Pelatihan Deep Learning Using Tensorflow

INSTITUT TEKNOLOGI
NASIONAL BANDUNG
2019

1. PREPARING DATASET

Sebelum kita memulai untuk membuat model dari CNN kita, kita akan membuat dataset dengan memecah menjadi data training, validation, dan testing. Berikut langkah langkah yang perlu dilakukan.

1. Buka aplikasi Spyder
2. Tentukan direktori yang akan kita gunakan sebagai working directori
3. Buat file bernama "build_dataset.py" pada spyder dan salin kode dibawah ini

```
# import the necessary packages
from pyimagesearch import config
from imutils import paths
import random
import shutil
import os

# grab the paths to all input images in the original input directory
# and shuffle them
imagePaths = list(paths.list_images(config.ORIG_INPUT_DATASET))
random.seed(42)
random.shuffle(imagePaths)

# compute the training and testing split
i = int(len(imagePaths) * config.TRAIN_SPLIT)
trainPaths = imagePaths[:i]
testPaths = imagePaths[i:]

# we'll be using part of the training data for validation
i = int(len(trainPaths) * config.VAL_SPLIT)
valPaths = trainPaths[:i]
trainPaths = trainPaths[i:]

# define the datasets that we'll be building
datasets = [
   ("training", trainPaths, config.TRAIN_PATH),
   ("validation", valPaths, config.VAL_PATH),
   ("testing", testPaths, config.TEST_PATH)
]

# loop over the datasets
for (dType, imagePaths, baseOutput) in datasets:
   # show which data split we are creating
   print("[INFO] building '{}' split".format(dType))

   # if the output base output directory does not exist, create it
   if not os.path.exists(baseOutput):
      print("[INFO] 'creating {}' directory".format(baseOutput))
      os.makedirs(baseOutput)

   # loop over the input image paths
   for inputPath in imagePaths:
```

```
        # extract the filename of the input image along with its
        # corresponding class label
        filename = inputPath.split(os.path.sep)[-1]
        label = inputPath.split(os.path.sep)[-2]

        # build the path to the label directory
        labelPath = os.path.sep.join([baseOutput, label])

        # if the label output directory does not exist, create it
        if not os.path.exists(labelPath):
           print("[INFO] 'creating {}' directory".format(labelPath))
           os.makedirs(labelPath)

        # construct the path to the destination image and then copy
        # the image itself
    p = os.path.sep.join([labelPath, filename])
```

2. BASIC CONVOLUTIONAL NEURAL NETWORK WITH KERAS TENSORFLOW

Pada modul pertama ini, kita akan mencoba untuk membangun sebuah model berdasarkan struktur sederhana dari CNN yang terdiri dari jaringan konvolusi, pooling, dan fully connected layer. Implementasi akan dilakukan menggunakan library keras dengan backend tensorflow. Berikut ini merupakan langkah-langkah yang dilakukan untuk membentuk model:

1. Buat File bernama Config.py yang berisi code python seperti pada dibawah ini.

```
import os
from imutils import paths

IM_WIDTH=224
IM_HEIGHT=224
EPOCH=2
batch_size=32

# initialize the path to the *original* input directory of images
ORIG_INPUT_DATASET = "dataset/Cat&Dog"

# initialize the base path to the *new* directory that will contain
# our images after computing the training and testing split
BASE_PATH = "dataset/Cat&Dog/"

# derive the training, validation, and testing directories
TRAIN_PATH = os.path.sep.join([BASE_PATH, "training"])
VAL_PATH = os.path.sep.join([BASE_PATH, "validation"])
TEST_PATH = os.path.sep.join([BASE_PATH, "testing"])

totalTrain = len(list(paths.list_images(TRAIN_PATH)))
totalVal = len(list(paths.list_images(VAL_PATH)))
```

2. Kemudian pada working directory buat direktori bernama Model dan buat file bernama Classical.py didalam direktori model seperti pada Gambar 1.

Gambar 1. *Working directory*

3. Salin kode berikut ini pada file Classical.py

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
```

```
def ClassicalModel(input_shape):
    model = Sequential()
    model.add(Conv2D(32, (3, 3), input_shape=input_shape))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(32, (3, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(64, (3, 3)))
    model.add(Activation('relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.add(Dense(64,activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(2, activation='softmax'))
    model.compile(optimizer='adam',
                  loss='categorical_crossentropy', metrics=['accuracy'])

    return model
```

4. Kemudian pada working direktori diluar direktori Model buat file bernama train_model.py dan salin kode dibawah ini

```
from keras.preprocessing.image import ImageDataGenerator
from keras import backend as K
from keras.callbacks import TensorBoard
from time import time
#from keras.utils import plot_model
import Config as cf
from Model import Classical as cl

# this is the augmentation configuration we will use for training
train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

# this is the augmentation configuration we will use for testing:
# only rescaling
val_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    cf.TRAIN_PATH,
    target_size=(cf.IM_WIDTH, cf.IM_HEIGHT),
    batch_size=cf.batch_size,
    class_mode='categorical')

validation_generator = val_datagen.flow_from_directory(
    cf.VAL_PATH,
    target_size=(cf.IM_WIDTH, cf.IM_HEIGHT),
    batch_size=cf.batch_size,
```

```
        class_mode='categorical')
    tensorboard = TensorBoard(log_dir="logs/{}".format(time()))
    model = cl.ClassicalModel(input_shape)
    model.fit_generator(
                train_generator,
                steps_per_epoch=cf.totalTrain // cf.batch_size,
                epochs=cf.EPOCH,
                validation_data=validation_generator,
                validation_steps=cf.totalVal // cf.batch_size,
                verbose=1,
                callbacks=[tensorboard])
    model.save_weights('CatDogModel5.h5')
model.save('CatDogModel5.h5')
```

5. Kemudian Run train_model.py dan tunggu proses training selesai hingga dihasilkan file model berekstensi ***.h5**

6. Untuk mengetahui kinerja dari model yang kita buat maka buatlah file bernama testing.py untuk mengujinya. Salinlah kode dibawah ini

```
    from keras.models import load_model
    from keras.preprocessing import image
    import numpy as np
    from os import listdir
    from os.path import isfile, join
    # dimensions of our images
    img_width, img_height = 224, 224

    # load the model we saved
    model = load_model('CatDogModel3.h5')
    model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accu
    racy'])

    mypath = "predict/"
    onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath, f))]
    print(onlyfiles)
    # predicting images
    cat_counter = 0
    dog_counter  = 0
    for file in onlyfiles:
        img     =     image.load_img(mypath+file,     target_size=(img_width,
    img_height,3))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)

        images = np.vstack([x])
        classes = model.predict_classes(images)
        classes = classes[0]

        if classes == 0:
            print(file + ": " + 'Cat')
            cat_counter += 1
```

```
    #    if classes==1:
    #        print(file + ": " + 'Apricot')
    #        apricot_counter += 1
        else:
            print(file + ": " + 'Dog')
            dog_counter += 1
    print("Total Cat :",cat_counter)
print("Total Dog :",cat_counter)
```

3. GOOGLE NET ARCHITECTURE WITH KERAS TENSORFLOW

Materi berikutnya mencoba untuk membangun sebuah berdasarkan pendekatan deep learning menggunakan arsitektur GoogLeNet. Implementasi dilakukan menggunakan library keras dengan backend tensorflow. Berikut ini merupakan langkah-langkah yang dilakukan untuk membentuk model:

1. Buat file bernama GoogleNet.py pada direktori model dan salin kode dibawah ini

```
from keras.layers import Conv2D, MaxPooling2D, AveragePooling2D
from keras.layers import Flatten, Dense, Dropout,BatchNormalization
from keras import regularizers
from keras.layers import Input, concatenate
from keras.models import Model
import keras.metrics as metric


def
conv2D_lrn2d(x,DATA_FORMAT,LRN2D_NORM,WEIGHT_DECAY,filters,kernel_size,

strides=(1,1),padding='same',dilation_rate=(1,1),activation='relu',
                 use_bias=True,kernel_initializer='glorot_uniform',
                 bias_initializer='zeros',kernel_regularizer=None,
                 bias_regularizer=None,activity_regularizer=None,
                 kernel_constraint=None,bias_constraint=None):
    #l2 normalization
    if WEIGHT_DECAY:
        kernel_regularizer=regularizers.l2(WEIGHT_DECAY)
        bias_regularizer=regularizers.l2(WEIGHT_DECAY)
    else:
        kernel_regularizer=None
        bias_regularizer=None

    x=Conv2D(filters=filters,kernel_size=kernel_size,strides=strides,

padding=padding,data_format=DATA_FORMAT,dilation_rate=dilation_rate,

activation=activation,use_bias=use_bias,kernel_initializer=kernel_initial
izer,

bias_initializer=bias_initializer,kernel_regularizer=kernel_regularizer,

bias_regularizer=bias_regularizer,activity_regularizer=activity_regulariz
er,

kernel_constraint=kernel_constraint,bias_constraint=bias_constraint)(x)

    if LRN2D_NORM:
        #batch normalization
        x=BatchNormalization()(x)
```

```
    return x

def
inception_module(x,DATA_FORMAT,LRN2D_NORM,params,concat_axis,padding='sam
e',
                     dilation_rate=(1,1),activation='relu',use_bias=True,

kernel_initializer='glorot_uniform',bias_initializer='zeros',
                     kernel_regularizer=None,bias_regularizer=None,
                     activity_regularizer=None,kernel_constraint=None,
                     bias_constraint=None,weight_decay=None):
    (branch1,branch2,branch3,branch4)=params
    if weight_decay:
        kernel_regularizer=regularizers.l2(weight_decay)
        bias_regularizer=regularizers.l2(weight_decay)
    else:
        kernel_regularizer=None
        bias_regularizer=None
    #1x1

pathway1=Conv2D(filters=branch1[0],kernel_size=(1,1),strides=1,padding=pa
dding,
                     data_format=DATA_FORMAT,dilation_rate=dilation_rate,
                     activation=activation,use_bias=use_bias,
                     kernel_initializer=kernel_initializer,
                     bias_initializer=bias_initializer,
                     kernel_regularizer=kernel_regularizer,
                     bias_regularizer=bias_regularizer,
                     activity_regularizer=activity_regularizer,
                     kernel_constraint=kernel_constraint,
                     bias_constraint=bias_constraint)(x)

    #1x1->3x3

pathway2=Conv2D(filters=branch2[0],kernel_size=(1,1),strides=1,padding=pa
dding,
                     data_format=DATA_FORMAT,dilation_rate=dilation_rate,
                     activation=activation,use_bias=use_bias,
                     kernel_initializer=kernel_initializer,
                     bias_initializer=bias_initializer,
                     kernel_regularizer=kernel_regularizer,
                     bias_regularizer=bias_regularizer,
                     activity_regularizer=activity_regularizer,
                     kernel_constraint=kernel_constraint,
                     bias_constraint=bias_constraint)(x)


pathway2=Conv2D(filters=branch2[1],kernel_size=(3,3),strides=1,padding=pa
dding,
                     data_format=DATA_FORMAT,dilation_rate=dilation_rate,
                     activation=activation,use_bias=use_bias,
```

```
                    kernel_initializer=kernel_initializer,
                    bias_initializer=bias_initializer,
                    kernel_regularizer=kernel_regularizer,
                    bias_regularizer=bias_regularizer,
                    activity_regularizer=activity_regularizer,
                    kernel_constraint=kernel_constraint,
                    bias_constraint=bias_constraint)(pathway2)

    #1x1->5x5

pathway3=Conv2D(filters=branch3[0],kernel_size=(1,1),strides=1,padding=pa
dding,
                    data_format=DATA_FORMAT,dilation_rate=dilation_rate,
                    activation=activation,use_bias=use_bias,
                    kernel_initializer=kernel_initializer,
                    bias_initializer=bias_initializer,
                    kernel_regularizer=kernel_regularizer
                    ,bias_regularizer=bias_regularizer,
                    activity_regularizer=activity_regularizer,
                    kernel_constraint=kernel_constraint,
                    bias_constraint=bias_constraint)(x)


pathway3=Conv2D(filters=branch3[1],kernel_size=(5,5),strides=1,padding=pa
dding,
                    data_format=DATA_FORMAT,dilation_rate=dilation_rate,
                    activation=activation,use_bias=use_bias,
                    kernel_initializer=kernel_initializer,
                    bias_initializer=bias_initializer,
                    kernel_regularizer=kernel_regularizer,
                    bias_regularizer=bias_regularizer,
                    activity_regularizer=activity_regularizer,
                    kernel_constraint=kernel_constraint,
                    bias_constraint=bias_constraint)(pathway3)

    #3x3->1x1
    pathway4=MaxPooling2D(pool_size=(3,3),strides=1,padding=padding,
                          data_format=DATA_FORMAT)(x)


pathway4=Conv2D(filters=branch4[0],kernel_size=(1,1),strides=1,padding=pa
dding,
                    data_format=DATA_FORMAT,dilation_rate=dilation_rate,
                    activation=activation,use_bias=use_bias,
                    kernel_initializer=kernel_initializer,
                    bias_initializer=bias_initializer,
                    kernel_regularizer=kernel_regularizer,
                    bias_regularizer=bias_regularizer,
                    activity_regularizer=activity_regularizer,
                    kernel_constraint=kernel_constraint,
                    bias_constraint=bias_constraint)(pathway4)
```

```
    return
concatenate([pathway1,pathway2,pathway3,pathway4],axis=concat_axis)

def create_model(DATA_FORMAT,LRN2D_NORM,WEIGHT_DECAY,DROPOUT,NB_CLASS):
    #Data format:tensorflow,channels_last;theano,channels_last
    if DATA_FORMAT=='channels_first':
        INP_SHAPE=(3,224,224)
        img_input=Input(shape=INP_SHAPE)
        CONCAT_AXIS=1
    elif DATA_FORMAT=='channels_last':
        INP_SHAPE=(224,224,3)
        img_input=Input(shape=INP_SHAPE)
        CONCAT_AXIS=3
    else:
        raise Exception('Invalid Dim Ordering')

    x=conv2D_lrn2d(img_input,DATA_FORMAT,False,WEIGHT_DECAY,64,(7,7),2,
                   padding='same')

x=MaxPooling2D(pool_size=(3,3),strides=2,padding='same',data_format=DATA_
FORMAT)(x)
    x=BatchNormalization()(x)


x=conv2D_lrn2d(x,DATA_FORMAT,False,WEIGHT_DECAY,64,(1,1),1,padding='same'
)


x=conv2D_lrn2d(x,DATA_FORMAT,True,WEIGHT_DECAY,192,(3,3),1,padding='same'
)

x=MaxPooling2D(pool_size=(3,3),strides=2,padding='same',data_format=DATA_
FORMAT)(x)

    x=inception_module(x,DATA_FORMAT,LRN2D_NORM,

params=[(64,),(96,128),(16,32),(32,)],concat_axis=CONCAT_AXIS) #3a
    x=inception_module(x,DATA_FORMAT,LRN2D_NORM,

params=[(128,),(128,192),(32,96),(64,)],concat_axis=CONCAT_AXIS) #3b

x=MaxPooling2D(pool_size=(3,3),strides=2,padding='same',data_format=DATA_
FORMAT)(x)

    x=inception_module(x,DATA_FORMAT,LRN2D_NORM,

params=[(192,),(96,208),(16,48),(64,)],concat_axis=CONCAT_AXIS) #4a
    x=inception_module(x,DATA_FORMAT,LRN2D_NORM,

params=[(160,),(112,224),(24,64),(64,)],concat_axis=CONCAT_AXIS) #4b
    x=inception_module(x,DATA_FORMAT,LRN2D_NORM,
```

```
params=[(128,),(128,256),(24,64),(64,)],concat_axis=CONCAT_AXIS) #4c
    x=inception_module(x,DATA_FORMAT,LRN2D_NORM,

params=[(112,),(144,288),(32,64),(64,)],concat_axis=CONCAT_AXIS) #4d
    x=inception_module(x,DATA_FORMAT,LRN2D_NORM,

params=[(256,),(160,320),(32,128),(128,)],concat_axis=CONCAT_AXIS) #4e

x=MaxPooling2D(pool_size=(3,3),strides=2,padding='same',data_format=DATA_
FORMAT)(x)

    x=inception_module(x,DATA_FORMAT,LRN2D_NORM,

params=[(256,),(160,320),(32,128),(128,)],concat_axis=CONCAT_AXIS) #5a
    x=inception_module(x,DATA_FORMAT,LRN2D_NORM,

params=[(384,),(192,384),(48,128),(128,)],concat_axis=CONCAT_AXIS) #5b

x=AveragePooling2D(pool_size=(7,7),strides=1,padding='valid',data_format=
DATA_FORMAT)(x)

    x=Flatten()(x)
    x=Dropout(DROPOUT)(x)
    x=Dense(output_dim=NB_CLASS,activation='linear')(x)
    x=Dense(output_dim=NB_CLASS,activation='softmax')(x)

    return x,img_input,CONCAT_AXIS,INP_SHAPE,DATA_FORMAT

def check_print(DATAFORMAT,LRN2D_NORM,WEIGHT_DECAY,DROPOUT,NB_CLASS):
    # Create the Model

x,img_input,CONCAT_AXIS,INP_SHAPE,DATA_FORMAT=create_model(DATAFORMAT,LRN
2D_NORM,WEIGHT_DECAY,DROPOUT,NB_CLASS)

    # Create a Keras Model
    model=Model(input=img_input,output=[x])
    model.summary()

    # Save a PNG of the Model Build
#    plot_model(model,to_file='GoogLeNet.png')

    model.compile(optimizer='adam',loss='categorical_crossentropy',
                  metrics=['acc',metric.top_k_categorical_accuracy])
    print('Model Compiled')
  return model
```

2. Buat file bernama train_google.py dan salin kode dibawah ini:

```
from keras.preprocessing.image import ImageDataGenerator
import Config as cf
from Model import GoogleNet as gm
```

```
import keras.metrics as metric

#train data
train_datagen = ImageDataGenerator(
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    featurewise_center=True
)
train_generator = train_datagen.flow_from_directory(
  cf.TRAIN_PATH,
  target_size=(cf.IM_WIDTH, cf.IM_HEIGHT),
  batch_size=cf.batch_size,
)

#vaild data
vaild_datagen = ImageDataGenerator(
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    featurewise_center=True
)
vaild_generator = train_datagen.flow_from_directory(
  cf.VAL_PATH,
  target_size=(cf.IM_WIDTH, cf.IM_HEIGHT),
  batch_size=cf.batch_size,
)

#test data
test_datagen = ImageDataGenerator(
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    featurewise_center=True
)
test_generator = train_datagen.flow_from_directory(
  cf.TEST_PATH,
  target_size=(cf.IM_WIDTH, cf.IM_HEIGHT),
  batch_size=cf.batch_size,
)
```

```
    model=
    gm.check_print(cf.DATA_FORMAT,cf.LRN2D_NORM,cf.WEIGHT_DECAY,cf.DROPOUT,cf
    .NB_CLASS)

    model.fit_generator(train_generator,validation_data=vaild_generator,epoch
    s=cf.EPOCH,
                        steps_per_epoch=train_generator.n/cf.batch_size,
                        validation_steps=vaild_generator.n/cf.batch_size)
    model.save('FruitsModel.h5')
model.metrics=['acc',metric.top_k_categorical_accuracy]
```

3. Atur terlebih dahulu NB_Class dan direktori dataset pada file Config.py

```
NB_CLASS=3
DROPOUT=0.4
WEIGHT_DECAY=0.0005
LRN2D_NORM=True
DATA_FORMAT='channels_last' # Theano:'channels_first' Tensorflow:'channels_last'
IM_WIDTH=224
IM_HEIGHT=224
EPOCH=10
batch_size=32



# initialize the path to the *original* input directory of images
ORIG_INPUT_DATASET = "dataset/Fruits"

# initialize the base path to the *new* directory that will contain
# our images after computing the training and testing split
BASE_PATH = "dataset/Fruits/"

# derive the training, validation, and testing directories
TRAIN_PATH = os.path.sep.join([BASE_PATH, "training"])
VAL_PATH = os.path.sep.join([BASE_PATH, "validation"])
TEST_PATH = os.path.sep.join([BASE_PATH, "testing"])
```

Gambar 2. Konfigurasi file Config.py

4. Jalankan train_google.py dan tunggu proses training selesai hingga dihasilkan file model berekstensi *.h5.
5. Salin dan jalankan kode dibawah ini untuk melakukan pengujian terhadap model yang telah kita buat.

```
from keras.models import load_model
from keras.preprocessing import image
import numpy as np
from os import listdir
from os.path import isfile, join
# dimensions of our images
img_width, img_height = 224, 224

# load the model we saved
model = load_model('FruitsModel.h5')
```

```
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accur
acy'])

mypath = "predict2/"
onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath, f))]
print(onlyfiles)
# predicting images
aple_counter = 0
apricot_counter = 0
guava_counter =0
for file in onlyfiles:
    img     =     image.load_img(mypath+file,     target_size=(img_width,
img_height,3))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)

    images = np.vstack([x])
    classes = model.predict(images)
    label = classes
    classes=np.argmax(classes)

    if classes == 0:
        print(file + ": " + 'Apel')
        aple_counter += 1
    elif classes==1:
        print(file + ": " + 'Apricot')
        apricot_counter += 1
    else:
        print(file + ": " + 'Guava')
        guava_counter += 1
print("Total Cat :",aple_counter)
print("Total Dog :",apricot_counter)
print("Total Cat :",guava_counter)
```

PREPARING DATASET WITH LABELIMG FOR OBJECT DETECTION

Materi berikutnya adalah membuat dataset sendiri dengan memberi label pada objek-objek yang ingin dideteksi pada setiap citra. Proses tersebut dilakukan menggunakan LabelImg. Berikut ini merupakan langkah-lagkah yang dilakukan untuk membuat dataset:

1. Download file LabelImg yang terdapat pada link berikut ini
   https://github.com/tzutalin/labelImg
2. Ekstrak file yang telah didownload tersebut.
3. Buka command prompt dan buka direktori tempat kita mengekstrak LabelImg.rar yang telah kita download.
4. Salin kode dibawah ini pada command prompt kemudian eksekusi

```
conda install pyqt=5
```

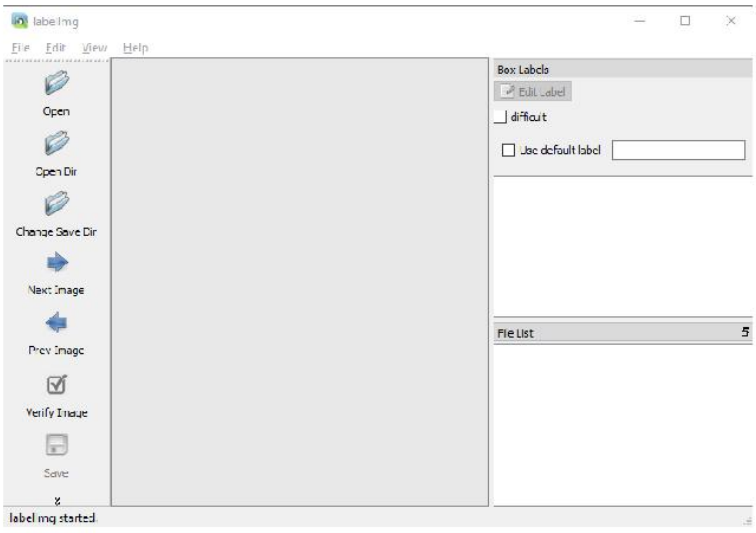
5. Kemudian salin kembali kode dibawah ini dan eksekusi

```
pyrcc5 -o libs/resources.py resources.qrc
```

6. Jalankan LabelImg dengan mengeksekusi kode dibawah ini

```
python labelImg.py
```

7. LabelImg akan terbuka dengan tampilan pada Gambar dibawah ini.



Gambar 3. Halaman LabelImg

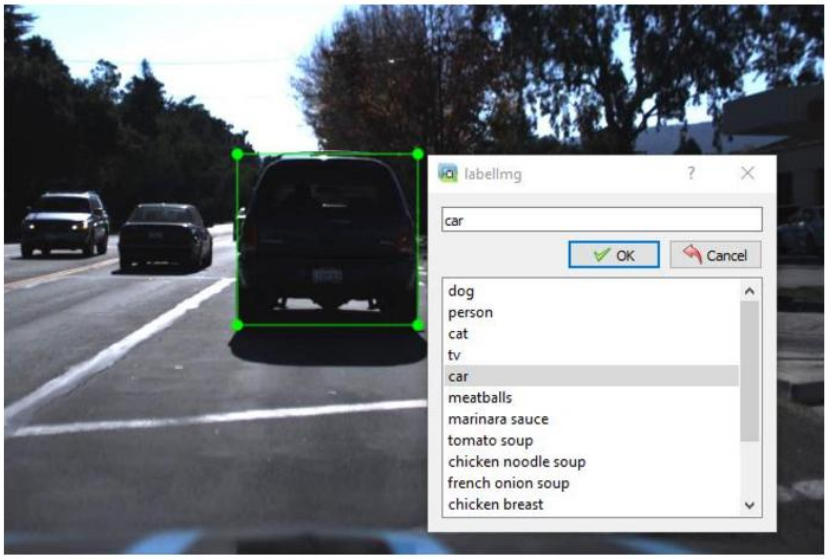
8. Pilih direktori tempat kita menyimpan dataset dengan memilih menu seperti gambar berikut

Gambar 4. Pilih Direktori

9. Kemudian pilih menu CreateRectBox seperti yang ditunjukan oleh gambar berikut.

Gambar 5. Menu CreateRectBox

10. Buatlah kotak di area objek yang akan menjadi dataset dan beri label atas objek yang kita tandai dengan kotak tersebut dan pilih oke



Gambar 6. Create Box Objek

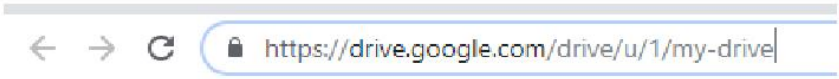
11. Setelah selesai memberi label pada setiap objek, maka pilih menu verify image atau save untuk menyimpan koordinat dari objek dataset.

Gambar 7. Verifikasi Gambar

TRAINING DATASET OBJECT DETECTION RETINANET

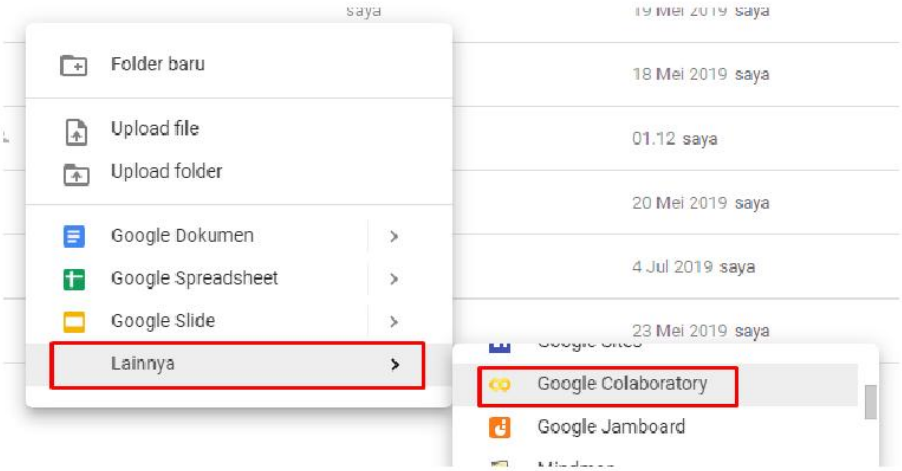
Pada modul *object detection* dengan algoritma RetinaNet akan memanfaatkan platform google colab untuk training. Google colab merupakan sebuah platform jupyter notebook python yang terdapat pada server google. Keuntungan menggunakan Google Colab adalah GPU Nvidia Tesla T4 sebagai runtime booster yang dapat mempercepat proses training. Berikut ini merupakan tahapan untuk melakukan training pada Google Colab.

1. Buka google drive dengan link seperti gambar dibawah ini



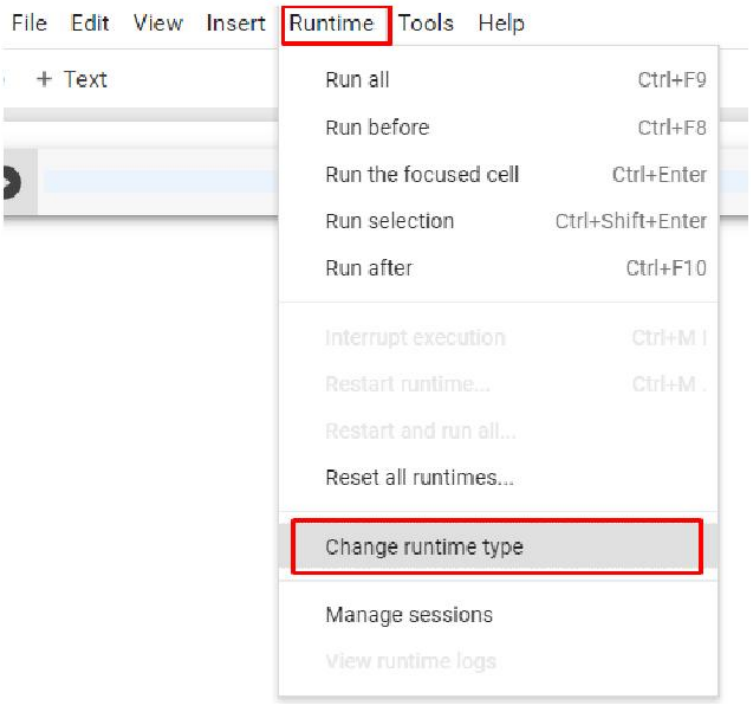
Gambar 8. Akses GDrive

2. Klik kanan pada halaman Google Colab kemudian pilih lainnya dan pilih google colaboratory seperti Gambar dibawah ini.



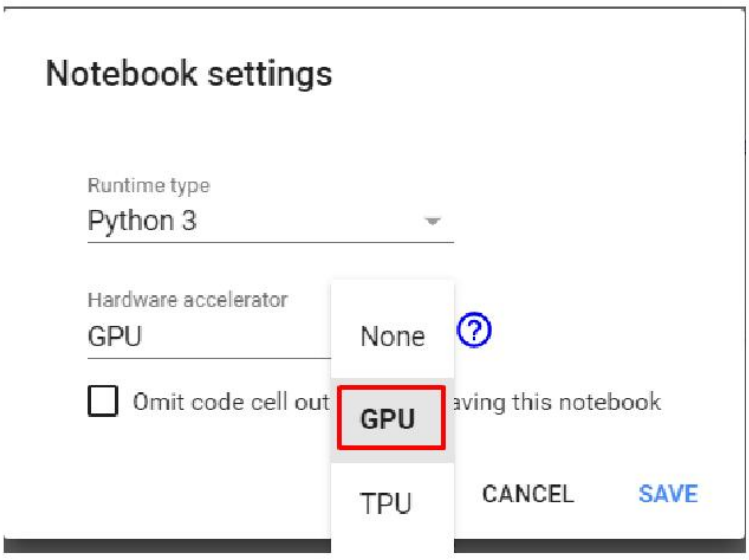
Gambar 9. *Google Colaboratory*

3. Setelah halaman google colaboratory terbuka selanjutnya aktifkan GPU runtime booster. Untuk mengaktifkan runtime booster, pilih menu "Runtime" dan pilih submenu "change runtime type" seperti pada gambar di bawah ini



Gambar 10. Aktifkan GPU Runtime Booster

4. Setelah langkah 3 dikakukan maka akan muncul menu seperti Gambar11, kemudian pilih "*Hardware accelerator*" dan pilih "GPU"

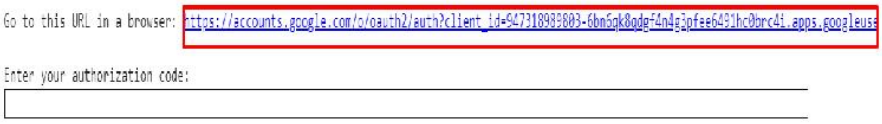


Gambar 11. *Notebook Setting*

5. Selanjutnya salin kode berikut dan tulis pada baris pertama Google Colab. Kemudian tekan tombol "Ctrl + Enter" atau pilih tombol play pada sebelah kiri baris kode untuk mengeksekusi kode. Kode pada baris pertama ini berfungsi sebagai permintaan permission untuk mengakses direktori pada google drive.

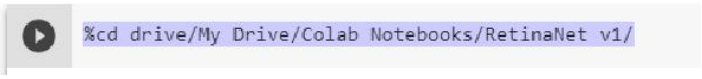
Gambar 12. Listing Kode

6. Setelah kode pertama dieksekusi maka akan muncul respon seperti pada Gambar 13. Kemudian klik link berwarna biru yang ditunjukan oleh kotak berwarna merah untuk mendapatkan verification code yang harus dimasukan pada textbox dibawahnya. Setelah *verification code* didapatkan, maka tekan tombol enter pada keyboard untuk dieksekusi kembali.



Gambar 13. Verification Code

7. Kemudian salin kode dibawah ini untuk menetapkan working directory, mulai dari My Drive/, ikuti direktori tempat file google colab yang dibuat. Kemudian tekan tombol Ctrl+Enter pada keyboard untuk mengeksekusi kode.

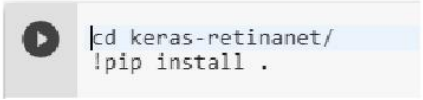


Gambar 14. Set Working Directory

8. Salin kode dibawah ini untuk mendapatkan kode retinanet yang dibutuhkan, kemudian eksekusi kode dan tunggu hingga proses clone selesai.

Gambar 15. Listing Code Retinanet

9. Kemudian salin dan eksekusi dua baris kode seperti yang ditunjukan pada Gambar 16 dan tunggu hingga instalasi selesai.



Gambar 16. Instalasi

10. Tulis dan eksekusi kode pada Gambar 17

```
!python setup.py build_ext --inplace
```

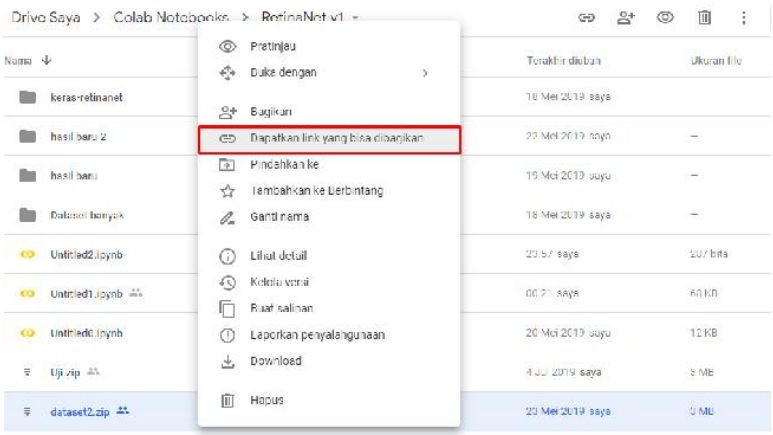
Gambar 17. Eksekusi

11. Import library yang tertera pada Gambar 18 kemudian eksekusi

```
import os
import stat
import shutil
import zipfile
import urllib
import xml.etree.ElementTree as ET
import numpy as np
import csv
import pandas
from google.colab import drive
from google.colab import files
```

Gambar 18. Import library

12. Upload file dataset dalam format *.zip dari dataset yang telah kita siapkan dari labelImg kemudian klik kanan pada file dataset kita dan pilih "Dapatkan link yang bisa dibagikan" seperti yang ditunjukan oleh Gambar 19.



Gambar 19. Upload file dataset

13. Paste link dari dataset yang telah kita dapatkan kemudian block dan copy id dari dataset tersebut seperti yang ditunjukan oleh Gambar 20

Gambar 20. Paste link dari dataset

14. Salin kode dibawah ini kemudian tempatkan id dataset yang telah kita copy sebelumnya pada bagian yang ditunjukan oleh kotak merah pada Gambar 21.

```
DATASET_DRIVEID = '1yKpNdfVxDVgfruF-nS15I_mYTkLKcagf'
DATASET_DIR = 'dataset'
ANNOTATIONS_FILE = 'carlabel_set.csv'
VALIDATION_FILE = 'carlabel_val.csv'
CLASSES_FILE = 'classes.csv'
```

Gambar 21. Konfigurasi Direktori Dataset

15. Salin kode dibawah ini untuk mengekstrak dataset yang masih dalam format *.zip

```
[ ]  drive_url = 'https://drive.google.com/uc?export=download&id=' + DATASET_DRIVEID
     file_name = 'dataset2.zip'

     urllib.request.urlretrieve(drive_url, file_name)
     print('Download completed!')

     os.makedirs(DATASET_DIR, exist_ok=True)
     with zipfile.ZipFile(file_name, 'r') as zip_ref:
       zip_ref.extractall(DATASET_DIR)
     os.remove(file_name)
     print('Extract completed!')
```

Gambar 22. Ekstraksi Dasaset

16. Salin dan eksekusi pada kode yang terdapat pada gambar 23 untuk membuat dataset kita sesuai format yang dapat digunakan pada proses training.

```
annotations = []
classes = set([])

for xml_file in [f for f in os.listdir(DATASET_DIR) if f.endswith(".xml")]:
  tree = ET.parse(os.path.join(DATASET_DIR, xml_file))
  root = tree.getroot()

  file_name = None

  for elem in root:
    if elem.tag == 'filename':
      file_name = os.path.join(DATASET_DIR, elem.text)
```

```
    if elem.tag == 'object':
      obj_name = None
      coords = []
      for subelem in elem:
        if subelem.tag == 'name':
          obj_name = subelem.text
        if subelem.tag == 'bndbox':
          for subsubelem in subelem:
            coords.append(subsubelem.text)
      item = [file_name] + coords + [obj_name]
      annotations.append(item)
      classes.add(obj_name)

with open(ANNOTATIONS_FILE, 'w') as f:
  writer = csv.writer(f)
  writer.writerows(annotations)

with open(CLASSES_FILE, 'w') as f:
  for i, line in enumerate(classes):
    f.write('{},{}\n'.format(line,i))
```

Gambar 23. Listing Program Proses Training

17. Buka file train.py pada direktori keras-resnet/bin kemudian tambahkan kode yang diberi kotak merah pada bagian compile dibawah optimizer seperti yang ditunjukan oleh gambar 24.

```
# compile model
training_model.compile(
    loss={
        'regression'    : losses.smooth_l1(),
        'classification': losses.focal()
    },
    optimizer=keras.optimizers.adam(lr=1e-5, clipnorm=0.001),
    metrics=['acc']
)
```

Gambar 24. file train.py

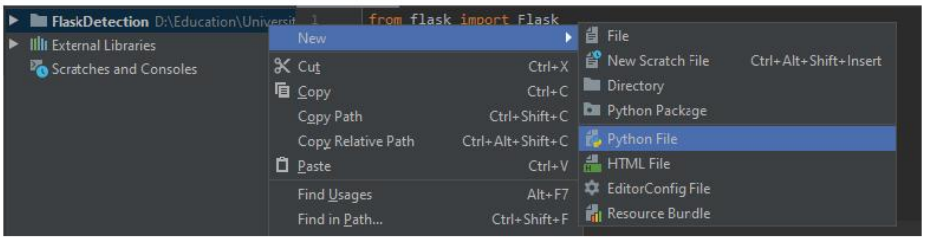
18. Kembali lagi ke google colab dan salin kode dibawah ini kemudian eksekusi, tunggu proses training selesai hingga dihasilkan model dengan ekstensi file *.h5 pada direktori snapshots.

```
!python3 keras_retinanet/bin/train.py --freeze-backbone --random-transform --weighted-average --batch-size 16 --epochs 100 --steps 100 csv annotations.csv classes.csv
```

OBJECT DETECTION DEPLOYMENT WITH FLASK AND HTML5UP

Pada kesempatan kali ini kita akan mencoba menerapkan sistem deteksi objek menjadi sebuah aplikasi web yang mudah untuk digunakan oleh user dengan memanfaatkan framework python flask sebagai back-end dan Html5UP sebagai front-end dari sistem ini. Berikut ini langkah-langkah yang dilakukan untuk membuat flask web app.

1. Buka Jetbrains pycharm dan create new project dengan nama project FlaskDetection.
2. Kemudian pada direktori FlaskDetection klik kanan pilih new dan pilih python file. Beri nama app.py seperti yang ditunjukan oleh Gambar 24.



Gambar 25. Direktori Flaskdetection

3. Buka file app.py dan salin kode yang terdapat pada Gambar 26

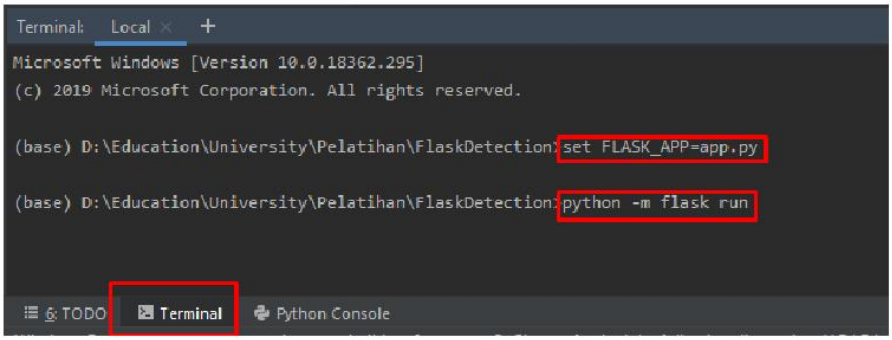
```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello World!"

if __name__ == '__main__':
    app.run()
```

Gambar 26. file app.py

4. Untuk mengecek terlebih dahulu apakah flask app dapat dijalankan maka buka terminal pada bagian bawah pycharm dan eksekusi dua kode yang ditunjukan oleh dua kotak merah pada Gambar 27.

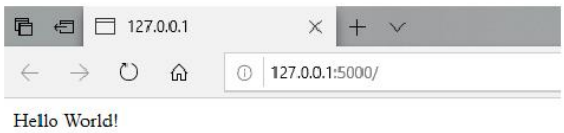


Gambar 27. Cek flask app

5. Setelah langkah 4 dilakukan maka akan tampil response seperti pada Gambar 4, kemudian klik link http://127.0.0.1:5000/ yang tertera pada terminal. Ketika link tersebut dibuka maka akan menampilkan halaman seperti pada Gambar 28.

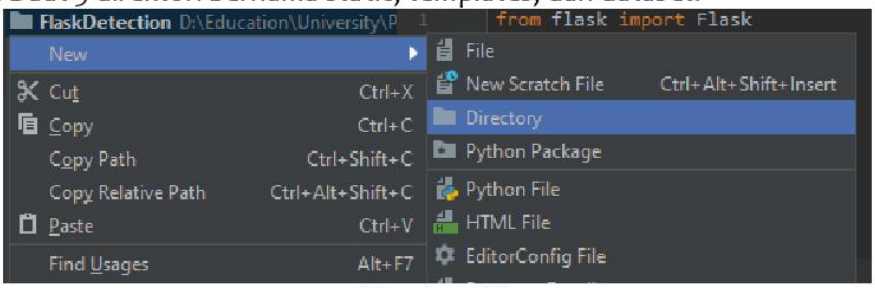
```
(base) D:\Education\University\Pelatihan\FlaskDetection>python -m flask run
 * Serving Flask app "app.py"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Gambar 28.



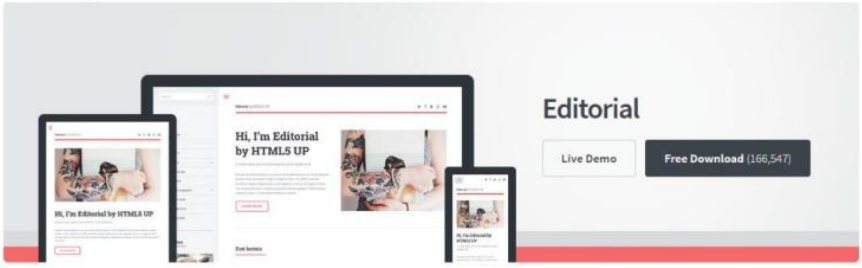
Gambar 29.

6. Klik kanan pada direktori FlaskDetection dan buat direktori baru seperti yang ditunjukan oleh Gambar 30. Buat 3 direktori bernama static, templates, dan dataset.



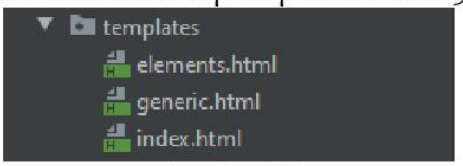
Gambar 30.

7. Download template html5Up yang akan kita gunakan sebagai template web kita pada html5up.net, untuk sementara kita gunakan template editor seperti yang ditunjukan oleh Gambar 31.



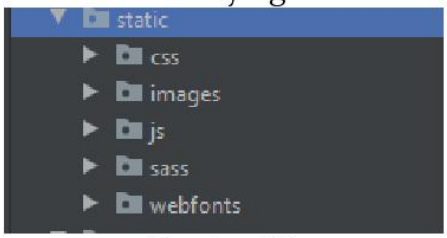
Gambar 31.

8. Buka File template yang telah didownload, kemudian masukan semua file berekstensi *.html ke direktori templates yang telah kita buat seperti pada Gambar 32.



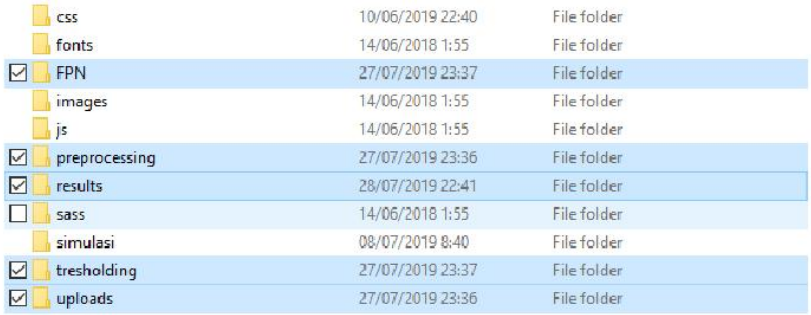
Gambar 32.

9. Masukan semua folder pada direktori assets dan direktori images dari file template yang telah kita download dan masukan ke direktori static yang telah kita buat seperti pada Gambar 33.



Gambar 33.

10. Buat Direktori FPN, preprocessing, results, tresholding, dan uploads pada direktori static seperti yang ditunjukan oleh Gambar 34.



Gambar 34.

11. Masukan direktori "keras_retinanet" yang telah kita gunakan pada saat training yang telah diclone dari link https://github.com/fizyr/keras-retinanet ke direktori FlaskDetection

12. Kemudian buka file app.py dan salin kode dibawah ini.

```
import keras
# import miscellaneous modules
import matplotlib.pyplot as plt
import cv2
cv2.__version__
import os
import sys
import numpy as np
import time
import tensorflow as tf


from flask import Flask, render_template, request, redirect, url_for,
send_from_directory, jsonify
from werkzeug import secure_filename
from keras_retinanet import models
from keras_retinanet.utils.image import read_image_bgr, preprocess_image,
resize_image
from keras_retinanet.utils.visualization import draw_box, draw_caption
from keras_retinanet.utils.colors import label_color
from keras.layers import MaxPooling2D,ZeroPadding2D,Conv2D,BatchNormalization
from keras.layers import Activation,initializers,Add

from io import StringIO
from PIL import Image
sys.path.append("..")

app = Flask(__name__)

app.config['UPLOAD_FOLDER'] = 'static/uploads/'
app.config['UPLOAD_SIMULASI'] = 'static/simulasi/'
app.config['RESULT_FOLDER'] = 'results/'
app.config['ALLOWED_EXTENSIONS'] = set(['png', 'jpg', 'jpeg'])

def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1] in app.config['ALLOWED_EXTENSIONS']
```

```python
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/upload', methods=['GET','POST'])
def upload():
    file = request.files['file']
    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        return redirect(url_for('uploaded_file',filename=filename))
        # return render_template('index.html')

@app.route('/uploads/<filename>')
def uploaded_file(filename):
    PATH_TO_TEST_IMAGES_DIR = app.config['UPLOAD_FOLDER']
    TEST_IMAGE_PATHS = [os.path.join(PATH_TO_TEST_IMAGES_DIR, filename.format(i))
for i in range(1, 2)]


    for image_path in TEST_IMAGE_PATHS:

        labels_to_names = {0: 'Car', 1: 'Terinfeksi'}
        image = read_image_bgr(image_path)


        # copy to draw on
        draw = image.copy()
        draw2 = image.copy()
        draw3 = image.copy()

        fileoriginal=app.config['UPLOAD_FOLDER']+filename
        filename.partition('.')
        filename, separator, extension = filename.partition('.')
        fileprepocess='static/preprocessing/'+filename+"-preprocess.jpg"
        filenameFPN=filename+"-FPN.jpg"


        # preprocess image for network
        preproc=preprocess_image(image)
        image = preprocess_image(image)
        image, scale = resize_image(image)

        #write Image for Preprocessing
        cv2.imwrite(fileprepocess,preproc)

        boxes,scores,labels=predictdata(image)

        # correct for image scale
        boxes /= scale

        filedrawbox=drawUntreshold(boxes,scores,labels,draw2,filenameFPN)
        filedrawtrbox=drawtreshold(boxes,scores,labels,draw3,filenameFPN)

        Car = 0
        for box, score, label in zip(boxes[0], scores[0], labels[0]):
            # scores are sorted so we can break    draw_box(draw, b, color=color)
```

```
            if score < 0.5:
                break

            color = label_color(label)

            b = box.astype(int)
            draw_box(draw, b, color=color, thickness=2)

            caption = "{} {:.3f}".format(labels_to_names[label], score)
            draw_caption(draw, b, caption)
            print(caption)

            if label == 0:
                Car = Car + 1


        print("Mobil= ", Car)

        file=filename+'-test.jpg'
        plt.figure(figsize=(20, 20))
        plt.axis('off')
        # plt.imshow(draw)
        cv2.imwrite('static/results/'+file,draw)
        # draw=plt.gcf()
        # draw.savefig('static/results/'+file,format='jpg')
        directoryfile="/static/results/"+file
        # plt.show()

        return jsonify(Name=directoryfile, OriginalFile=fileoriginal,
Preprocessing=fileprepocess,
                       Drawbox=filedrawbox, Treshold=filedrawtrbox, EritNormal=Car)


def get_session():
    config = tf.ConfigProto()
    config.gpu_options.allow_growth = True
    return tf.Session(config=config)

def predictdata(ImageFile):
    keras.backend.tensorflow_backend.set_session(get_session())

    model_path = os.path.join('dataset', sorted(os.listdir('dataset'),
reverse=True)[0])
    # model_path = os.path.join('snapshotsbaru',
sorted(os.listdir('snapshotsbaru'), reverse=True)[0])

    # load retinanet model
    model = models.load_model(model_path, backbone_name='resnet101')
    model = models.convert_model(model)
    print(model.summary())

    # labels_to_names = {0: 'Normal', 1: 'Terinfeksi'}

    # process image
    start = time.time()
    boxes, scores, labels = model.predict_on_batch(np.expand_dims(ImageFile,
axis=0))
```

```
    print("processing time: ", time.time() - start)
    return boxes,scores,labels

def drawUntreshold(boxes,scores,labels,draw,file):
    # visualize detections
    for box, score, label in zip(boxes[0], scores[0], labels[0]):
        # scores are sorted so we can break    draw_box(draw, b, color=color)

        color = label_color(label)

        b = box.astype(int)
        draw_box(draw, b, color=color, thickness=4)

    filename='static/FPN/'+file+'-FPN.jpg'
    cv2.imwrite(filename, draw)
    return filename

def drawtreshold(boxes,scores,labels,draw,file):
    # visualize detections
    for box, score, label in zip(boxes[0], scores[0], labels[0]):
        # scores are sorted so we can break    draw_box(draw, b, color=color)
        if score < 0.5:
            break

        color = label_color(label)

        b = box.astype(int)
        draw_box(draw, b, color=color, thickness=4)
    filenametres = 'static/tresholding/' + file + "-tres.jpg"
    cv2.imwrite(filenametres, draw)
    return filenametres

    filename='static/FPN/'+file+'-FPN.jpg'
    cv2.imwrite(filename, draw)
    return filename


if __name__ == '__main__':
    app.run(debug=True,host='0.0.0.0',port=5000)
```

13.Sesuaikan file index.html dengan kode dibawah ini termasuk kode javascript pada bagaian bawah.

```
<html>

<style>



</style>

<head>
```

```
    <title>Implementasi Object Detection </title>

    <meta charset="utf-8"/>

    <meta name="viewport" content="width=device-width, initial-scale=1, user-
scalable=no"/>

    <link rel="stylesheet" href="{{ url_for('static',filename='css/main.css')}}"/>

</head>

<body class="is-preload">



<!-- Wrapper -->

<div id="wrapper">



    <!-- Main -->

    <div id="main">

        <div class="inner">

            <!-- Header -->

            <header id="header">

                <h1 href="index.html" class="logo">CAR DETECTION RETINANET</h1>



            </header>



             <section id="banner">

                <div class="content">

                    <header>

                        <h1>Pilih Citra Uji<br/>
```

```
                    </header>

                    <form id="uploadform" method="post" enctype="multipart/form-
data">

                        <input type="file" name="file" accept="image/*"
id="inputFile1"><br/><br/>

                        <ul class="actions">

                            <li><a id="submit" class="button big">Proses
Citra</a></li>

                        </ul>

                    </form>

                </div>

                <span class="image object">

                    <img id="imagen" src="{{
url_for('static',filename='images/pic10.jpg')}}" alt=""/>

                </span>

            </section>



<!--            Process-->

            <section>

                <header class="major">

                    <h2>Hasil Proses</h2>

                </header>

                <div class="posts">

                    <article>

                        <a href="#" class="image"><img id="ImgPreprocessing"
src="{{ url_for('static',filename='images/pic01.jpg')}}" alt=""/></a>
```

```
                        <h3>Preprocessing</h3>

                        <ul class="actions">

                            <li><a id="btnPreprocessing" href="#"
class="button">More</a></li>

                        </ul>

                    </article>



                    <article>

                        <a href="#" class="image"><img id="ImgFPN1" src="{{
url_for('static',filename='images/pic03.jpg')}}" alt=""/></a>

                        <h3>Hasil FPN</h3>

                        <ul class="actions">

                            <li><a id="btnFPN1" href="#"
class="button">More</a></li>

                        </ul>

                    </article>

                    <article>

                        <a href="#" class="image"><img id="ImgTresh1" src="{{
url_for('static',filename='images/pic05.jpg')}}" alt=""/></a>

                        <h3>Hasil Regresi Box </h3>

                        <ul class="actions">

                            <li><a id="btnTresh1" href="#"
class="button">More</a></li>

                        </ul>

                    </article>
```

```
                    <article>

                        <a href="#" class="image"><img id="ImgKlasifikasi1" src="{{
url_for('static',filename='images/pic05.jpg')}}" alt=""/></a>

                        <h3>Hasil Klasifikasi </h3>

                        <ul class="actions">

                            <li><a id="btnHasil1" href="#"
class="button">More</a></li>

                        </ul>

                         <h2>Hasil Identifikasi :</h2>

                          <table style="margin: 8px 0px 0px;" border="1">

                        <tbody>

                        <tr>

                            <td>Jumlah Mobil Terdeteksi</td>

                            <td id="EritNormal"></td>

                        </tr>

                        </tbody>

                    </table>

                        <br>

                        <h3>Diagnosis: <h3 id="Diagnosis1"></h3> </h3>

                    </article>

                </div>

            </section>

        </div>

    </div>
```

```
    <!-- Sidebar -->

    <div id="sidebar">

        <div class="inner">



            <!-- Search -->

            <section id="search" class="alt">

                <form method="post" action="#">

                    <input type="text" name="query" id="query"
placeholder="Search"/>

                </form>

            </section>



            <!-- Menu -->

            <nav id="menu">

                <header class="major">

                    <h2>Menu</h2>

                </header>

                <ul>

                    <li><a href="/">Homepage</a></li>

                </ul>

            </nav>

            <!-- Section -->

            <section>

                <header class="major">
```

```
                    <h2>About Us</h2>
                </header>
                <p>Pelatihan Deep Learning </p>
            </section>

            <!-- Footer -->
            <footer id="footer">
                <p class="copyright">&copy;  All rights reserved.
                    Design: <a href="https://html5up.net">HTML5 UP</a>.</p>
            </footer>

        </div>
    </div>

</div>

<!-- Scripts -->
<script src="{{ url_for('static',filename='js/jquery.min.js')}}"></script>
<script src="{{ url_for('static',filename='js/browser.min.js')}}"></script>
<script src="{{ url_for('static',filename='js/breakpoints.min.js')}}"></script>
<script src="{{ url_for('static',filename='js/util.js')}}"></script>
<script src="{{ url_for('static',filename='js/main.js')}}"></script>
```

```
<script>

    function init() {

        var inputFile = document.getElementById('inputFile1');

        inputFile.addEventListener('change', mostrarImagen, false);

    }

    function mostrarImagen(event) {

        var file = event.target.files[0];

        var reader = new FileReader();

        reader.onload = function (event) {

            var img = document.getElementById('imagen');

            img.src = event.target.result;

        //    img.width = 500;

          //  img.height = 500;

        }

        reader.readAsDataURL(file);

    }

    window.addEventListener('load', init, false);



    $(document).ready(function(){

      $('#submit').click(function() {

        event.preventDefault();

        var form_data = new FormData($('#uploadform')[0]);

        $.ajax({
```

```
            type: 'POST',
            url: '/upload',
            data: form_data,
            contentType: false,
            processData: false,
            dataType: 'json'
        }).done(function(data, textStatus, jqXHR){
            console.log(data);
            console.log(textStatus);
            console.log(jqXHR);
            console.log('Success!');
            $("#EritNormal").text(data['EritNormal']);
            $('#imageresult').attr('src',data['Name']);
            $('#ImgPreprocessing').attr('src',data['Preprocessing']);
            $('#ImgFPN1').attr('src',data['Drawbox']);
            $('#ImgTresh1').attr('src',data['Treshold']);
            $('#ImgKlasifikasi1').attr('src',data['Name']);
            $('#ImgPreprocessing').attr('href',data['Preprocessing']);
            $('#ImgFPN1').attr('href',data['Drawbox']);
            $('#ImgKlasifikasi1').attr('src',data['Name']);
            $('#btnPreprocessing').attr('href',data['Preprocessing']);
            $('#btnFPN1').attr('href',data['Drawbox']);
            $('#btnTresh1').attr('href',data['Treshold']);
            $('#btnHasil1').attr('href',data['Name']);
```

```
            $("html, body").animate({ scrollTop: $("#EritNormal").offset().top});

        }).fail(function(data){
            alert('error!');
        });
    });
    });



</script>

<!--export FLASK_APP=app.py-->
<!--python -m flask run-->
</body>
</html>
```

14. Masukan file “resnet101_csv_100.h5” pada direktori dataset yang telah kita buat.
15. Buka terminal pada pycharm dan jalankan Flask app seperti pada langkah 4.