan pada jaringan konvolusi. *Default* untuk tinggi (*height*) dan lebar (*weight*) berturut turut adalah 300 dan 300.
      4) Baris 141. Nilai disesuaikan dengan nilai *batch size*. Jika digunakan *batch size* 16, maka pada konfigurasi ditulis,

```
batch_size: 16
```

      5) Baris 156. Konfigurasi diisi dengan *path* direktori di mana model *pretrained* disimpan, misalnya seperti berikut.

```
fine_tune_checkpoint:
"/home/ngu01234/Tensorflow/workspace/melon_leaf/pre
-trained-model/ssd_mobilenet_v2_coco/model.ckpt"
```

6) Baris 162. Nilai diisi dengan jumlah *step* saat pelatihan. Jika pelatihan akan dilakukan dengan 10.000 *step*, maka dapat ditulis,

```
num_steps: 200000
```

7) Baris 163-170. Konfigurasi dengan teknik augmentasi yang digunakan untuk pelatihan. Jika digunakan teknik augmentasi *random horizontal flip* dan SSD *random crop*, maka pada konfigurasi dituli,

```
    data_augmentation_options {
        random_horizontal_flip {
        }
      }
      data_augmentation_options {
        ssd_random_crop {
        }
}
```

Jika diinginkan *random horizontal flip* saja, maka cukup ditulis,

```
    data_augmentation_options {
        random_horizontal_flip {
        }
}
```

Jika pelatihan tidak menggunakan teknik augmentasi apapun, maka skrip tersebut dihapus.

8) Baris 175. Konfigurasi diisi dengan *path* ke direktori di mana *file* TFRECORD (*.record) untuk data latih disimpan. Misalnya,

```
input_path:
"/home/ngu01234/Tensorflow/workspace/melon_leaf/ann
otations/train.record"
```

9) Baris 177. Konfigurasi diisi dengan *path* ke direktori di mana *file label map* (*.pbtxt) disimpan. Misalnya,

```
label_map_path:
"/home/ngu01234/Tensorflow/workspace/melon_leaf/ann
otations/label_map.pbtxt"
```

10) Baris 180. Nilai diisi dengan jumlah data uji, jika diinginkan evaluasi dengan COCO API sekaligus dengan proses pelatihan (opsional). Misalnya,

```
num_examples: 8000
```

11) Baris 189. Konfigurasi dengan *path* ke direktori di mana *file* TFRECORD (*.record) untuk data uji disimpan, jika diinginkan evaluasi dengan COCO API sekaligus dengan proses pelatihan (opsional). Misalnya,

```
input_path:
"/home/ngu0270075/Tensorflow/workspace/melon_leaf/a
nnotations/test.record"
```

12) Baris 191. Konfigurasi diisi dengan *path* ke direktori di mana *file label map* (*.pbtxt) disimpan, jika diinginkan evaluasi dengan COCO API sekaligus dengan proses pelatihan (opsional). Misalnya,

```
label_map_path:
"/home/ngu0270075/Tensorflow/workspace/melon_leaf/a
nnotations/label_map.pbtxt"
```

2. Pengunduhan dan ekstraksi model *pretrained*
   a. Unduh model *pretrained* yang tersedia pada halaman https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md sesuai dengan model yang akan digunakan pada pelatihan. Misalnya, untuk melakukan pelatihan model SSD MobileNetV2, maka model *pretrained* yang diunduh adalah **ssd_mobilenet_v2_coco**.
   b. Ekstrak *file* model *pretrained* yang telah diunduh pada direktori **Tensorflow/workspace/melon_leaf/pre-trained-model/ssd_mobilenet_v2_coco**.
3. Proses pelatihan model
   a. Salin *file* **train.py** yang terdapat pada direktori **Tensorflow/models/research/object_detection/legacy**, kemudian letakkan pada direktori **Tensorflow/workspace/melon_leaf**.
   b. Ubah direktori (`cd`) ke **Tensorflow/workspace/melon_leaf**, kemudian jalankan perintah berikut.

```
python train.py --logtostderr --train_dir=training/ --
pipeline_config_path=training/ssd_mobilenet_v2_coco.co
nfig
```

4. Pascapelatihan model
   Setelah pelatihan model selesai, terdapat beberapa *file* baru pada direktori **Tensorflow/workspace/melon_leaf/training**. Jika akan dilakukan pelatihan model berikutnya, pindahkan seluruh *file* baru yang tercipta dari hasil pelatihan ke direktori lain, misalnya ke **Tensorflow/workspace/melon_leaf/training/ssd_mobilenet_v2**, agar *file* dari beberapa hasil pelatihan tidak bertumpuk.
   Terdapat tiga *file* utama yang tersimpan dengan tiga format yang berbeda, yaitu **model.ckpt-XXXX.data-00000-of-00001**, **model.ckpt-XXXX.index**, dan **model.ckpt-XXXX.meta**, yang mana XXXX merupakan *step* ketika model tersebut disimpan. Misalnya, jika model

yang tersimpan adalah dari *step* ke-10.000, maka ketiga *file* utama adalah **model.ckpt-10000.data-00000-of-00001**, **model.ckpt-10000.index**, dan **model.ckpt-10000.meta**.

5. *Export* model menjadi *inference graph*
   Pada tahap ini dilakukan *export* model *object detection* yang sudah dilatih menjadi *file inference graph.* Adapun langkah-langkahnya adalah sebagai berikut.
   a. Salin *file* **export_inference_graph.py** yang berada pada direktori **Tensorflow/models/research/object_detection**, kemudian letakkan di direktori **Tensorflow/workspace/melon_leaf**.
   b. Misalkan model yang akan di*export* adalah model SSD MobileNetV2 dari *step* ke-10.000, maka caranya adalah ubah direktori (`cd`) ke **Tensorflow/workspace/melon_leaf**, kemudian jalankan perintah berikut.

```
python export_inference_graph.py --input_type
image_tensor --pipeline_config_path training/
ssd_mobilenet_v2_coco.config --
trained_checkpoint_prefix training/
ssd_mobilenet_v2/model.ckpt-10000 --output_directory
trained-inference-graphs/ssd_mobilenet_v2
```

   Setelah beberapa saat, jika proses barhasil maka akan muncul beberapa *file* baru di direktori **Tensorflow/workspace/melon_leaf/ trained-inference-graphs/ssd_mobilenet_v2**.
   c. Sebagai tambahan, untuk mengubah nilai IOU untuk proses NMS, dapat dilakukan dengan mengubah konfigurasi pada *file* **ssd_mobilenet_v2_coco.config** baris ke-131 (`iou_threshold: 0.6`) sebelum melakukan *export* model ke *inference graph*. *Default* nilai IOU untuk NMS adalah 0,6.

Lampiran 10 Ukuran *file*

File	Metode	*Backbone*	Ukuran (MB)
*.data-00000-of-00001	Faster R-CNN	InceptionV2	102,255
		ResNet50	281,400
	SSD	InceptionV2	208,192
		MobileNetV2	72,450
	YOLOv3	Darknet-53	322,367
*.index	Faster R-CNN	InceptionV2	0,025
		ResNet50	0,022
	SSD	InceptionV2	0,053
		MobileNetV2	0,041
	YOLOv3	Darknet-53	0,018
*.meta	Faster R-CNN	InceptionV2	5,229
		ResNet50	4,806
	SSD	InceptionV2	10,607
		MobileNetV2	8,255
	YOLOv3	Darknet-53	15,717

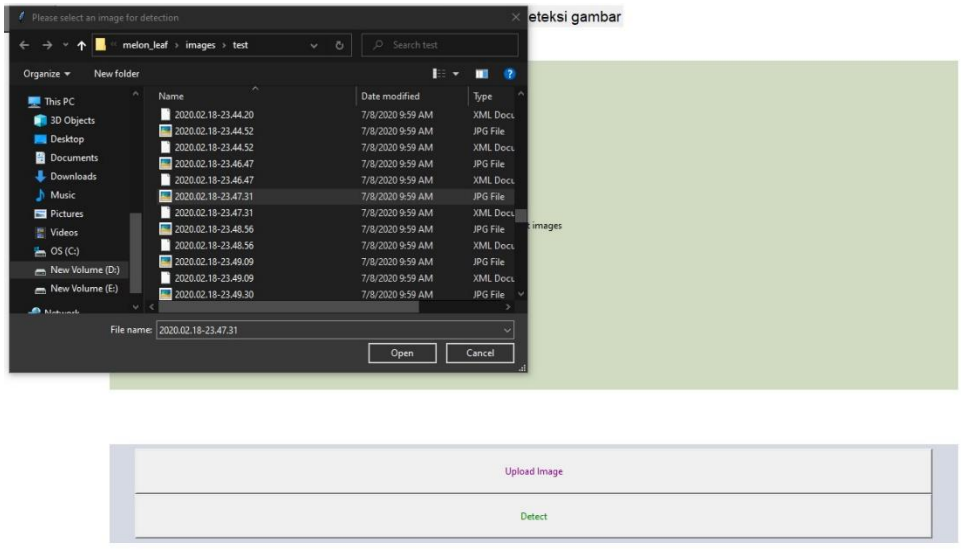
File	Metode	*Backbone*	Ukuran (MB)
inference graph (*.pb)	Faster R-CNN	InceptionV2	51,880
		ResNet50	112,178
	SSD	InceptionV2	52,725
		MobileNetV2	18,766

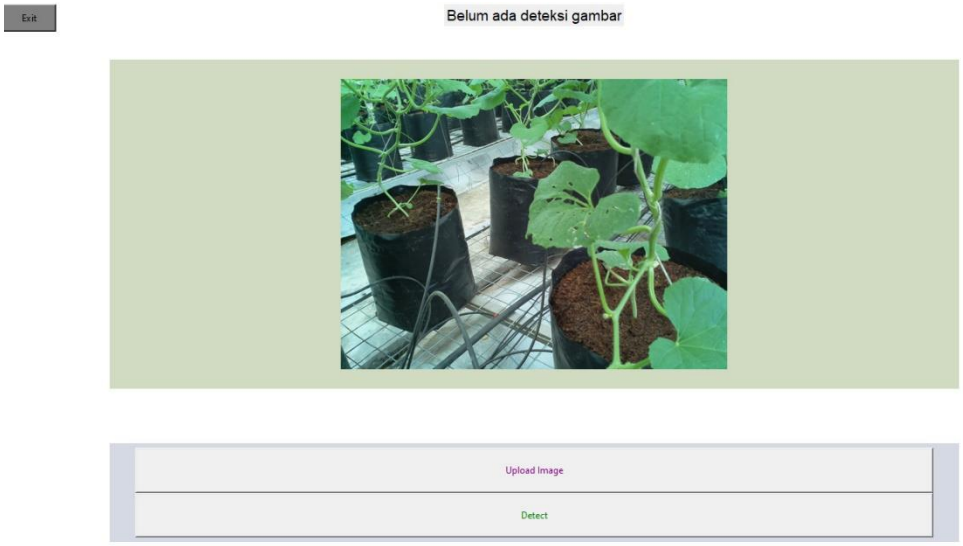
Lampiran 11 Aplikasi *object detection* untuk deteksi kelainan daun melon

Beserta penelitian ini dibuat sebuah aplikasi untuk mendeteksi kelainan pada daun tanaman melon. Adapun tampilan dan fungsi yang terdapat pada aplikasi tersebut adalah sebagai berikut.

1. Halaman depan

2. Unggah gambar
   Unggah gambar dilakukan dengan menekan tombol "Upload Image", kemudian memilih gambar yang akan dilakukan *object detection*.





3. *Object detection* pada kelainan daun melon
   Fungsi *object detection* dilakukan dengan menekan tombol "Detect". Keluaran yang dihasilkan adalah gambar, *bounding* box, label kelas, dan jumlah masing-masing kelas yang terdeteksi.