Face Mask Detection using Deep Learning and Computer Vision

Swetha Mohan[1], Ankit Kumar[2] , Abinash Kushwaha[3]
School of Computer Science Engineering,
Vellore Institute of Technology, Vellore

Abstract - **Wearing a mask is among the non-pharmaceutical measures that can be used to cut the primary source of COVID droplets expelled by an infected individual. To contribute towards communal health, this project aims to devise a highly accurate and real-time technique that can efficiently detect non-mask faces in public and thus, enforce them to wear masks. Although numerous researchers have committed efforts in designing efficient algorithms for face detection and recognition, there exists an essential difference between 'detection of the face under mask' and 'detection of mask over face'.**

As per available literature, very little body of research has attempted to detect masks over face. Thus, this work aims to develop techniques that can accurately detect masks over the face in public. Further, it is not easy to detect faces with/without a mask in public as the dataset available for detecting masks on human faces is relatively small leading to the hard training of the model. So, the concept of transfer learning is used here to transfer the learned kernels from networks trained for a similar face detection task on an extensive dataset. The dataset covers various face images including faces with masks, faces without masks, faces with and without masks in one image and confusing images without masks.

Keywords - Transfer learning, kernels, COVID droplets, training of models, mask over face.

I. INTRODUCTION

With the reopening of countries from COVID-19 lockdown, Government and Public health agencies are recommending face masks as essential measures to keep us safe when venturing into public to curtail the spread of Coronavirus and thereby contributing to public healthcare. Regardless of discourse on medical resources and diversities in masks, all countries are mandating coverings over the nose and mouth in public.

To mandate the use of facemasks, it becomes essential to devise some techniques that enforce individuals to apply a mask before exposure to public places. This application can be very useful in public areas such as airports, railway stations, crowded markets, malls, etc. The proposed method used here is carried out in two steps. The first step is to train the face mask detector using transfer learning. The second step is to use this trained face mask detector on images or videos of people to identify if they are wearing a mask.

II. LITERATURE SURVEY

MobileNet Mask Model

All governments around the world are struggling against COVID-19, which causes serious health crises. Therefore, the use of face masks regulatory can slow down the high spread of this virus. Dey proposed a deep learning-based model for detecting face mask. This model named "MobileNet Mask" is multiphase. A pretrained model of the ResNet-10 architecture is utilized to find faces in video stream. Also, numerous steps are used such as charging the classifier (MobileNet), building the FC layer, and testing phase. All the experimental cases are supervised on Google Colab that runs in the cloud and is provided with over 12 GB of RAM. Different performance metrics (accuracy, F1-score, precision, and recall) are used to judge the performance of the proposed model.

ResNet-50 with YOLO-V2 Model

Annotating and localizing medical face masks in real-life images is among the most important object detection applications. In this context, the main objective of Loey is to explain and delimit the objectives of the medical face masks, especially in real images. They proposed a model consisting of two steps: medical face masks and feature extraction.

The two public datasets of medical face masks are merged in one dataset to be explored in their research. The first one is Medical Masks Dataset (MMD). It contains 682 images with more than 3000 faces wearing masks. The second one is Face Mask Dataset (FMD), which contains 853 images. Combining both datasets resulted in a dataset of just 1415 pictures after deleting bad quality pictures.

Deep Learning Tools and CXR Image-Based COVID-19 Detection

Radiography is a technique used to quantify the functional and structural consequences of chest diseases, to provide high-resolution images on disease progression. Several works have been carried out in this context. Echtioui proposed a new CNN-based method for COVID-19 recognition, through analysing radiographic images of a patient's lungs. The aim of this scheme is to provide clinical decision support for healthcare workers and also for researchers. Hence, performance results, as well as the accuracy value of about 91.34%, and the other metrics in terms of recall, precision, and F1-score, prove the efficiency of the method. In the same context, Ozturk introduced a new automatic COVID-19 detection model using CXR images denoted by the “DarkCovidNet.” It is used to provide correct diagnosis for both a binary classification (COVID-19 VS no findings) and a multiclass classification (COVID-19 VS pneumonia VS no findings).

Deep Learning Tools and CT Image-Based COVID-19 Detection

Computed tomography scan or CT scan is a medical imaging technique utilized in radiology in order to get detailed images of the body for diagnosis purposes. Accurate and fast COVID-19 screening is achievable using CT scan images. Various works have been carried out in this context. Shah proposed distinct deep learning techniques to differentiate CT scan images of both COVID-19 and non-COVID-19, which helps in diagnosis. In the dataset, we find 349 images corresponding to patients with COVID-19 and 463 images corresponding to patients without COVID-19. These images were divided into three sets: 80% of them for training set, 10% for validation, and 10% for testing.

Methods Using CXR and CT Images

Combining two types of images in one dataset is an effective method to detect a disease. Sedik presented two deep learning models: CNN and ConvLSTM. To simulate the models, two datasets are assumed. The first dataset includes CT images while the second set includes X-ray images. Each dataset contains COVID-19 and non-COVID-19 image categories. The image categories, COVID-19 and pneumonia, were classified to certify the proposed models.

The first model based on CNN includes five convolutional layers (CNVLs) accompanied by five pooling layers (PLs). Two layers (fully connected layer (FC) and classification layer) make up the classification network. The second model is a hybrid one. It combines ConvLSTM and CNN at the same time.

Related Work

Machine learning: Machine learning is a method of teaching prediction based on some data. It is a branch of artificial intelligence. Which numerically improves on data over as more data as add in algorithm the performance of the system is improved. These are the three types of machine learning:

Supervised learning: In supervised learning we have several data points or samples described using predictive variables or features and the target variable our data is represented in table structure. Game supervised learning is build a model its able to predict the target variable

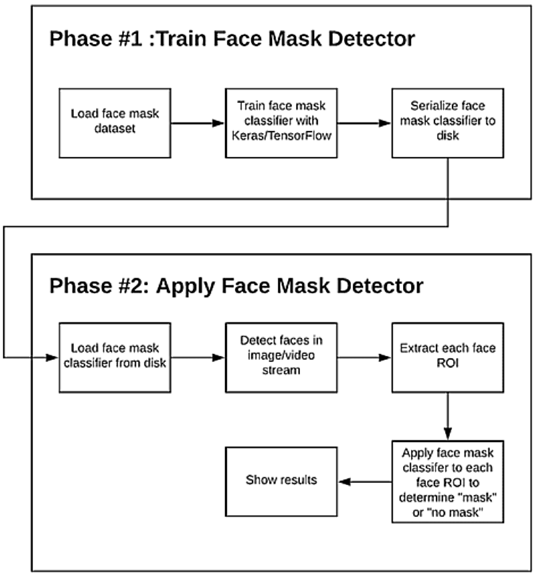
Unsupervised learning is a machine learning task of uncovering hidden patterns from unlabeled data.

Reinforcement learning (RL) in which machine or software agents interact with an environment reinforcement learning agent can automatically figure out how to optimize their behavior given a system of reward and punishments reinforcement learning draws inspiration from behavioral psychology.

Computer Vision: It is a field that includes processing, analyzing and understanding images in general high dimensional data from the real world in order to produce numerical and symbolic information or it is a technology of science and machines that see it obtain information from images.

Deep Learning: Deep learning is a powerful set of techniques for learning using neural networks. Neural networks are a beautiful biologically inspired programming paradigm which enables a computer to learn from data. These are learning algorithms.

III. METHODOLOGY


Phase #1 :Train Face Mask Detector
Load face mask dataset
Train face mask classifier with Keras/TensorFlow
Serialize face mask classifier to disk
Phase #2: Apply Face Mask Detector
Load face mask classifier from disk
Detect faces in image/video stream
Extract each face ROI
Apply face mask classifer to each face ROI to determine "mask" or "no mask"
Show results

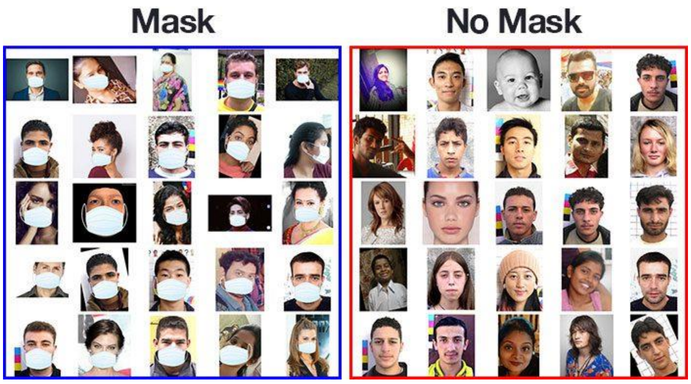
Phases and individual steps for building a COVID-19 face mask detector with computer vision and deep learning using Python, OpenCV, and TensorFlow/Keras.

In order to train a custom face mask detector, we need to break our project into two distinct phases, each with its own respective sub-steps:

Training*:* Here we’ll focus on loading our face mask detection dataset from disk, training a model (using Keras/TensorFlow) on this dataset, and then serializing the face mask detector to disk.

Deployment*:* Once the face mask detector is trained, we can then move on to loading the mask detector, performing face detection, and then classifying each face as with_mask or without_mask

COVID-19 face mask detection dataset



This dataset consists of 1,376 images belonging to two classes: with_mask: 690 images, without_mask: 686 image

Objective
Our goal is to train a custom deep learning model to detect whether a person is or is not wearing a mask.

Modules
Python3, OpenCV, Keras, TensorFlow.

IV. PSEUDOCODE

Training:
1. Generate your own annotation file and class names file:
Row Format: image_file_path box 1, box2, ... boxN.
Box format: x_min, y_min, x_max, y_max, class_id

2. Convert the pre-trained weights to '.h5' format as required by Keras.

3.Freeze all layers except the final layers and train for some epochs until Plateau (no improvement stage) is reached.

4. Unfreeze all the layers and train all the weights while continuously reducing the learning rate until again plateau is reached.

5. End

Testing using OpenCV:
1. Capture the video through webcam or any saved video testing file.

2. Pass each frame of video or captured frame from webcam through the model.

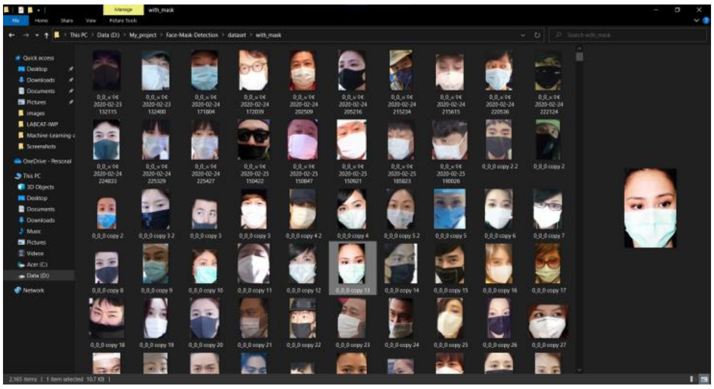
3. Get the boxes, scores and classes obtained as output and draw the boxes on the frame accordingly (colour depending on the class of each box).

4. Display a total number of boxes of both classes at the bottom of the screen and record them in a variable.

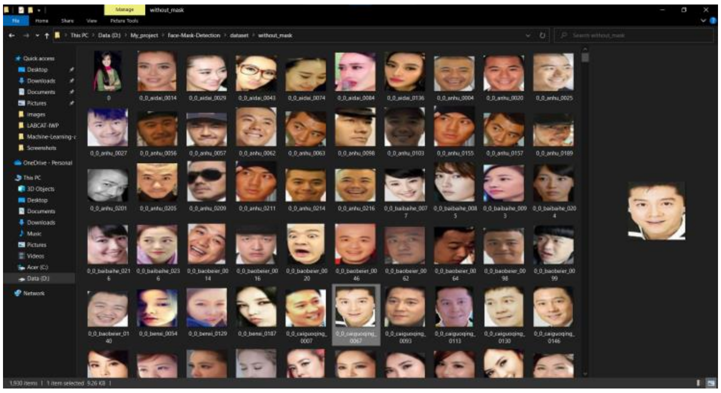
5. After iterating through each frame, we shall be able to get the output video with the results we wanted.

V. EXPERIMENTAL RESULTS

Dataset Used :

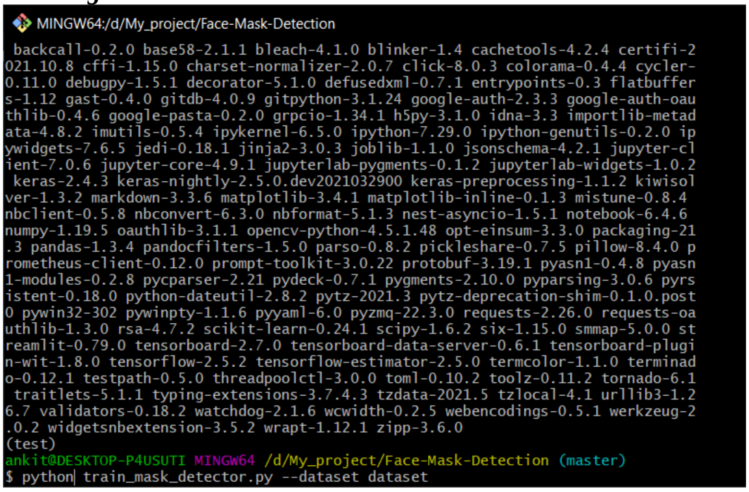


With mask

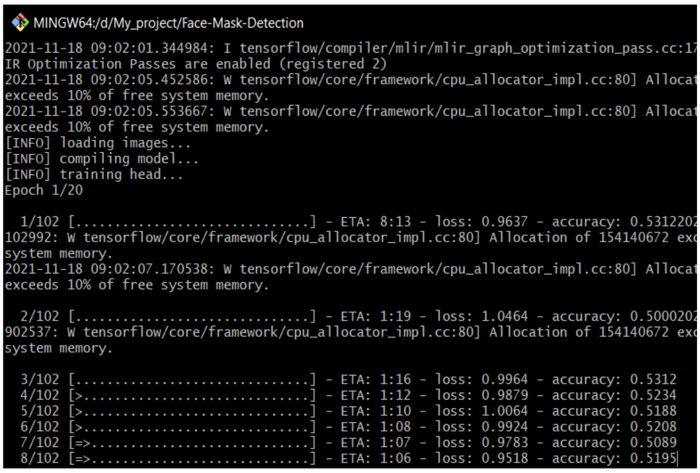


Without mask

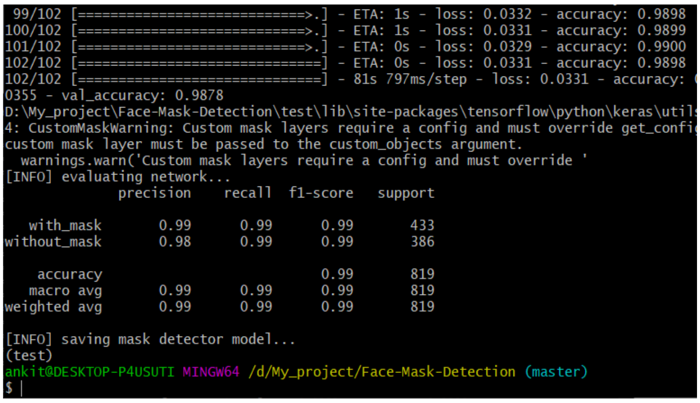
Training the model:



```
MINGW64:/d/My_project/Face-Mask-Detection
 backcall-0.2.0 base58-2.1.1 bleach-4.1.0 blinker-1.4 cachetools-4.2.4 certifi-2
021.10.8 cffi-1.15.0 charset-normalizer-2.0.7 click-8.0.3 colorama-0.4.4 cycler-
0.11.0 debugpy-1.5.1 decorator-5.1.0 defusedxml-0.7.1 entrypoints-0.3 flatbuffer
s-1.12 gast-0.4.0 gitdb-4.0.9 gitpython-3.1.24 google-auth-2.3.3 google-auth-oau
thlib-0.4.6 google-pasta-0.2.0 grpcio-1.34.1 h5py-3.1.0 idna-3.3 importlib-metad
ata-4.8.2 imutils-0.5.4 ipykernel-6.5.0 ipython-7.29.0 ipython-genutils-0.2.0 ip
ywidgets-7.6.5 jedi-0.18.1 jinja2-3.0.3 joblib-1.1.0 jsonschema-4.2.1 jupyter-cl
ient-7.0.6 jupyter-core-4.9.1 jupyterlab-pygments-0.1.2 jupyterlab-widgets-1.0.2
 keras-2.4.3 keras-nightly-2.5.0.dev2021032900 keras-preprocessing-1.1.2 kiwisol
ver-1.3.2 markdown-3.3.6 matplotlib-3.4.1 matplotlib-inline-0.1.3 mistune-0.8.4
nbclient-0.5.8 nbconvert-6.3.0 nbformat-5.1.3 nest-asyncio-1.5.1 notebook-6.4.6
numpy-1.19.5 oauthlib-3.1.1 opencv-python-4.5.1.48 opt-einsum-3.3.0 packaging-21
.3 pandas-1.3.4 pandocfilters-1.5.0 parso-0.8.2 pickleshare-0.7.5 pillow-8.4.0 p
rometheus-client-0.12.0 prompt-toolkit-3.0.22 protobuf-3.19.1 pyasn1-0.4.8 pyasn
1-modules-0.2.8 pycparser-2.21 pydeck-0.7.1 pygments-2.10.0 pyparsing-3.0.6 pyrs
istent-0.18.0 python-dateutil-2.8.2 pytz-2021.3 pytz-deprecation-shim-0.1.0.post
0 pywin32-302 pywinpty-1.1.6 pyyaml-6.0 pyzmq-22.3.0 requests-2.26.0 requests-oa
uthlib-1.3.0 rsa-4.7.2 scikit-learn-0.24.1 scipy-1.6.2 six-1.15.0 smmap-5.0.0 st
reamlit-0.79.0 tensorboard-2.7.0 tensorboard-data-server-0.6.1 tensorboard-plugi
n-wit-1.8.0 tensorflow-2.5.2 tensorflow-estimator-2.5.0 termcolor-1.1.0 terminad
o-0.12.1 testpath-0.5.0 threadpoolctl-3.0.0 toml-0.10.2 toolz-0.11.2 tornado-6.1
 traitlets-5.1.1 typing-extensions-3.7.4.3 tzdata-2021.5 tzlocal-4.1 urllib3-1.2
6.7 validators-0.18.2 watchdog-2.1.6 wcwidth-0.2.5 webencodings-0.5.1 werkzeug-2
.0.2 widgetsnbextension-3.5.2 wrapt-1.12.1 zipp-3.6.0
(test)
ankit@DESKTOP-P4USUTI MINGW64 /d/My_project/Face-Mask-Detection (master)
$ python train_mask_detector.py --dataset dataset
```



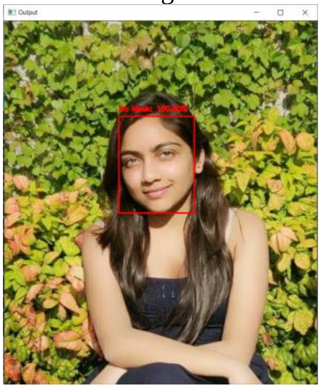
```
MINGW64:/d/My_project/Face-Mask-Detection
2021-11-18 09:02:01.344984: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:17
IR Optimization Passes are enabled (registered 2)
2021-11-18 09:02:05.452586: W tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocat
exceeds 10% of free system memory.
2021-11-18 09:02:05.553667: W tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocat
exceeds 10% of free system memory.
[INFO] loading images...
[INFO] compiling model...
[INFO] training head...
Epoch 1/20

  1/102 [..............................] - ETA: 8:13 - loss: 0.9637 - accuracy: 0.5312202
102992: W tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of 154140672 exc
system memory.
2021-11-18 09:02:07.170538: W tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocat
exceeds 10% of free system memory.

  2/102 [..............................] - ETA: 1:19 - loss: 1.0464 - accuracy: 0.5000202
902537: W tensorflow/core/framework/cpu_allocator_impl.cc:80] Allocation of 154140672 exc
system memory.

  3/102 [..............................] - ETA: 1:16 - loss: 0.9964 - accuracy: 0.5312
  4/102 [>.............................] - ETA: 1:12 - loss: 0.9879 - accuracy: 0.5234
  5/102 [>.............................] - ETA: 1:10 - loss: 1.0064 - accuracy: 0.5188
  6/102 [>.............................] - ETA: 1:08 - loss: 0.9924 - accuracy: 0.5208
  7/102 [=>............................] - ETA: 1:07 - loss: 0.9783 - accuracy: 0.5089
  8/102 [=>............................] - ETA: 1:06 - loss: 0.9518 - accuracy: 0.5195
```



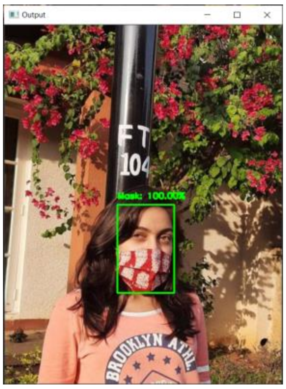
```
 99/102 [============================>.] - ETA: 1s - loss: 0.0332 - accuracy: 0.9898
100/102 [============================>.] - ETA: 1s - loss: 0.0331 - accuracy: 0.9899
101/102 [============================>.] - ETA: 0s - loss: 0.0329 - accuracy: 0.9900
102/102 [==============================] - ETA: 0s - loss: 0.0331 - accuracy: 0.9898
102/102 [==============================] - 81s 797ms/step - loss: 0.0331 - accuracy: 0
0355 - val_accuracy: 0.9878
D:\My_project\Face-Mask-Detection\test\lib\site-packages\tensorflow\python\keras\utils
4: CustomMaskWarning: Custom mask layers require a config and must override get_confi
custom mask layer must be passed to the custom_objects argument.
  warnings.warn('Custom mask layers require a config and must override '
[INFO] evaluating network...
              precision    recall  f1-score   support

   with_mask       0.99      0.99      0.99       433
without_mask       0.98      0.99      0.99       386

    accuracy                           0.99       819
   macro avg       0.99      0.99      0.99       819
weighted avg       0.99      0.99      0.99       819

[INFO] saving mask detector model...
(test)
ankit@DESKTOP-P4USUTI MINGW64 /d/My_project/Face-Mask-Detection (master)
$ |
```

Our model gave 98% accuracy for Face Mask Detection after training via tensorflow-gpu==2.5.0.

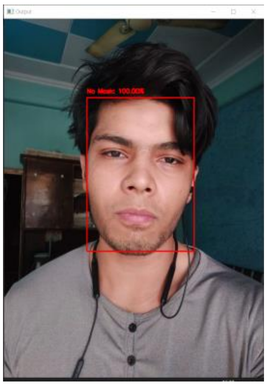
We used our own images to verify the working of the custom deep learning model to detect whether a person *is* or *is not* wearing a face mask.

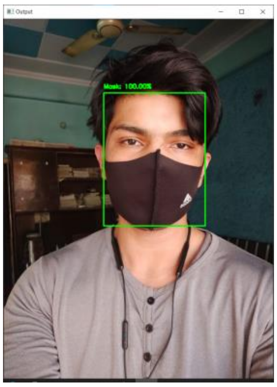


without mask



with mask





VI. CONCLUSION

Due to the urgency of controlling COVID-19, the application value and importance of real-time mask and social distancing detection are increasing. Our face mask detector doesn't use any morphed masked images dataset and the model is accurate. It is computationally efficient, thus making it easier to deploy the model to embedded systems (Raspberry Pi, Google Coral, etc.).

This system can therefore be used in real-time applications which require face-mask detection for safety purposes due to the outbreak of Covid-19. This project can be integrated with embedded systems for application in airports, railway stations, offices, schools, and public places to ensure that public safety guidelines are followed.

VII. REFERENCES

[1] Alexey Bochkovskiy, Chien-Yao Wang and Hong-Yuan Mark Liao, "Yolov4: Optimal speed and accuracy of object detection", 2020.

[2] Narinder Singh Punn, Sanjay Kumar Sonbhadra and Sonali Agarwal, "Monitoring COVID-19 social distancing with person detection and tracking via fine-tuned YOLO v3 and Deepsort techniques", 2020.

[3] Mahdi Rezaei and Mohsen Azarmi, "Deepsocial: Social distancing monitoring and infection risk assessment in covid-19 pandemic", *Applied Sciences*, vol. 10, no. 21, pp. 7514, 2020.

[4] Farooque Hassan Kumbhar, Syed Ali Hassan and Soo Young Shin, "New Normal: Cooperative Paradigm for Covid-19 Timely Detection and Containment using Internet of Things and Deep Learning", 2020.

[5] Enoch Arulprakash and Martin Aruldoss, "A study on fight against COVID-19 from latest technological intervention", *SN Computer Science*, vol. 1, no. 5, pp. 1-3, 2020.

[6] Dongfang Yang et al., "A vision-based social distancing and critical density detection system for covid-19", pp. 24-25, 2020.

[7] Zhanchao Huang, Jianlin Wang, Xuesong Fu, Tao Yu, Yongqi Guo and Rutong Wang, "DC-SPP-YOLO: Dense connection and spatial pyramid pooling based YOLO for object detection", *Information Sciences*, vol. 522, pp. 241-258, 2020, ISSN 002-0255.

[8] Jiahui Yu et al., "Unitbox: An advanced object detection network", *Proceedings of the 24th ACM international conference on Multimedia*, 2016.

[9] Z. Zhao, P. Zheng, S. Xu and X. Wu, "Object Detection With Deep Learning: A Review", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212-3232, Nov. 2019.

[10] Mohammad Javad Shafiee et al., "Fast YOLO: A fast you only look once system for real-time embedded object detection in video", 2017.