Futuristic Work Place- AI Sanitised Bank!

Madhumathi Rajesh

Abstract:

Insomnia, backaches, stress and anxiety are amongst the consequence, people are facing from prolonged work from home. We spend most of our time either in front of the laptop screen or mobile phone screen sometimes stretching hours. Initially work from home was enjoyable. With the passage of time it has take the toll on the health. Attending meetings and interacting with people virtually is painful now a days. Yet the fear for fatality threatens the organizations to revive work place.

Our AI solution would bring some confidence on how we could possibly bring our physical presence in our favourite work place once again. Three mantras to protect our human resource from the pandemic are

1. Wear Face Mask
2. Social Distancing
3. Regular Hand sanitizing

The paper is designed to demonstrate the feasible solutions for the three mantras with an end to end computer vision/deep learning pipeline with 3 phases namely Mask detection along with temperature check, Image segmentation based social distancing and regular hand sanitisation reminder in the following sections.

Introduction:

Offices and other businesses are perfect environments for video surveillance. Although they don't face the same theft and security risks as convenience stores or gas stations, offices have unique security risks that security cameras can help prevent. Five important work of cameras are Crime deterrent, Monitors activities, collect evidence, Decision Making and Keeping records. When it comes to settling disputes, footage from security cameras can be incredibly important. Adding to the above said utility, cameras can be further exploited in ambitious ways.

Wearing a face mask will help prevent the spread of infection and prevent the individual from contracting any airborne infectious germs. When someone coughs, talks, sneezes they could release germs into the air that may infect others nearby. Face masks are part of an infection control strategy to eliminate cross-contamination. The paper describes an efficient utilization of the pictures captured from the existing cameras installed in the bank for mask monitoring. Images captured from the live cameras would be considered as input to our model that identifies if the person is wearing the mask or not along with the temperature measurement calculated by converting the picture to thermal image.

Social and physical distancing is deliberately increasing the physical space between people to avoid spreading illness. Staying at least six feet away from other people lessens the chances of catching COVID-19. Though unintentionally, people are habituated to walk and move around in common areas together, monitoring the distance between the person is also a need of the hour. Our model is trained on segmentation approach which locates the presences of human in the picture and calculates the physical distance between them and would potentially raise an alarm with the location coordinates.

Further reading of the paper would explain the technical background of the implementation of the proposed idea and the role of AI in resolving the COVID-19 pandemic.

Mask and Temperature detection Model:

A noncontact, camera captures the real time objects (employee face). Paper proposes a three-step temperature and mask detection namely,

Training Face: Face localization and masked face data preparation with region of interest for temperature detection and mask detection. ROI for Temperature detection would be forehead while for mask detection would be the regions surrounding the nose, cheeks and chin.

Deploying the Detector: Loading the mask detector in our server to perform face detection, and then classifying each face as with or without mask.

Temperature Detector: With the conversion of thermal images and the located ROI in the facial part raw value of body surface temperature is extracted and then passed through the calibrated formula to predict the allowable temperature.

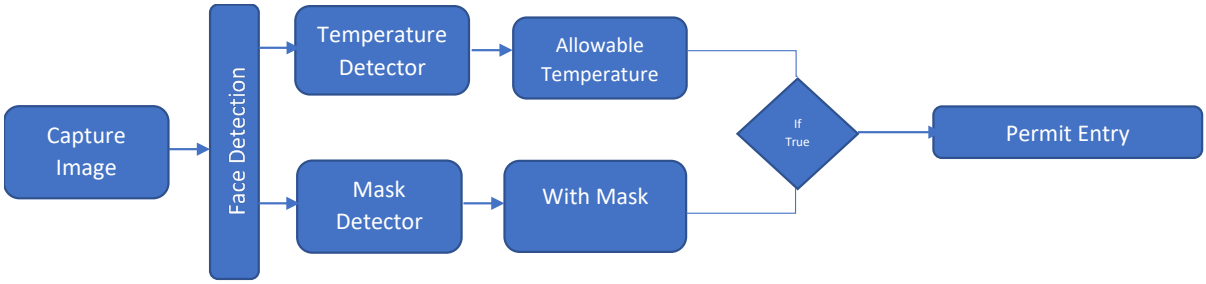

Figure 1: Module description

Dataset and Data Preparation: The dataset contains 7049 facial images. The model did not utilize the key point data available in the dataset as our target locations did not meet the requirements. The source of the dataset is available here.

Facial Key point location: Facial landmarks are used to localize and represent salient regions of the face, such as Forehead, Eyes, Eyebrows, Nose, Mouth and Jawline. Facial landmarks have been successfully applied to face alignment, head pose estimation, face swapping, blink detection and much more. Detecting facial landmarks is a subset of the shape prediction problem. Given an input image and normally an ROI that specifies the object of interest, a shape predictor attempts to localize key points of interest along the shape. In the context of facial landmarks, our goal is to detect important facial structures on the face using shape prediction methods. Model suggests dlib's facial landmark detector to locate the key points.

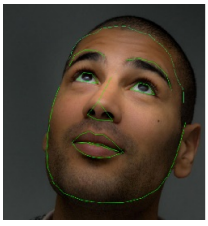

Figure 2: Facial Land marks

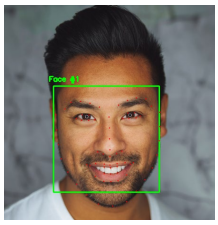

Figure 3: Facial Key-points

Figure 4 : Synthetic data with Mask

The pre-trained facial landmark detector inside the dlib library is used to estimate the location of 68 (x, y)-coordinates that map to facial structures. These annotations are part of the 68-point iBUG 300-

```
## Load the model
shape_predictor= r'..\shape_predictor_68_face_landmarks.dat'
detector =  dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(shape_predictor)

# load the input image, resize it, and convert it to grayscale
image = cv2.imread(image)
image = imutils.resize(image, width=500)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# detect faces in the grayscale image
rects = detector(gray, 1)

# loop over the face detections
for (i, rect) in enumerate(rects):
    shape = predictor(gray, rect)
    shape = face_utils.shape_to_np(shape)
```

W dataset which the dlib facial landmark predictor was trained on. The below code snippet was used to locate the required mask coordinates. A transparent mask is applied to the face by using the facial landmarks namely the points along the chin and nose to compute where the mask will be placed.

Face mask Detector:

Building a classifier model to identify the person with or without mask could be done with state-of-the-art topless object detection architecture like Yolo, SSD, RCNN etc. The paper focus on two major things namely face detection and image segmentation for social distancing. Hence a simple architecture along with segmentation of the object location is proposed that could be used for transfer learning.

There are various levels of granularity in which the computers can gain an understanding of images. The most fundamental building block in Computer Vision is the Image classification problem where given an image, model would output a discrete label namely Mask/No-Mask, which is the main object in the image. Combining localization along with the discrete label, we confine where exactly the object is present in the image. This localization is typically implemented using a bounding box which can be identified by some numerical parameters with respect to the image boundary considering only one object per image. We extend Object Detection to the next level where the image is not constrained to have only one object, but can contain multiple objects. With semantic image segmentation where each pixel of an image with a corresponding class of what is being represented would also be identified. Instance segmentation is one step ahead of semantic segmentation wherein along with pixel level classification, the model classifies each instance of a class separately. Figure 5 narrates the instance segmentation. The need for segmentation would be further discussed in the social distancing section.

Figure 5: Instance Segmentation

We propose customised UNet architecture which contains two paths. First path is the contraction path (encoder) which is used to capture the context in the image. The encoder is just a traditional stack of convolutional and max pooling layers. The second path is the symmetric expanding path (decoder) which is used to enable precise localization using transposed convolutions. It is an end-to-end fully convolutional network (FCN), which only contains convolutional layers and does not contain any dense layer and hence it can accept image of any size.

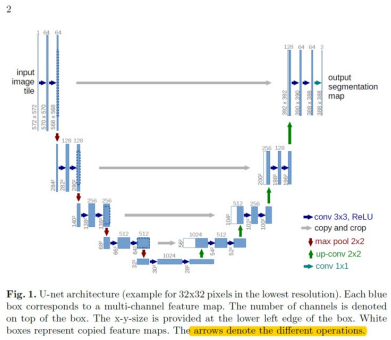

Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Figure: 6 Original UNet Architecture proposed by Olaf et. al

Since our input image is small (128x128) when compared with the size proposed in the paper(527x527), we build a custom architecture to serve the purpose.

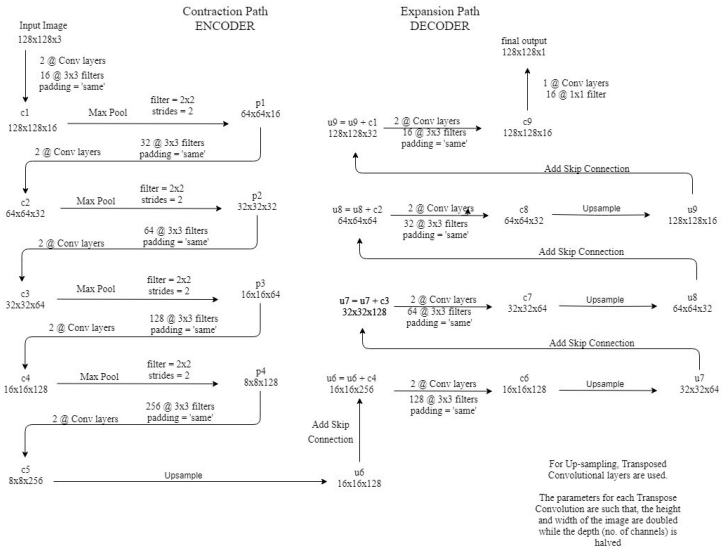

Figure 7: Proposed Unet Architecture

On a high level, we have the following relationship: Input (128x128x1) => Encoder =>(8x8x256) => Decoder =>Output (128x128x1). For the mask detection purpose, we consider the encoder part called as contracting path of the unet, followed by softmax classifier. On the other hand, the exact Unet architecture was utilized for instance segmentation thereby supporting code reusability. The keras implementation of network architecture is presented below.

```
def unet(input_img, n_filters = 16, dropout = 0.1, batchnorm = True):
    # Contracting Path
    in1 = Input(shape=(IMG_HEIGHT, IMG_WIDTH, 3 ))

    conv1 = Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(in1)
    conv1 = Dropout(0.2)(conv1)
    conv1 = Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(conv1)
    pool1 = MaxPooling2D((2, 2))(conv1)

    conv2 = Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(pool1)
    conv2 = Dropout(0.2)(conv2)
    conv2 = Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(conv2)
    pool2 = MaxPooling2D((2, 2))(conv2)

    conv3 = Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(pool2)
    conv3 = Dropout(0.2)(conv3)
    conv3 = Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(conv3)
    pool3 = MaxPooling2D((2, 2))(conv3)

    conv4 = Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(pool3)
    conv4 = Dropout(0.2)(conv4)
    conv4 = Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(conv4)

    # Expansive Path

    up1 = concatenate([UpSampling2D((2, 2))(conv4), conv3], axis=-1)
    conv5 = Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(up1)
    conv5 = Dropout(0.2)(conv5)
    conv5 = Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(conv5)

    up2 = concatenate([UpSampling2D((2, 2))(conv5), conv2], axis=-1)
    conv6 = Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(up2)
    conv6 = Dropout(0.2)(conv6)
    conv6 = Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(conv6)

    up2 = concatenate([UpSampling2D((2, 2))(conv6), conv1], axis=-1)
    conv7 = Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(up2)
    conv7 = Dropout(0.2)(conv7)
    conv7 = Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(conv7)
    segmentation = Conv2D(3, (1, 1), activation='sigmoid', name='seg')(conv7)

    outputs = Conv2D(1, (1, 1), activation='sigmoid')(c9)
    model = Model(inputs=[input_img], outputs=[outputs])
    return model

## Classification

Classify_Mask = AveragePooling2D(pool_size=(7, 7))( conv4)
Classify_Mask = Flatten(name="flatten")( Classify_Mask)
Classify_Mask = Dense(128, activation="relu")( Classify_Mask)
Classify_Mask = Dropout(0.5)( Classify_Mask)
Classify_Mask = Dense(2, activation="softmax")( Classify_Mask)
```

Social Distancing:

We train the end to end model of the Unet architecture with the person data set, available here. This dataset contains 13360 elaborately annotated human instances within 5081 images. With an average 0.573 MaxIoU of each person, OCHuman is the most complex and challenging dataset related to humans. With the live location of image segmentation (instance segmentation) we propose a distance

```
    model = Model(inputs=[in1], outputs=[segmentation])
    losses = {'seg': 'binary_crossentropy'  }
    metrics = {'seg': ['acc'] }
    model.compile(optimizer="adam", loss = losses, metrics=metrics)
```

metrics that calculate the distance between the person instance with the minimum distance

measuring 6 feet. Model would generate an alarm based on the object distance measures based on the location in the grid of the workspace.

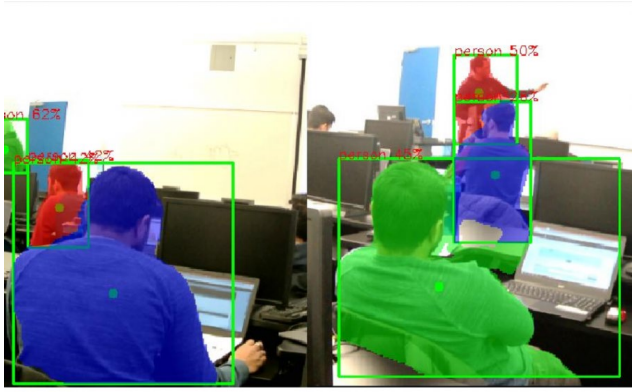

Figure 8: Instance segmentation for measuring distance

Temperature Model:

To detect the temperature of a person much advised is a thermal sensor camera. The paper proposes the reuse of the unet model described earlier training with the thermal dataset found here. Tufts Face Database is the most comprehensive, large-scale face dataset that contains 7 image modalities namely visible, near-infrared, thermal, computerised sketch, LYTRO, recorded video, and 3D images. The dataset contains over 10,000 images, where 74 females and 38 males from more than 15 countries with an age range between 4 to 70 years old are included.

Figure 9 sample Thermal data

Figure 10 Face detection

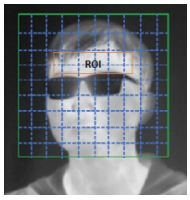

Figure 11: Locating the Region of Interest

Once the face is detected, kernel correlation filter (KCF) tracker is used to track the face region across each image. KCF is a highly accurate tracking algorithm. The general flow of the algorithm is as follows:

Kernel Correlation Filter Algorithm:

1. *Calculate the HOG features within the tracking target*
2. *Train a Correlation Filter (CF) with HOG features*
3. *Calculate the correlation between CF and the next frame*
4. *Get the most relevant place*

After the face detection and tracking, we can determine the position of the person's face in consecutive frames captured in the motion video. For the subsequent measurement of body temperature, we shall use the facial features to define the forehead area as our ROI. When radiometric mode is enabled in the camera, pixel values would be stabilized and normalized. The signal from the

camera is called flux-linear because it is linear to the radiometric flux within Lepton's spectral band. The flux-linear signal is related to scene temperature by the Planck curve determined by the following formula.

$$S = \int_{\lambda_1}^{\lambda_2} \frac{2\pi hc^2}{\lambda^5} \frac{1}{\exp(\frac{hc}{\lambda k T_k}) - 1} R(\lambda) \cdot \delta\lambda$$

where S is the output signal, λ_1 and λ_2 are define the spectral band, h is Planck's constant c is the velocity of light, K is Boltzmann's constant, R(λ)is the camera responsivity, T_k is absolute temperature in units of Kelvin. Above formula would be customised and implemented in python as,

$$S = \frac{R}{\exp\left(\frac{B}{T_k}\right) - F} + O \qquad T_k = \frac{B}{\ln\left(\frac{R}{S - O} + F\right)}$$

where S denotes the output signal from the camera R, B, F and O are parameters generated during calibration, T_k is the target's absolute temperature in units of Kelvin. In order to get accurate human body temperature, we shall perform radiometry calibration on this sensor. Ideally, the heat source should be located at the distance similar to targets of interest in the applications (40-80cm away). Average values in a ROI of the image are in the range as follows: R = [10000,10000], B= [1200,1700], F = [0.5,3], O = [-16384,16383]. Figure 12 shows the target setup for capturing the parameters.

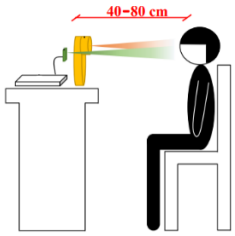

Figure 12 : Target setup

Hand Sanitation Alarm:

The third mantra to protect our employees from the pandemic is proper hand sanitization. This is an addon module that could probably be implemented with an alarm in locations were the entry and exit are controlled. The alarm would request for the subject to self-sanitize while entering the work place. Posture based prediction model would be deployed to predict if the subject is using sanitizer or not while entering the work area. To collect the data our paper proposes a motion sensing device known as Microsoft Kinect using which the skeletal points of the human body could be detected and observed for movements.

We suggest Graph convolutional networks (GCNs), as it is more common and general in the fields, such as social network, molecule and parse tree. How to operate on graphs has been explored extensively over decade and now the most popular solution is to use the graph convolutional networks which generalize CNNs to more generic non-Euclidean structures, that has achieved remarkable performance for skeleton-based action. The GCNs is similar with the traditional CNNs, but it can generalize the convolution from image to graph of arbitrary size and shape. The topology of the graph is set heuristically and fixed over all the model layers and input data. Information of the skeleton data, namely the length and orientation of the bones, which carries more informative and discriminative

datapoints for the human action recognition would be considered. The graph topology in our model can be either uniformly or individually learned based on the input data in an end-to-end manner.

The input to traditional CNNs is usually low-dimensional regular grids, such as image, video and audio. However, it is not straightforward to use the CNN to model the graph data, which always has arbitrary size and shape. The principle of constructing GCNs mainly follows two streams: spatial perspective and spectral perspective. Spatial perspective methods directly perform convolution on the graph vertexes and their neighbours. The key lies in how to construct the locally connected neighbourhoods from the graph with the implicit order of vertexes and edges. A normalization algorithm is proposed to crop excess vertexes and pad dummy vertexes. In contrast to the spatial perspective methods, spectral perspective methods use the eigenvalues and eigen vectors of the graph Laplace matrices. These methods perform graph convolution in the frequency domain with the help of the graph Fourier transform.

Graph Construction: The raw skeleton data in one frame are represented by a sequence of vectors. Each vector represents the 2D or 3D coordinates of the corresponding human joint. A complete action contains multiple frames with different lengths for different samples. We shall use a spatiotemporal graph to model the structured information among these joints along both the spatial and temporal dimensions. Here, the spatial dimension refers to the joints in the same frame, and the temporal dimension refers to the same joints over all of the frames. Figure 14 narrates the spatiotempoal graph. The coordinate vector of each joint is set as the attribute of the corresponding vertex. Since the graph is intrinsic and is built based on the natural connectivity of the human body, we shall refer it as the human-body-based graph.

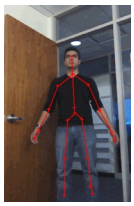

Figure 13: Kinect to loacate skelital coordinates

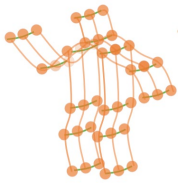

Figure 14: Illustration of the spatiotemporal graph

The graph convolution operation on vertex v_i is formulated as:

$$f_{out}(v_i) = \sum_{v_j \in \mathcal{B}_i} \frac{1}{Z_{ij}} f_{in}(v_j) \cdot w(l_i(v_j))$$

where f denotes the feature map and v denotes the vertex of the graph. Bi denotes the sampling area of the convolution for v_i, which is defined as the 1-distance neighbouring vertexes (v_j) of the target vertex (v_i). w is the weighting function similar to the traditional convolution operation, which provides a weight vector based on the given input. The number of weight vectors of convolution is fixed, while the number of vertexes in B_i is varied. A mapping function l_i is required to map all neighbouring vertexes into a fix-numbered subset each of which is associated with a unique weight vector. The implementation of the graph convolution for the spatial dimension is not straightforward. Concretely, the feature map of the network is actually a tensor $f \in R^{C \times T \times N}$, where N denotes the number of vertexes, T denotes the temporal length and C denotes the number of channels.

Dataset used for training the model is available here. Kinetics-Skeleton: Kinetics is a large-scale human action dataset that contains 300,000 videos clips in 400 classes. The video clips are sourced from YouTube videos and have a great variety. It only provides raw video clips without skeleton data, estimate the locations of 18 joints on every frame of the clips using the publicly available OpenPose

toolbox would be utilized. Simple keras based sequential implementation of graph convolution is demonstrated below for visualizing graph CNN.

```
from keras_dgl.layers import GraphCNN
graph_conv_filters = K.constant(graph_conv_filters)
model = Sequential()
model.add(GraphCNN(Y.shape[1], NUM_FILTERS, graph_conv_filters, input_shape=(X.shape[1],), activation='relu', kernel_regularizer=l2(5e-4)))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.01), metrics=['acc'])
model.summary()
```

The idea of posture recognition to find if the person has performed sanitization is still experimental and hence the architecture to determine the same shall not be further elaborated.

Conclusion:

As the world grapples with COVID-19, every ounce of technological innovation and ingenuity harnessed to fight this pandemic brings us one step closer to overcoming it. Artificial intelligence (AI) and machine learning are playing a key role in better understanding and addressing the COVID-19 crisis. Every kind of organization, whether small or large, public or private, is finding new ways to operate effectively and to meet the needs of their customers and employees as social distancing and quarantine measures remain in place. At this juncture exploring and exploiting the AI algorithms with the available resources to bring a productive and safe work place is the highlight of the paper. We have discussed how the three mantras could be implemented with the support of computer vision models namely face mask detection, temperature monitoring, social distancing and hand sanitization check. The paper implemented Unet based architecture for mask, social distancing and ROI localization for temperature measure module while GCNN, graph Convolution nets for posture identification to visualize if the employee uses hand sanitizer while entering the work place.

References:

[1] https://arxiv.org/abs/1505.04597

[2] https://arxiv.org/pdf/1901.00596.pdf

[3] http://dlib.net/

[4] DoI: 10.1109/GCCE.2018.8574800

[5] https://link.springer.com/article/10.1186/s40537-019-0197-0

[6] https://github.com/open-mmlab/mmskeleton

[7] http://tdface.ece.tufts.edu/

[8] https://cg.cs.tsinghua.edu.cn/dataset/form.html?dataset=ochuman