SOURCE CODE

```
import cv2
import numpy as np
import random
import os
from PIL import Image
import time
import imutils
import telepot
from tensorflow.keras.models import load_model
token = '6199878006:AAFcjwAYiEGH5eAiA1x2dUpabggFiYTsTa8' # telegram token
receiver_id = 1838134505 # https://api.telegram.org/bot<TOKEN>/getUpdates
bot = telepot.Bot(token)
os.environ['TF_FORCE_GPU_ALLOW_GROWTH'] = 'true'
net = cv2.dnn.readNet("yolov3-custom_7000.weights", "yolov3-custom.cfg")
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)
model = load_model('helmet-nonhelmet_cnn.h5')
```

```
print('model loaded!!!')

#cap = cv2.VideoCapture(1)

cap = cv2.VideoCapture('video.mp4')

COLORS = [(0,255,0),(0,0,255)]

##fourcc = cv2.VideoWriter_fourcc(*"XVID")

##writer = cv2.VideoWriter('output.avi', fourcc, 5,(888,500))

###writer = cv2.VideoWriter('output.avi',(frame.shape[1], frame.shape[0]))

##writer.open()

def helmet_or_nohelmet(helmet_roi):

    try:

        helmet_roi = cv2.resize(helmet_roi, (224, 224))

        helmet_roi = np.array(helmet_roi,dtype='float32')

        helmet_roi = helmet_roi.reshape(1, 224, 224, 3)

        helmet_roi = helmet_roi/255.0

        return int(model.predict(helmet_roi)[0][0])

    except:

            pass

layer_names = net.getLayerNames()

#output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]
```

```
output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]

ret = True

while ret:

    ret, img = cap.read()

    img = imutils.resize(img,height=500)

    # img = cv2.imread('test.png')

    height, width = img.shape[:2]

    blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)

    net.setInput(blob)

    outs = net.forward(output_layers)

    confidences = []

    boxes = []

    classIds = []

    for out in outs:

        for detection in out:

            scores = detection[5:]

            class_id = np.argmax(scores)

            confidence = scores[class_id]
            if confidence > 0.3:

                center_x = int(detection[0] * width)

                center_y = int(detection[1] * height)

                w = int(detection[2] * width)

                h = int(detection[3] * height)

                x = int(center_x - w / 2)

                y = int(center_y - h / 2)

                boxes.append([x, y, w, h])

                confidences.append(float(confidence))

                classIds.append(class_id)

    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

    for i in range(len(boxes)):

        if i in indexes:

            x,y,w,h = boxes[i]

            color = [int(c) for c in COLORS[classIds[i]]]

            # green --> bike

            # red --> number plate

            if classIds[i]==0: #bike
```

```
                helmet_roi = img[max(0,y):max(0,y)+max(0,h)//4,max(0,x):max(0,x)+max(0,w)]
            else: #number plate
                x_h = x-60
                y_h = y-350
                w_h = w+100
                h_h = h+100
                cv2.rectangle(img, (x, y), (x + w, y + h), color, 7)
                # h_r = img[max(0,(y-330)):max(0,(y-330 + h+100)) , max(0,(x-80)):max(0,(x-80 + w+130))]
                if y_h>0 and x_h>0:
                    h_r = img[y_h:y_h+h_h , x_h:x_h +w_h]
                    c = helmet_or_nohelmet(h_r)
                    print('helmet or no-helmet')
                    print(c)
##                      if c == 1:
##                          cv2.imwrite('test.jpg', img)
##                          bot.sendMessage(receiver_id, 'NO HELMET') # send a activation message to telegram receiver id
##                              bot.sendPhoto(receiver_id, photo=open('test.jpg', 'rb')) # send
message to telegram

                cv2.putText(img,['helmet','no-helmet'][c],(x,y-
100),cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0),2)

                 cv2.rectangle(img, (x_h, y_h), (x_h + w_h, y_h + h_h),(255,0,0), 10)

    #writer.write(img)

  cv2.imshow("Image", img)

    if cv2.waitKey(1) == 27:

        break

writer.release()

cap.release()

cv2.waitKey(0)

cv2.destroyAllWindows()
```

TRAINING CODE

```
#from tensorflow.compat.v1 import ConfigProto
#from tensorflow.compat.v1 import InteractiveSession
#config = ConfigProto()
#config.gpu_options.per_process_gpu_memory_fraction = 0.5
#config.gpu_options.allow_growth = True
#session = InteractiveSession(config=config)

import tensorflow as tf
print(tf.__version__)



# import the libraries as shown below

from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import numpy as np
from glob import glob
#import matplotlib.pyplot as plt

from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import load_model
import matplotlib.pyplot as plt


# re-size all the images to this
IMAGE_SIZE = [224, 224]
train_path = 'Datasets/train'
valid_path = 'Datasets/test'
vgg16 = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet',
include_top=False)
print("Stage 1")
for layer in vgg16.layers:
    layer.trainable = False
 # useful for getting number of output classes
folders = glob('Datasets/train/*')
print("Stage 2")
x = Flatten()(vgg16.output)
print("x")
print (x)
prediction = Dense(len(folders), activation='softmax')(x)
print("prediction"
print (prediction)
print("Stage 3")
model = Model(inputs=vgg16.input, outputs=prediction)
```

```
model.summary()
print("Stage 4")
model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)
print("Stage 5")
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)
print("test_datagen")
print (test_datagen)
print("Stage 6")
training_set = train_datagen.flow_from_directory('Datasets/train',
                                                 target_size = (224, 224),
                                                 batch_size = 4,
                                                 class_mode = 'categorical')

print("training_set")
print (training_set)
print("Stage 7")
test_set = test_datagen.flow_from_directory('Datasets/test',
                                            target_size = (224, 224),
                                           batch_size = 4,
                                           class_mode = 'categorical')

print("test_set")
print (test_set)
print("Stage 7")
print(len(training_set))
print(len(test_set))
#NN.fit_generator
#model.fit
FE_r = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=20,
  steps_per_epoch=2,
  validation_steps=2
)
print("FE_r")
print (FE_r)
print("Stage 8")
plt.plot(FE_r.history['loss'], label='train loss')
plt.plot(FE_r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
#plt.savefig('LossVal_loss')
print("Stage 9")
# plot the accuracy
```

```
#plt.plimage = cv2.imread(image_file)
#lab_image = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)
#cv2_imshow(lab_image)ot(FE_r.history['accuracy'], label='train acc')
#plt.plot(FE_r.history['val_accuracy'], label='val acc')
#plt.legend()
#plt.show()
#plt.savefig('accuracy')
print("Stage 10")
model.save('model_TCE_new_1.h5')
print("Stage 11")
```