AKDENIZ UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER SCIENCE & ENGINEERING DEPARTMENT

IMAGE PROCESSING

"Face Mask Detection" PROJECT

FINAL REPORT

Project ID: 14

PROJECT GROUP MEMBERS

Ayşe Nida DİNÇ - 20160808047
Asrın ÖZTİN - 20170808042

INSTRUCTOR: ASST.PROF.DR. Mustafa Berkay YILMAZ

ANTALYA, DECEMBER 2020

CONTENTS

1. SURVEY

Mask detection is a new problem that we face recently so, there is not that much different methods to determine if there is a mask in a picture or in a video. The methods to detect the mask can be simply seperated as “face detection + adge detection” and “training the program + deep learning”.

In the first case we can talk about the history of face detection which is a long-time problem when compared to mask detection. [5]

We can say that recently face detection algorithms are having much more importance and so, engineers work on that to improve it and to get better results. The main problem they are trying to figure out is to detect even if the environment is complex such as cluttered backgrounds and low quality images. Some of the algorithms that used are still too computationally expensive to be apply for a real time processing. However, this can be fixed with coming improvements in computer hardware technology.

We can analyse methods seperated as “feature-based” and “image-based”.

Fetaure based methods can be used in real-time systems where color and motion is available. The main problem with that is, these methods cannot always provide visual cues to focus attention due to exhaustive multiresolution window scanning cannot always be preferable. In that case, the common approach to fix that problem is “skin color detection”.

Image-based approaches are the most powerfull techniques to process gray-scale images. Sung and Poggio and Rowley et al. Developed an algorithm on that topic and that algorithms is still can be used because it is still comperable with recent common algorithms.

The high computional cost can be decreased with avoiding multiresolution window scanning with combining these two approachs with using visual clues like skin color when we are trying to find the face.

To conclude, detecting a face is still a hard problem to solve, considering the changes in faces over time like facial hair, glass usage, etc.

In the second case which we used to determine if there is a mask or not, we basically followed the steps which are: collect a dataset with and without mask, load the dataset and train the program with selecting which ones have mask and which ones do not, serialize model to disk, decet faces and extract them ROI, apply our model to detect faces and check for a mask and finally show the result.

It is much more robust solution because in that case we did not have to deal with the problems like cluttered background sor low quality images, we also did not deal with collecting a dataset to use visual clues about the skin color. To sum up, we did not focus on the face but the mask.

The other specific reason that we picked that one is to learn the recent technological approaches to problems when compared to the old ones.

2. DATASET

For first method (face mask detection using keras/tensorflow, python, opencv and mobilenet) we have used 3835 images for dataset. All images are extracted from Kaggle datasets and RMFD dataset, Bing Search API. And all images are real. Images from all three sources are equated. The ratio of masked faces to unmasked faces indicates that the data set is balanced.

If our dataset and model require a lot of training, that is, if the model has too many parameters to adjust, then we have to use a larger dataset for training, which is our case.

We needed to divide our dataset into two parts; train dataset & test dataset. And first of all, we have decided to use 60% of dataset to training the model. And other 40% of the dataset to use for testing the model. After that, we have decided to retrain & retest our model by using 80% of the dataset for training the model & 20% for the test. We want to see difference of two training's accurancy, recall & precision.

The dataset has 3835 images of faces. All images collected from following sources: [3] [4]

- Kaggle datasets
- Rmfd dataset
- Bing search API

And each image provides only 1 face. Dataset is balanced & divided into two categories:

- without_mask : 1916 images
- with_mask : 1919 images

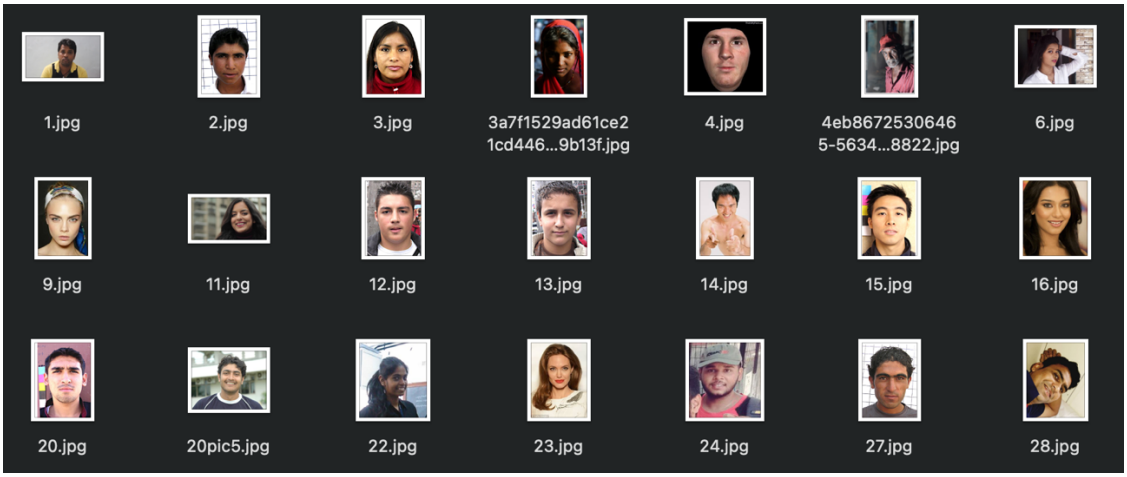

Figure 1: Dataset sample of without_mask

Figure 2: Dataset sample of with_mask

3. EXPERIMENTATION

First of all, we have used 60% of the dataset to training the model. And we tested the model using 40% of the dataset.. And we archived 0.85 accuracy using mobileNet & max-pooling method. And the equations of precision, recall and accuracy are as follows:

$$Precision = \frac{True\ Positives}{Positives + False\ Positives}$$

$$Recall = \frac{True\ Positives}{Positives + False\ Negatives}$$

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Positives + Negatives}$$

classification report:

	precision	recall	f1-score	support
with_mask	0.87	0.81	0.84	384
without_mask	0.83	0.88	0.85	386
accuracy			0.85	770
macro avg	0.85	0.85	0.85	770
weighted avg	0.85	0.85	0.85	770

Figure 8: Accurancy, recall & precision 60% of the dataset

After that, we have used 80% of the dataset for training the model. After training we archieved 0.9909 accurancy using mobileNet & max-pooling method.

```
Epoch 17/20
95/95 [==============================] - 141s 1s/step - loss: 0.0352 - accuracy: 0.9933 - val_loss: 0.0389 - val_accuracy: 0.9896
Epoch 18/20
95/95 [==============================] - 137s 1s/step - loss: 0.0300 - accuracy: 0.9913 - val_loss: 0.0390 - val_accuracy: 0.9896
Epoch 19/20
95/95 [==============================] - 145s 2s/step - loss: 0.0298 - accuracy: 0.9904 - val_loss: 0.0367 - val_accuracy: 0.9883
Epoch 20/20
95/95 [==============================] - 129s 1s/step - loss: 0.0280 - accuracy: 0.9914 - val_loss: 0.0340 - val_accuracy: 0.9909
[INFO] evaluating network...
              precision    recall  f1-score   support

   with_mask       0.99      0.99      0.99       383
without_mask       0.99      0.99      0.99       384

    accuracy                           0.99       767
   macro avg       0.99      0.99      0.99       767
weighted avg       0.99      0.99      0.99       767

[INFO] saving mask detector model...
nidadinc@nida-MacBook-Pro face2 % python3 detect_mask_video.py
```

Figure 9: Accurancy, recall & precision 80% of the dataset

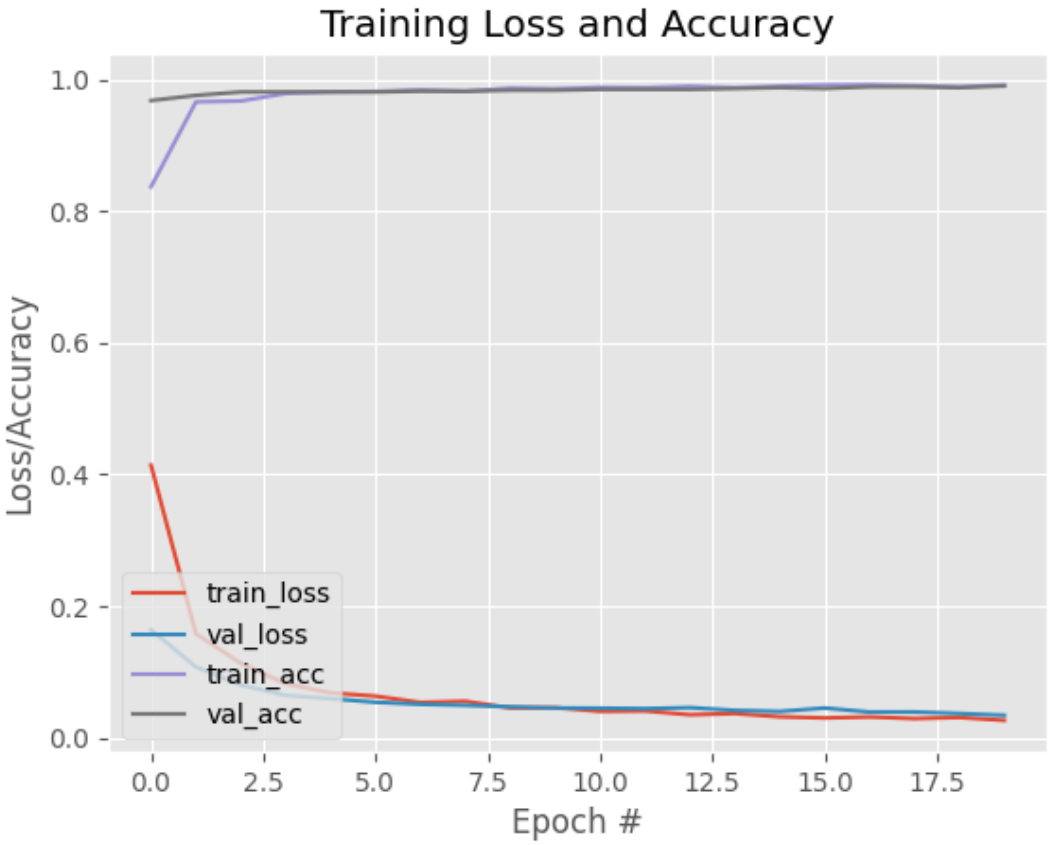

Figure 10: Accurancy & loss after training 80% of the dataset

As a result of training, 99% precision and 99% recall in the segmentation of face detection When we use noisy images, accurancy dropped to 0.8689.

4. SOURCE CODE

4.1 Face Mask Detection using Keras/Tensorflow, Python, OpenCV and MobileNet [1]

a. Train the detector Python codes (only important parts) :

```
# Determine initial learning rate,
INIT_LR = 1e-4
#number of epochs to be trained,
EPOCHS = 20
#and group size
BS = 32
# we applied less learning rate because less learning rate means loss will be calculated
properly.



# get the list of images from dataset directory,
DIRECTORY = r"/Users/nidadinc/Desktop/face2/dataset"
CATEGORIES = ["with_mask", "without_mask"]

#then start the data list
print("[INFO] PLEASE WAIT, LOADING...")

data = []
labels = []

# withmask and without mask images will append to data array and categories will append to
labels array
for category in CATEGORIES:
    #directory & path declared
    path = os.path.join(DIRECTORY, category)

    for img in os.listdir(path):
       img_path = os.path.join(path, img)

       #all images will be 224 x 224
       image = load_img(img_path, target_size=(224, 224))

       #images to array using imagetoarray function
       image = img_to_array(image)

       # mobileNet used for model so, we used preprocess_input fuction
       image = preprocess_input(image)

       data.append(image)
       labels.append(category) #with_mask and without_mask

# data is numerical values, but labels are alphbetic variables, withmask and withoutmask.
# We should convert them to numerical values by using LabelBinarizer() method. that comes
from the sklearn.Preprocesser model


#train & test data 0.2 for testing and 0.8 for training


# create training image generator for data augmentation
# ImageDataGenerator() creates many images from single image
# by changing some of the properties like rotating, shifting etc.
aug = ImageDataGenerator(

   horizontal_flip=True,
   height_shift_range=0.2,
   width_shift_range=0.2,
   rotation_range=20,
   shear_range=0.15,
   zoom_range=0.15,
   fill_mode="nearest")
```

```
# MobileNetV2 network loading

#it means height 224, weight 224, and 3 channels R,G & B
baseModel = MobileNetV2(weights="imagenet", include_top=False,
   input_tensor=Input(shape=(224, 224, 3)))



# create the model head

headModel = baseModel.output
#creating the pooling 7 by 7
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
#doing flatten this layers
headModel = Flatten(name="flatten")(headModel)
# dense layer and relu is non-linear use cases
headModel = Dense(128, activation="relu")(headModel) headModel = Dropout(0.5)(headModel)
#dropout for avoiding
headModel = Dense(2, activation="softmax")(headModel)


# loop through all base model layers and
# freeze them so they don't update during initial training
for layer in baseModel.layers:
   layer.trainable = False

# serializing the model
print("[INFO] mask detector model is saving...")
model.save("mask_detector.model", save_format="h5")
```

b. Detect mask from real-time video Python codes (only important parts):

```
def detect_predictMask(frame, faceNet, maskNet):
   # capture the frame dimensions
   (h, w) = frame.shape[:2]
   # and create a blob
   blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
      (104.0, 177.0, 123.0))


   # loop the detections
   for i in range(0, detections.shape[2]):
      # removing confidence (i.e. probability) associated with detection
      confidence = detections[0, 0, i, 2]

      # Filter out weak perceptions by ensuring that trust is
      # higher than minimum confidence
      if confidence > 0.5:
         # compute the (x, y) coordinates of the bounding box for the object
         box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
         (startX, startY, endX, endY) = box.astype("int")

         # make sure the bounding boxes is same as the frame dimesions
         (startX, startY) = (max(0, startX), max(0, startY))
         (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

         # Extract face ROI
         face = frame[startY:endY, startX:endX]

         # convert from BGR to RGB channel
         face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)

         # resize to 224x224
         face = cv2.resize(face, (224, 224))

         #and preprocess
         face = img_to_array(face)
         face = preprocess_input(face)

         # add the face & boxes
         faces.append(face)
```

```
        locs.append((startX, startY, endX, endY))

# Upload our serialized face detector model to detect faces from disk
prototxtPath = r"/Users/nidadinc/Desktop/face2/face_detector/deploy.prototxt"
weightsPath =
r"/Users/nidadinc/Desktop/face2/face_detector/res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# install the model at disk

# go the frames in the video stream
while True:
   # frame the video stream and resize it to a maximum of 400 pixels width
   frame = vs.read()

   frame = imutils.resize(frame, width=400)

   # Into the frame detect faces & determine the person with mask or without mask.
   (locs, preds) = detect_predictMask(frame, faceNet, maskNet)

   # loop over detected face positions and their corresponding positions
```

5. REFERANCES

[1]- https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/

[2]- https://www.mygreatlearning.com/blog/real-time-face-detection/

[3]- https://drive.google.com/drive/folders/1XDte2DL2Mf_hw4NsmGst7QtYoU7sMBVG

[4]- https://github.com/X-zhangyang/Real-World-Masked-Face-Dataset

[5]- https://www.cin.ufpe.br/~rps/Artigos/Face%20Detection%20-%20%20A%20Survey.pdf