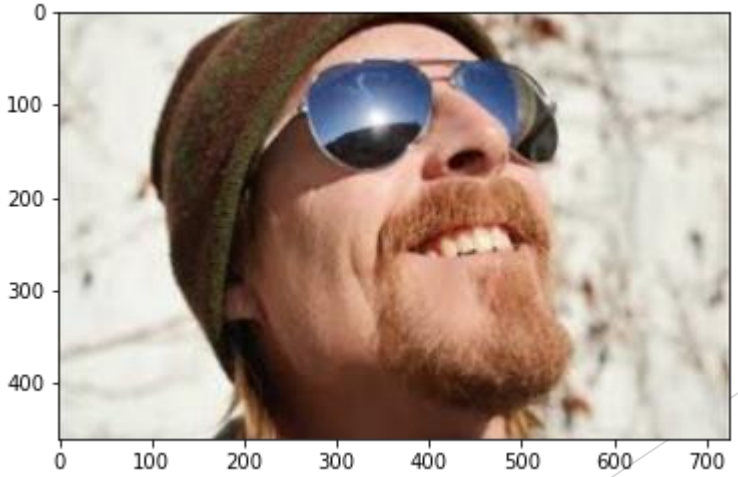
IIT Roorkee Final Project (Cloudxlab)

ABIJITH PRADEEP

Objective

- The problem statement of this project is to detect faces with masks and faces with no masks.
- We are supposed to find out the box co-ordinates surrounding the face.



Approach

- Using the face mask detection dataset in Kaggle I have collected, explored and pre-processed the data required for training the model.
- The two files are train.csv and submission.csv
- The train.csv file is used for training the model but the submission.csv contains the images with null values which I will use for classifying and detected the box around the face of the image.

Out[5]:

	name	x1	x2	y1	y2	classname
0	2756.png	69	126	294	392	face_with_mask
1	2756.png	505	10	723	283	face_with_mask
2	2756.png	75	252	264	390	mask_colorful
3	2756.png	521	136	711	277	mask_colorful
4	6098.jpg	360	85	728	653	face_no_mask

Out[6]:

	name	x1	x2	y1	y2	classname
0	1800.jpg	NaN	NaN	NaN	NaN	NaN
1	1800.jpg	NaN	NaN	NaN	NaN	NaN
2	1800.jpg	NaN	NaN	NaN	NaN	NaN
3	1799.jpg	NaN	NaN	NaN	NaN	NaN
4	1799.jpg	NaN	NaN	NaN	NaN	NaN

Approach

- I have used the mtcnn library which contains a detector for detecting and extracting the box of faces in an image.
- Then I used the training data and fit it into the keras sequential model. The model is Long Short term Memory (LSTM) model with batch size as 5 and 30 epochs.
- The mtcnn's detect_faces function returns a dictionary of dictionary. The internal dictionary named 'box' contains the co-ordinates for the rectangular plot around the face(s) of the image.
- The mtcnn is just for the accurate visualisation. The LTSM model that I trained will be used to predict the test images. As said before, the detect faces gets the box co-ordinates and we can signal the viewers that the face has been detected. The model is used for predicting what class the test images lie under.

Approach

Model Fitting

[28]:
```
from keras.layers import LSTM
model=Sequential()
model.add(Conv2D(100,(3,3),input_shape=x.shape[1:],activation='relu',strides=2))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(50, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(2, activation='softmax'))
```

[29]:
```
opt = tf.keras.optimizers.Adam(lr=1e-3, decay=1e-5)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(x,y,epochs=30,batch_size=5)
```

Model Summary

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 24, 24, 100)       1000
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 100)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 10, 64)        57664
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense_1 (Dense)              (None, 50)                80050
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 102
=================================================================
Total params: 138,816
Trainable params: 138,816
Non-trainable params: 0
_________________________________________________________________
None
```

Results

- The results are stored in a csv file called submission_1.csv.
- They have the predicted class name, predicted co-ordinate for rectangular plot.

[38]:
```
pd.read_csv('submission_1.csv')
```

Out[38]:

	Unnamed: 0	classname	name	x1	x2	y1	y2
0	3802	face_no_mask	2181.png	42	150	167	200
1	3801	face_with_mask	2179.png	249	188	251	314
2	3800	face_with_mask	2171.png	29	284	255	301
3	3799	face_with_mask	2163.png	453	124	45	68
4	3798	face_with_mask	2160.png	158	178	351	436
...	...	...	...	...	...	...	...
3798	3	face_no_mask	0006.jpg	441	668	57	70
3799	2	face_with_mask	0004.jpg	630	176	212	266
3800	1	face_with_mask	0003.jpg	1292	218	865	1088
3801	3716	face_no_mask	0002.png	349	102	212	268
3802	0	face_no_mask	0001.jpg	441	108	341	416

3803 rows × 7 columns

Inferences

- We have now detected the faces in each image by drawing a rectangular plot in the image around the face.
- If there are multiple faces in the image then it’ll draw the same number of box plots.

Improvising

- The project can be towards becoming a scanner that after detecting the face tells how correct the mask has been worn.
- By using another model to train images that have people wearing masks correctly.
- One can add a web cam feature to scan the face correctly while entering an organization. For instance, if the person isn't wearing the mask correctly (not covering nose or hanging mask below chin etc.) he won't be allowed to enter the organization.