Ψηφιακή Επεξεργασία Εικόνας

Μαυροπουλος
Ανδρεας
ΑΜ: 217129

αναγνώριση και ταυτοποίηση

Το θέμα το οποίο διάλεξα είναι η αναγνώριση και η ταυτοποίηση, καθώς πολλές εταιρίες έχουν αρχίσει να ασχολούνται με την αναγνώριση μεσώ χρήσης machine learning, neural networks και computer vision. Η χρήση επεξεργασίας εικόνων βοηθά λοιπόν στην δημιουργία πιο ακριβή (machine learning) μοντέλων τα οποία στην συνεχεία μπορούν να χρησημοποιηθουν σε μια πληθώρα από εφαρμογές εφόσον υπάρχει στήριξη.

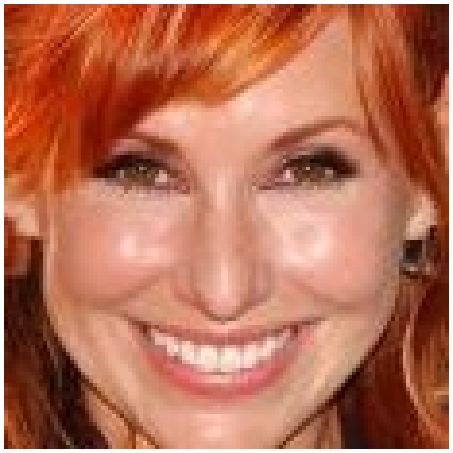
Ένα πρόβλημα που αποφάσισα να λύσω είναι η αναγωριση προσώπου σε 2 τάξης: Φοράει Μάσκα, Δεν φοράει Μάσκα. Το να φοράς μάσκα είναι ακόμα μια απαιτούμενη ενεργεία και η αυτοματοποιση αναγωρισης θα βοηθήσει τα καταστήματα να ξέρουν ποτέ μπορούν να δεχτούν ανθρώπους και ποτέ όχι.

Το φωτογραφικό υλικό που χρησημοποιησα ήταν από το kaggle, μια σελίδα που προσφέρει ητε φωτογραφικό ητε γραπτό υλικό για προσωπική χρήση. Οι εικόνες έχουν επιλεχθεί από ειδικούς, οι πήγες τους ήταν κυρίως από τα μέσα μαζικής ενημέρωσης, διαφημησεις open-source stock εικόνες και φωτογραφίες από διάσημους ανθρώπους. Η επιλογή των εικόνων έχει γίνει με βάση το πρόβλημα που θέλουν να επιλυθεί από την χρήση τους.

* https: //www. kaggle. com/andrewmvd/face-mask-detection - 852 unlabeled Photos - για testing
Παράδειγμα φωτογραφίας:

* https: //www. kaggle. com/ashishjangra27/face-mask-12k-images-dataset - ~12k labeled Photos - για training
Παράδειγματα φωτογραφίων:
WithMask:

WithouMask:



Για αυτήν την εργασία χρησημοποιησα το google colab το οποίο προσφέρει τα κατάλληλα υλικά για την δημιουργία μοντέλων. Μέσα από αυτό χρησημοποιησα python μαζί με matplotlib, numpy, pandas, computer vision και tensorlflow όλα από τα οποία είναι open source. Για την επεξεργασία εικόνων χρησημοποιησα το ImageDataGenerator το οποίο είναι ένα εργαλείο που έρχεται μαζί με το tensorflow.

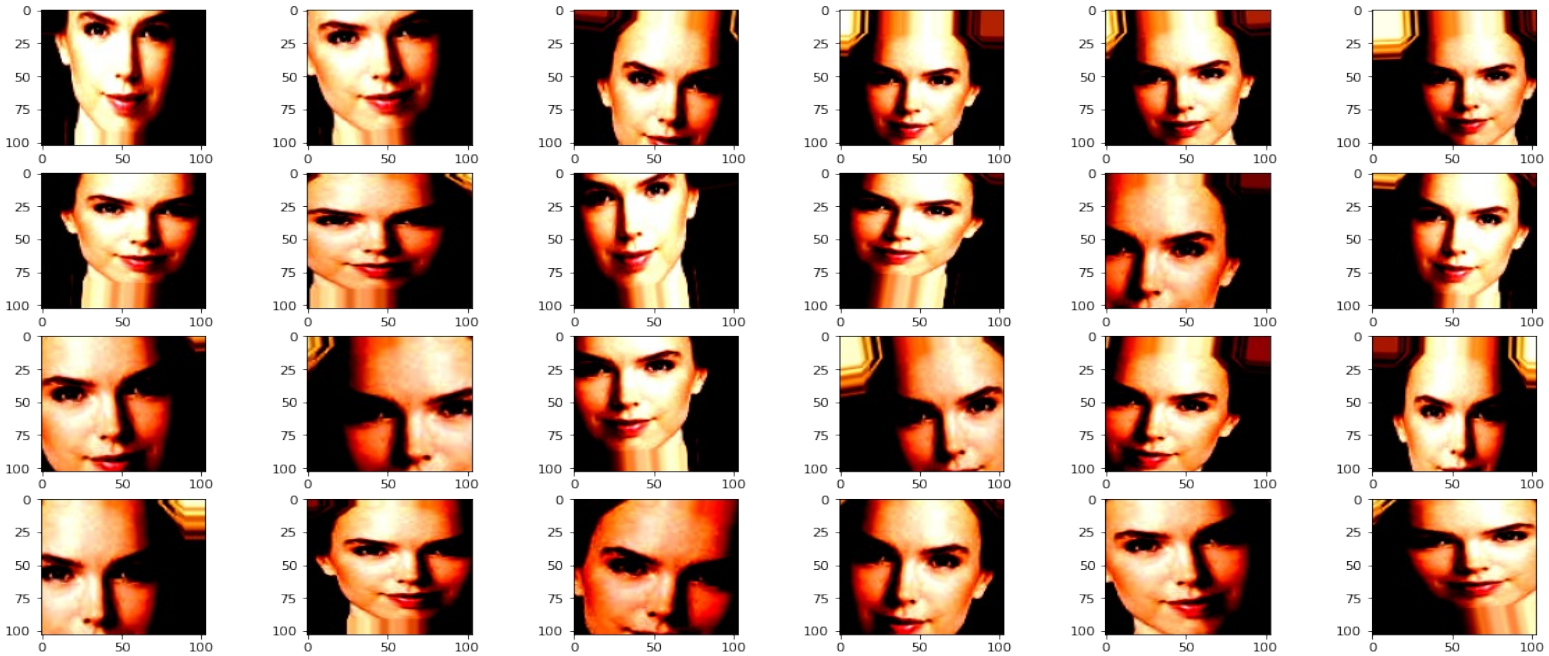
Το ImageDataGenerator:

```
# Training Image Generator
train_datagen = ImageDataGenerator(rescale= 1./255, # Normalizing our data,
                                   rotation_range = 10, # Rotates image
                                   width_shift_range = 0.2, # Slightly shifts image
                                   height_shift_range=0.2,  # Slightly shifts image
                                   zoom_range=0.25,         # Slight zoom to image
                                   horizontal_flip=True,    # Mirroring effect
                                   samplewise_center=True,  #  Set each sample mean to 0.
                                   samplewise_std_normalization=True, # Divide each input by its std.
                                   fill_mode = "nearest")            # Fills points outside of boundaries

```

Πριν γίνει edit:

Μετα το edit:



Παρατηρούμε πως από 1 εικόνα φτιάξαμε 24 μεσώ του ImageDataGenerator, βλέπουμε από το κομμάτι κωδικά πως έγινε η κάθε φωτογραφία και τι τεχνική χρησημοποιηθηκε για αυτήν. Βλέπουμε πως αρχικά η φωτογραφία έγινε normalized δηλαδή όλο η φωτογραφία μεταμορφώθηκε σε ένα array από 0 και 1, αυτό γίνεται για διευκόλυνση και για να μπορεί να τρέξει πιο γρήγορα το πρόγραμμα. Στην συνεχεία κάναμε rotate την εικόνα επιτα την κουνήσαμε μερικούς πόντους αριστερά και δεξιά και μετά πάνω και κάτω. Μέτα από αυτό κάναμε ένα μικρό zoom προς το κέντρο της. Και στο τέλος την κάναμε να έχει ένα mirroring effect. Όλα αυτά σε συνδυασμό κατάφεραν να κάνουν το dataset μας ουσιαστικά 24 φόρες μεγαλύτερο. Κάνοντας δηλαδή έστω και μια μικρή επεξεργασία στην κάθε εικόνα θα μπορέσουμε να φτιάξουμε ένα πιο ακριβή μοντέλο. Για την επεξεργασία εικόνας θα μπορούσαν να χρησημοποιηθουν πολλά διαφορετικά εργαλεία όπως πχ το Gimp με αποτέλεσμα να μην υπάρχει το drag effect στης εικόνες, αλλά θα έπρεπε να γίνει χειροκινητα για κάθε μια φωτογραφία. Επειδή λοιπόν έχουμε datasets που αποτελούνται από πάνω από 12 χιλιάδες φωτογραφίες είναι προτιμότερο να χρησημοποιηθει το ImageDataGenerator ακόμα και αν έχει το drag effect.

Μέτα από την επεξεργασία των εικόνων έχει σειρά η δημιουργία μοντέλου. Το μοντέλο που χρησημοποιησα ήταν το DenseNet201 ένα pre-trained μοντέλο πάνω στο ImageNet dataset το οποίο έχει πάνω από 1 εκκατομυριο εικόνες. Χρησημοποιησα δηλαδή transfer learning και χρησημοποιησα αυτό το μοντέλο πάνω στα δεδομένα που φτιάξαμε εμείς πάνω στης τάξης που φτιάξαμε (Φοράει Μάσκα, Δεν φοράει Μάσκα). Για να "μάθει" το μοντέλο πάνω στα δεδομένα μας χρειάστηκε περίπου 2 ώρες αλλά βλέπουμε ότι η ακρίβεια του σχεδόν είναι 99% και στο test set και στο validation set.

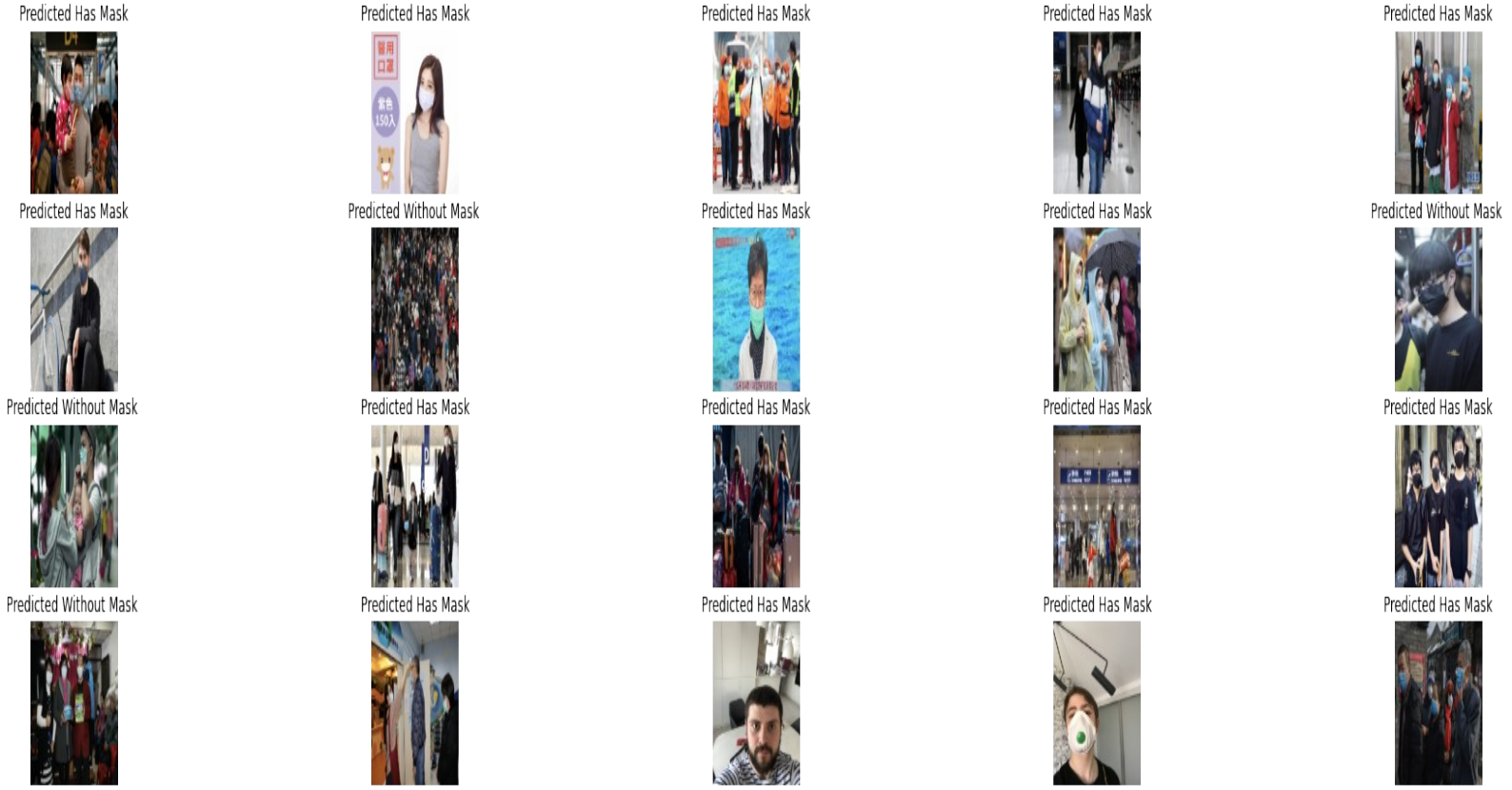
Time to fit our model to the data>

```
history_1 = model_1.fit(train_set,
                        epochs = 8 ,
                        steps_per_epoch = len(train_set),
                        validation_data = validation_set,
                        callbacks = [callback],
                        verbose = 1)
```

```
Epoch 1/8
157/157 [==============================] - 3064s 19s/step - loss: 0.1043 - accuracy: 0.9714 - val_loss: 0.0723 - val_accuracy: 0.9850
Epoch 2/8
157/157 [==============================] - 125s 793ms/step - loss: 0.0555 - accuracy: 0.9828 - val_loss: 0.0612 - val_accuracy: 0.9850
Epoch 3/8
157/157 [==============================] - 126s 802ms/step - loss: 0.0482 - accuracy: 0.9859 - val_loss: 0.0578 - val_accuracy: 0.9862
Epoch 4/8
157/157 [==============================] - 124s 791ms/step - loss: 0.0426 - accuracy: 0.9865 - val_loss: 0.0475 - val_accuracy: 0.9850
Epoch 5/8
157/157 [==============================] - 124s 791ms/step - loss: 0.0419 - accuracy: 0.9879 - val_loss: 0.0448 - val_accuracy: 0.9875
Epoch 6/8
157/157 [==============================] - 124s 790ms/step - loss: 0.0353 - accuracy: 0.9893 - val_loss: 0.0450 - val_accuracy: 0.9862
Epoch 7/8
157/157 [==============================] - 124s 790ms/step - loss: 0.0349 - accuracy: 0.9881 - val_loss: 0.0433 - val_accuracy: 0.9875
Epoch 8/8
157/157 [==============================] - 124s 791ms/step - loss: 0.0337 - accuracy: 0.9893 - val_loss: 0.0433 - val_accuracy: 0.9875
```

Τώρα εφόσον τελειώσαμε το μοντέλο έχει έρθει η ώρα να δούμε πως τα πάει σε labeled φωτογραφίες.(Οπου πράσινο το μοντέλο εχει ματντεψει σωστά)

Ώρα να το δοκιμάσουμε σε φωτογραφίες που δεν έχει ξαναδεί:

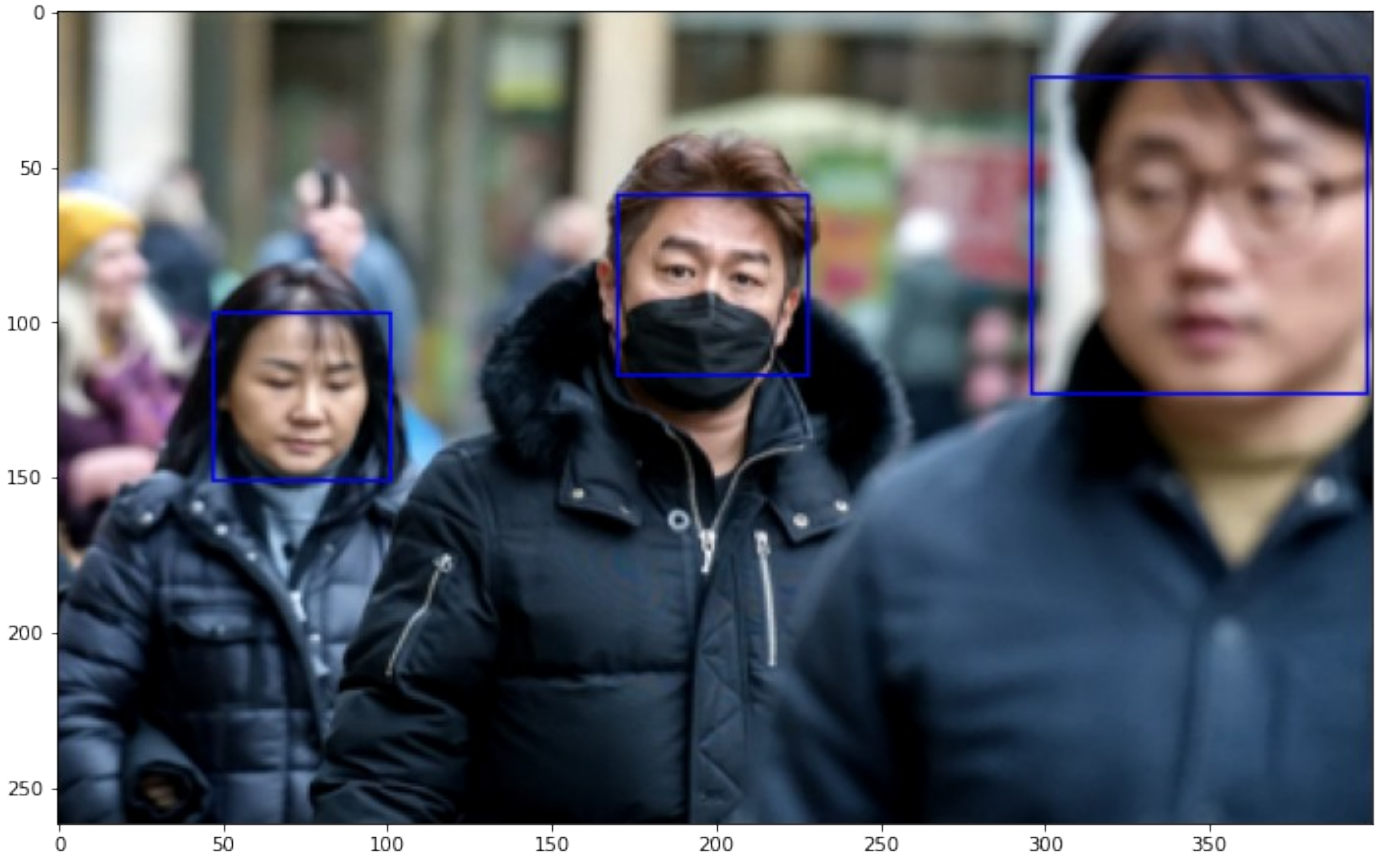


Βλέπουμε ότι το πρόγραμμα έχει πρόβλημα στο να μαντεύει για φωτογραφίες πολλών ανρθωπον και επίσης ότι μπορεί να μπερδέψει τα μούσια για πρόσωπο. Για να το φτιάξω λοιπόν αυτό χρησημοποιησα computer vision μαζί με το haarcascade για να αναγνωρίσει πρόσωπα. Στην συνεχεία να τα κάνει crop και να μαντεύει το μοντέλο μόνο στο cropped κομμάτι προσώπου.

Since the model is complete its time to add face detection using haar cascade

```
[ ]  1 face_model = cv2.CascadeClassifier("/content/drive/MyDrive/Face_Mask_Detection/haarcascades/haarcascade_frontalface_default.xml")
     2 img = cv2.imread('/content/drive/MyDrive/Face_Mask_Detection/images/Actual_Images/maksssksksss244.png')
     3
     4
     5 img = cv2.cvtColor(img, cv2.IMREAD_GRAYSCALE)
     6
     7 faces = face_model.detectMultiScale(img,scaleFactor=1.1, minNeighbors=4) #returns a list of (x,y,w,h) tuples
     8
     9 out_img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR) #colored output image
    10
    11 #plotting
    12 for (x,y,w,h) in faces:
    13     cv2.rectangle(out_img,(x,y),(x+w,y+h),(0,0,255),1)
    14 plt.figure(figsize=(12,12))
    15 plt.imshow(out_img);
```

Το πρώτο αποτέλεσμα του haarcascade

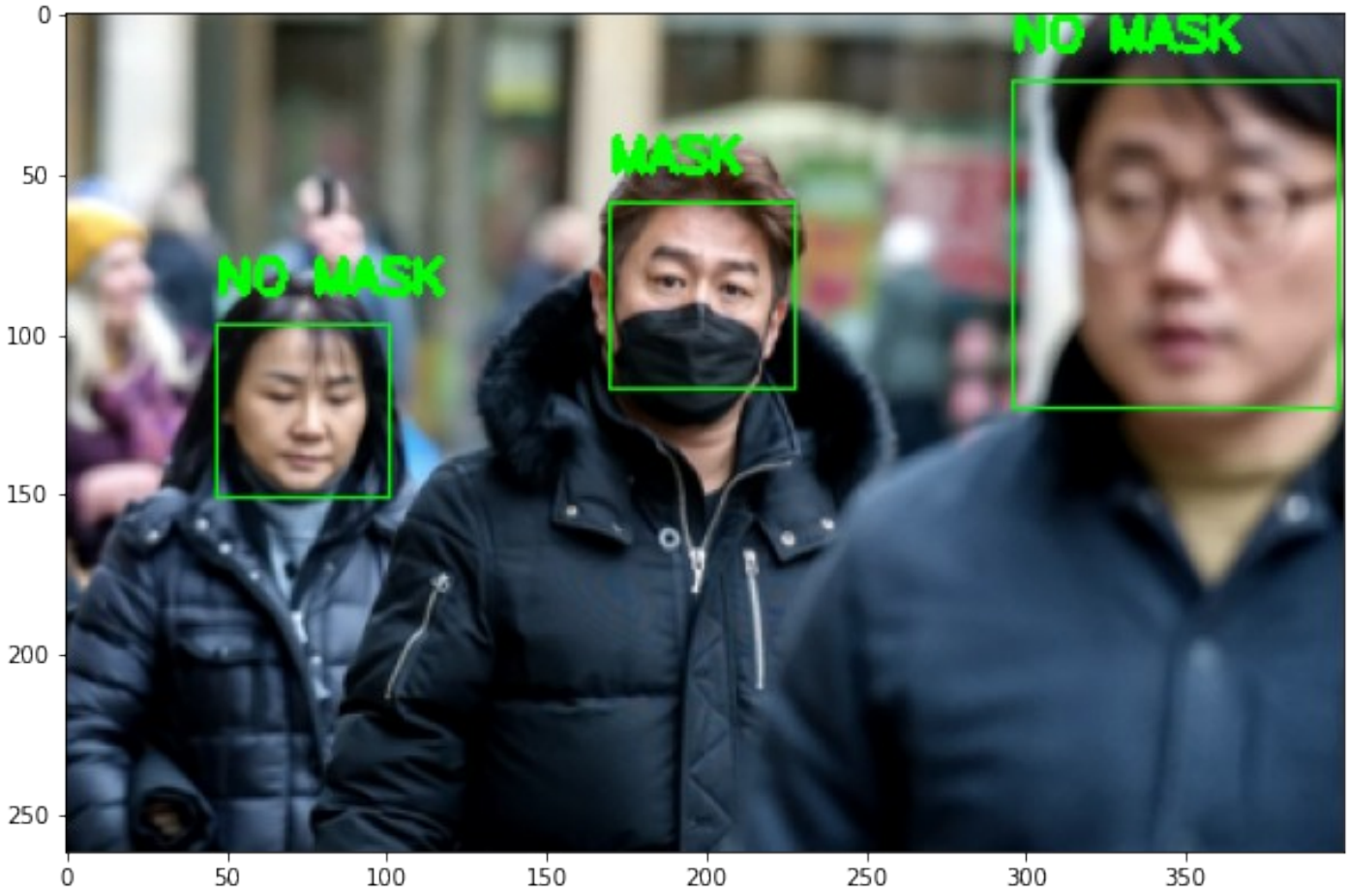


Face Detection + Model predcitions:

```
mask_label = {0:'MASK',1:'NO MASK'}
dist_label = {0:(0,255,0),1:(255,0,0)}

if len(faces)>= 1:
    label = [0 for i in range(len(faces))]
    new_img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR) #colored output image

    for i in range(len(faces)):
        (x,y,w,h) = faces[i]
        crop = new_img[y:y+h,x:x+w]
        crop = cv2.resize(crop,(128,128))
        crop = np.reshape(crop,[1,128,128,3])/255.0
        mask_result = model_1.predict(crop)
        cv2.putText(new_img,mask_label[mask_result.argmax()],(x, y-10),cv2.FONT_HERSHEY_SIMPLEX,0.5,dist_label[label[i]],2)
        cv2.rectangle(new_img,(x,y),(x+w,y+h),dist_label[label[i]],1)
    plt.figure(figsize=(10,10))
    plt.imshow(new_img)
```



για το ολοκληρωμενο προτζεκτ:
(https://github.com/adrikosm/university_projects/tree/main/Photo_Editing/%CE%95%CE%BE%CE%B1%CE%BC%CE%B7%CE%BD%CE%B9%CE%B1%CE%B9%CE%B1%20%CE%B5%CF%81%CE%B3%CE%B1%CF%83%CE%B9%CE%B1)

Βλέπουμε τώρα πόσο καλύτερα δουλεύει το μοντέλο όταν κάνει απλά predict πάνω μόνο σε πρόσωπα.
Μέχρι τώρα τα καταστήματα χρησημοποιουσαν έναν άνθρωπο για κάνει αυτήν την δουλειά, να δίνει δηλαδή δικαιώματα στο να μπει κάποιος η όχι με βάση άμα φοράει μάσκα. Η λύση που προτείνω είναι το μοντέλο που έχω χρησηοποιησει να μεταφερθεί σε tensorflow lite για να μπορεί να τρέχει και σε javascript πλατφόρμες και σε μικα circuits. Μπορεί αυτό δηλαδή πάνω σε ένα board με κάμερα να βγάζει μια φωτογραφία του ανθρώπου που θέλει εισέλθει στης εγκαταστάσεις και μέσα σε δευτερόλεπτα να αποφασίζει άμα φοράει μάσκα η όχι. Θα μπορούσε να γίνει όμως ακόμα και πιο άμεσα άμα όλο το μοντέλο μεταφοροταν σε computer vision, να μπορεί δηλαδή να βγάζει αποτελέσματα και από video

feed. Είναι ένας αρκετά αποτελεσματικός τρόπος καθώς θα αυτοματοποίηση μια κουραστική δουλειά και θα αφαιρή την ανθρώπινη προκατάληψη. Το κόστος της εφαρμογής είναι μόνο το κόστος για το cirbuit board και την κάμερα. Θα μπορούσε επιπλέον στο πρότζεκτ να προστεθεί Distance computation μεσώ scipy spatial distance, έτσι να ξέρει το πρόγραμμα άμα τηρούνται και τα μετρά της απόστασης.

Ωσο για τον Covid-19 η επίλυση που δίνει αυτό το πρότζεκτ μειώνει αρκετά την χρήση των ανρθωπον για μια απλή δουλειά, που σημαίνει λιγότερη επαφή. Επίσης άμα εμπλουτιστεί μαζί με Distance computation, θα μπορούσαμε να βοηθήσουμε την καθημερινότητα του κόσμου εφόσον σε εγκαταστάσεις θα τηρούνται όλα τα περιοστικα μετρά. Θα μπορούσε ακόμα να χρησημοποιηθει στα μέσα μαζικής μεταφοράς οπού δεν τηρούνται πάντα τα περιοριστικά μετρά σωστά.
Όπως είπα και παραπάνω η λύση η οποία έχουμε τώρα για αυτά τα προβλήματα είναι κάποιος άνθρωπος η απλά η εμπιστοσύνη των άλλων ανρθωπων στο να τηρήσουν τα μετρά. Η λύση που προτείνω θα αφαίρεση αυτές της ανθρώπινες οντότητες για να της αντικατάσταση με μια κάμερα η οποία με πολύ καλή ακρίβεια θα αναγκάσει τους ανρθωπους να τηρήσουν τα μετρά. Για παράδειγμα άμα σε ένα λεωφορείο είναι ήδη 20 άτομα και κρατάνε σωστά τα μετρά απόστασης και φοράνε όλοι μάσκα, ο οδηγός θα γίνει notified ότι δεν πρέπει να δεχθεί άλλους ανρθωπους μέχρι κάποιος να κατεβεί. Το καταλαβαίνω ότι η λύση που προτείνω δεν είναι η καλύτερη καθώς πολλοί ανρθωποι θα χάσουν λεωφορεία και χρόνο από αυτό, αλλά τουλάχιστον θα είναι ασφαλές.

Συμπερασματικά πιστεύω ότι λόγω του COVID-19 είναι καλύτερο να βάζουμε την ασφάλεια των ανρθωπων σαν κυριότητα, επίσης μεσώ της λύσης μου μειώνουμε την ανρθωπινη επαφή και ταυτόχρονα έχουμε άμεσα δεδομένα για την τήρηση μέτρων. Έτσι εφόσον τηρηθούν σωστά τα μετρά θα μπορούμε να δούμε πως επηρεάζουν τελικά το ποσοστό μόλυνσης ανθρώπων αυτα και πως θα μπορούμε να τα αλλάξουμε για καλύτερα αποτελέσματα