Journal of Management and Service Science, 2021,
Vol. 01, Iss. 02, S. No. 003, pp. 1-21
ISSN (Online): 2583-1798

Face Mask Detection using Deep Learning to Manage Pandemic Guidelines

Ankita Singh[1], Pawan Singh[2]

Amity School of Engineering and Technology, Amity University, Lucknow Campus, Uttar Pradesh, India[1,2]
ankitasingh8019@gmail.com[1], pawansingh51279@gmail.com[2]

How to cite this paper: A. Singh and P. Singh (2021) Face Mask Detection using Deep Learning to Manage Pandemic Guidelines. *Journal of Management and Service Science, 1(2), 3, pp. 1-20.*

https://doi.org/10.54060/JMSS/001.02.003

Received: 16/05/2021
Accepted: 27/05/2021
Published: 29/05/2021

Abstract

The field of Computer Vision is a branch of science of the computers and systems of software in which one can visualize and as well as comprehend the images and scenes given in the input. This field is consisting of numerous aspects for example image recognition, the detection of objects, generation of images, image super resolution and more others. Object detection is broadly utilized for the detection of faces, the detection of vehicles, counting of pedestrians on a certain street, images displayed on the web, security systems and cars with the feature of self-driving. This process also encompasses the precision of every technique for recognizing the objects. The detection of objects is a crucial task; however, it is also a very challenging vision task. It is an analytical subdivide of various applications such as searching of images, image auto-annotation or scene understanding and tracking of various objects. The tracking of objects in motion of video image sequence was one of the most important subjects in computer vision.

Keywords

Object Detection, Machine Learning, Deep Learning, Deep neural Networks, Convolutional Neural Network, Single Shot Detector, You Only Look Once, Open CV, Tensor flow

1. Introduction

1.1 Background

1.1.1 Computer Vision

The field of computer vision is a study focused on the issue of helping computers to see. It is a multidisciplinary field that could broadly be called a subfield of artificial intelligence and machine learning, which may involve the use of specialized methods and make use of general learning algorithms. Object recognition is a task which deals with describing a number of co-related tasks of computer vision which includes activities like identifying the objects in a digital photograph. Classification of images includes various tasks such as predicting the class of an object in a given input image. [1] Localization of objects refers to identifying the location of one or more objects in an image and drawing a rectangular bounding box around the extent of the object. This detection of objects simply is the work which combines these two processes and localize and classifies one or a number of objects in an image.

1.1.2 Artificial Intelligence

Artificial intelligence (AI) is the way of a digital computer or computer-controlled by a robot to undergo some task that are usually associated with intelligent being. This term is usually applied to the task of creating systems endowed with the intellectual processes characteristic of us humans, for example, as our capability to reason, bring to light meanings or to generalize or master from our past experience. Considering the evolution of the digital computer in the 1940s, it has been illustrated that computers can be programmed to take out very difficult task such as, for example, discovering proofs for mathematical theorems or playing chess—with huge competence. Despite of the continuation in advances in computer processing pace and memory, there are as of yet no programs that can counterpart human flexibility over a wider domain or in jobs requiring much everyday comprehension. Whereas, there are some programs that have accomplished the performance level of humans as experts and professionals in executing certain tasks, so that artificial intelligence in its restricted sense can be sought in application as varied as medical diagnosis, computer search engines and many more.

1.1.3 Machine learning

Machine learning is an operation that requires teaching any computer system how to forge accurate predictions when any type of data has been fed into the computer system. These predictions could be the answer to whether a piece of vegetable in a photo is a broccoli or a beetroot, analyzing twitter comments as negative, positive or neutral, predicting stock prices of the market, whether an email is a spam or not, or recognizing a speech accurately so as to generate some captions for a video. [1] The main difference between a traditional computer software and a machine learning model is that a human has not written any code that tells the computer system how to tell the difference between the broccoli and the Beetroot. Instead, the machine-learning model has been taught how to accurately differentiate between the Vegetables by training the model on a large amount of data, for example, a large number of photos with vegetables.

1.1.3.1 Types of Machine learning

a) Supervised Learning

In machine learning, this type of learning is the type where we can view that the learning is assisted by a someone, for example, a professor. We have the dataset which act as the professor and the role of a dataset i.e., the professor is to train the models or the machine. After the model got trained, the model can then process making a prediction, decision when the newly data is fed to it.

b) Unsupervised Learning

A model learns throughs the observations and find structure in those data. When the model is given the dataset, it can automatically find certain pattern and relationship in that dataset by constructing some clusters in it. What the model cannot perform is that it can’t add label to the cluster, like it could not say that this thing is a group of vegetables or fruits, however it can discriminate all the vegetables from the fruits. For instance, we present some images of cars, trucks and planes to a model, so what it will do, depending on some patterns and relationship, will create some cluster and divide that particular data set into those clusters. Later on, if a fresh data is put into the model as input, it puts it to one of the already built clusters.

c) Reinforcement learning

It is a potential of an agent, to inter-relate with its environment and figure out what would be the perfect possible outcome. The agent follows the notion of the hit and trial method. [2] The agent is then guerdoned with one point for one correct or penalized by one point for one wrong answer, and on the basis of the positive reward points gained by the agent, the model will train itself. Then once again, when that model is trained, the model would get prepared to make predictions regarding the fresh data given to it.

1.1.4 Deep learning

Deep learning is a sub-division of machine learning. It is a branch that is found on learning and ameliorating on its very own by inspecting certain computer algorithms. While machine learning utilizes easier concepts, here deep learning performs with various neural networks, that are constructed to emulate how a human learns and thinks. Until now, these neural networks were just restricted by the computing power and consequently were confined in complexity. Advancements in the big data analytics have allowed greater, sophisticated neural networks, allowing the computers to think, learn, and react to byzantine situations much faster than us human. Deep learning has assisted in image classification, translation of language etc.

1.1.5 Convolutional Neural Network

A Convolutional Neural Network also called ConvNet or CNN is a Deep Learning algorithm which takes in an image as input and then assigns importance i.e. weights and biases to certain aspects or the objects in a given image and be able to differentiate them from the other. The pre-processing required in a Convolutional neural network is lower as compared to the other classification algorithms available. Technically, deep learning CNN models take each image as input and then pass it through a number of convolution layers with filters (also called Kernals), Pooling layers, fully connected layers (FC) and then apply activation functions to classify the object with probabilistic values between 0 and 1.

1.1.5.1 Steps for building a Convolutional neural network

1. Firstly, we provide an input image into the convolution layer
2. Next, we choose some parameters, apply filters with paddings and strides, if required. We then perform convolution on the input image and apply activation functions.
3. We then need to perform pooling to reduce dimensionality size
4. Many convolution layers can be added as per need
5. Flatten the output and feed into a dense layer
6. Finally, train the model using the training set.

1.1.6 Image Classification VS Object Detection VS Image Segmentation

In the computer vision field, one of the most common doubts which most of us have is what is the difference between the classification of images, the detection of objects and image segmentation.

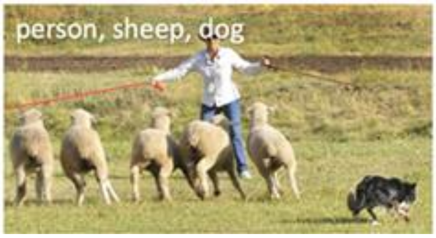


Figure 1. Classification

Figure 1 shows us image classification, A computer is able to identify the objects in the picture given to it. It can differentiate between the objects dog, person and sheep. [3] However, it does not localize the objects. It cannot pinpoint where the objects are located in the frame. This is Image classification

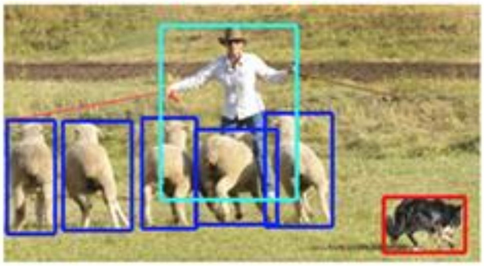


Figure 2. Detection

Figure 2 shows us image detection. A computer on processing the image given to it can identify the different objects in the picture, in this case, it can identify the objects dog, sheep and person in the picture. Furthermore, it can also localize the objects in the picture. Localization of the objects means that the exact location of the objects in the picture is also determined. A bounding box surrounding the identified object in the picture shows its location in the frame of the picture. This is object detection.

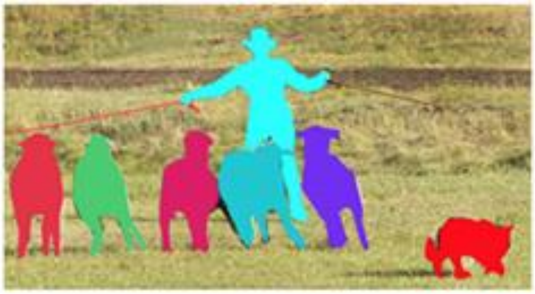


Figure 3. Segmentation

Figure 3 shows us Image segmentation. In segmentation, the objects are identified and are further visualized using color. This means that every pixel of the object in the picture is colored and so are the other objects with other colors. For example, in image 3, the object person is colored with light blue and the object dog is colored with red. This is how image segmentation is viewed.

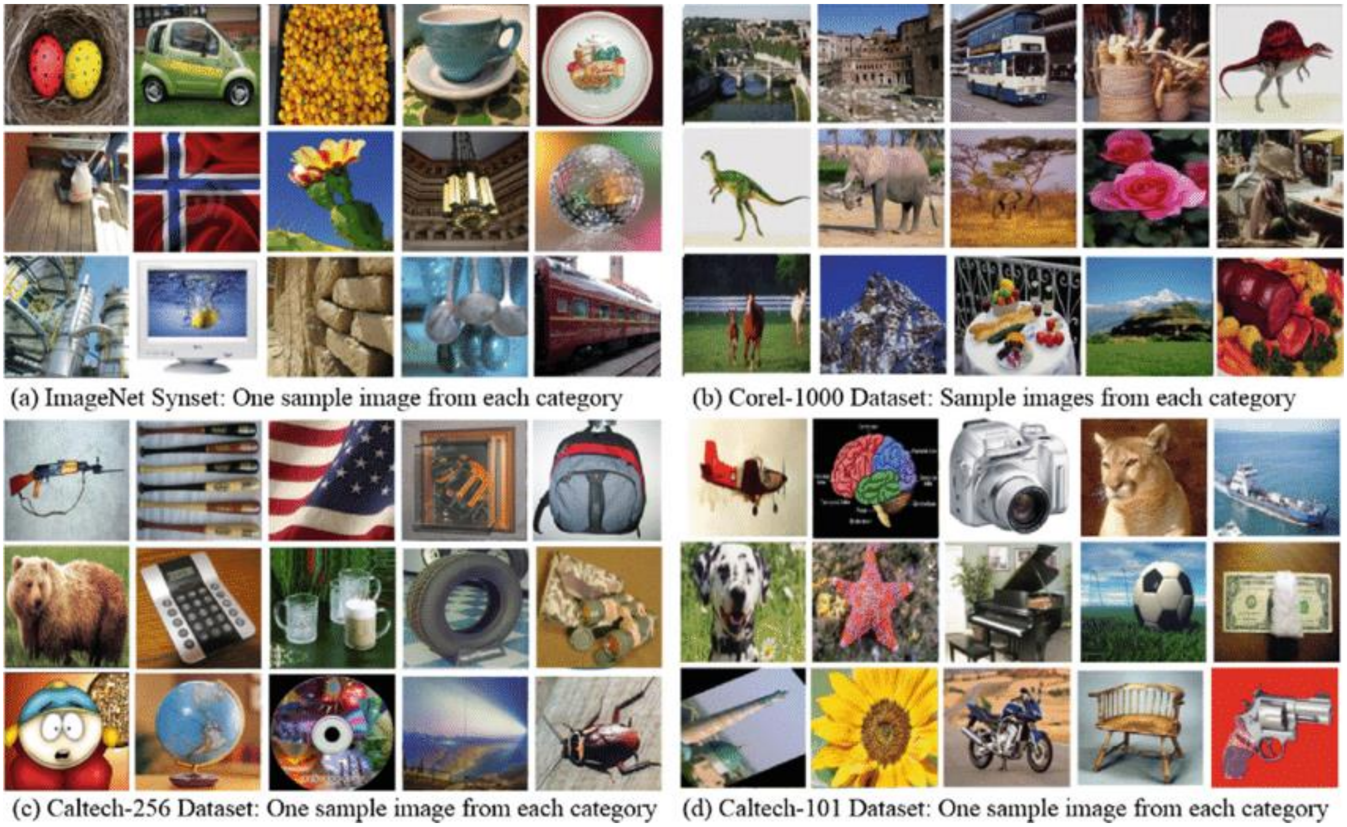
1.1.7 General object detection framework

Generally, three important steps are involved in the framework of an object detection job.

1. Firstly, a model or its algorithm is to be utilized to create areas of interest or proposal regions. Such region proposals act as a big collection of abounding boxes which span the entire image (which is a component of object localization).
2. For the next step, the optic features are then extracted for each of the abounding boxes. Then they are tested, and it is determined whether or not and also what objects are available in the proposals constructed on the optic features (that is, a component of object classification). [4]
3. For the last post processing step, all the overlapping boxes is combined to a single abounding box (which is the non-maximum suppression).

1.1.8 Popular Object Detection Datasets

Image Net**:** Image Net dataset consists of around 14 million images in total for 21,841 different categories of objects (*data as of 12th Feb 2020*). Some of the popular categories of objects in Image Net are Animal (fish, bird, mammal, invertebrate), Plant (tree, flower, vegetable) and Activity (sport).



(a) ImageNet Synset: One sample image from each category

(b) Corel-1000 Dataset: Sample images from each category

(c) Caltech-256 Dataset: One sample image from each category

(d) Caltech-101 Dataset: One sample image from each category

Figure 4. Image Net dataset

Common Objects in Context (COCO): COCO is a large-scale object detection, segmentation, and captioning dataset. It contains around 330,000 images out of which 200,000 are labelled for 80 different object categories.

Google's Open Images: Open Images is a dataset of around 9M images annotated with image-level labels, object bounding boxes, object segmentation masks, and visual relationships. It contains a total of 16M bounding boxes for 600 object classes on 1.9M images, making it the largest existing dataset with object location annotations.

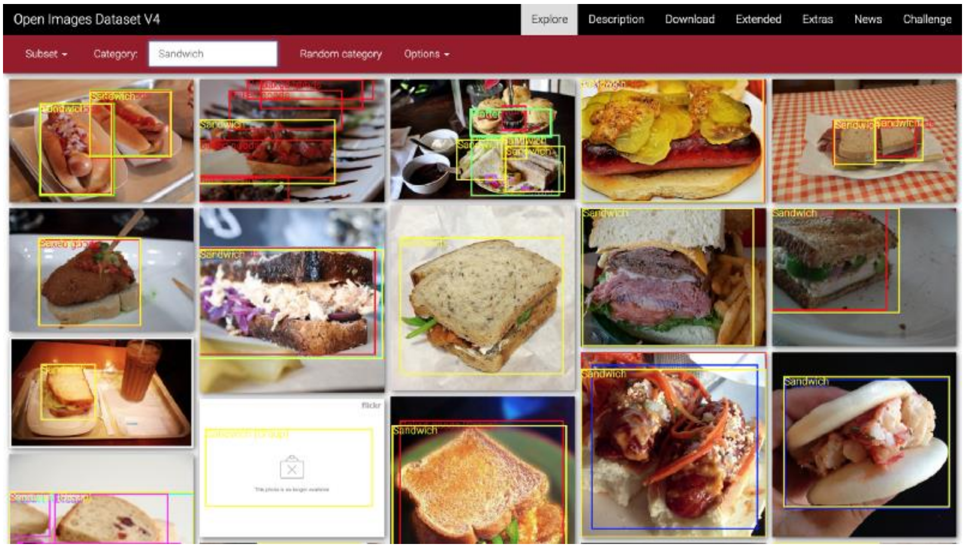


Figure 5. Google's Open Images

MNIST handwritten datasets: This dataset has a total of 70,000 images of handwritten digits and is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

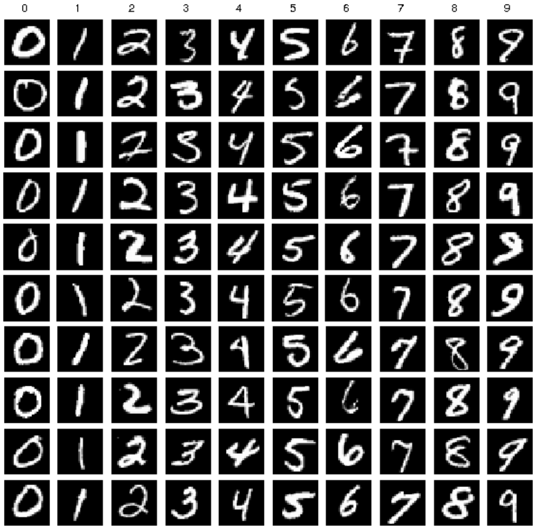


Figure 6. MNIST dataset

Cityscapes Dataset: This dataset focuses on semantic understanding of urban street scenes. It contains around 20,000 annotated images for 30 different classes.

1.2 Objectives

The task of Object detection is one of the most paramount and demanding issues in the field of computer vision. It has received attention widely in the current years. The development of this field in the previous one decade is also regarded as an epitome in the history of computer vision. Now, if we imagine the task of object detection in today's time as a technical aesthetics placed beneath the potentiality of the method of deep learning, then going back 20 years to the past, we would have witnessed the wisdom of cold weapon era. The detection of objects is a paramount task in the field of computer vision that deals with the detection of instances of a visual object of a particular class (for example animals, vehicles, or fruits) in a digital image. The aim of object detection is to create a model of computation and create a technique that could provide one of the most rudimentary pieces of information required by the applications of computer vision which is, what objects are where?

1.3 Applicability and Achievements

a) Tracking objects

An item/object detection framework is additionally utilized in tracking the objects, for instance tracking a ball during a match in the football world cup, tracking the swing of a cricket bat, tracking an individual in a video. Object tracking has an assortment of uses, some of which are surveillance and security, traffic checking, video correspondence, robot vision and activity.

b) People Counting

Object detection can be additionally utilized for People counting. It is utilized for dissecting store execution or group measurements during festivals.

c) Person Detection

Person detection is necessary and critical work in any intelligent video surveillance framework, as it gives the essential data to semantic comprehension of the video recordings. It has a conspicuous augmentation to automotive applications because of the potential for improving security frameworks. [5] Person detection is undertakings of Computer vision frameworks for finding and following individuals. The detection of a person is just the job of finding all examples of individuals present in an image, and it has been most broadly achieved via looking through all areas in the picture, at all potential scales, and contrasting a little region at every area with known layouts or examples of individuals. Person detection is commonly viewed as the initial procedure in a video surveillance pipeline and can take care of into more significant level thinking modules, for example, action recognition and dynamic scene analysis

d) Vehicle Detection

Vehicle Detection is one of the most important parts in our daily life. As the world is moving faster and the numbers of cars are kept on increasing day by day, Vehicle detection is very important. By using Vehicle Detection technique, we can detect the number plate of a speeding car or accident affected car. This also enables for security of the society and decreasing the number of crimes done by car.

e) Self-driving cars

Another unique application of object detection technique is definitely self-driving cars. A self-driving car can only navigate through a street safely if it could detect all the objects such as people, other cars, road signs on the road, in order to decide what action to take.

f) Detecting anomaly

Another useful application of object detection is definitely spotting an anomaly and it has industry specific usages. For instance, in the field of agriculture object detection helps in identifying infected crops and thereby helps the farmers take measures accordingly. It could also help identify skin problems in healthcare. [5] In the manufacturing industry the object detection technique can help in detecting problematic parts really fast and thereby allow the company to take the right step.

2. Survey of Technologies

2.1 OpenCV

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e., whatever operations one can do in Numpy can be combined with OpenCV.

2.2 Labeling

Labeling is a graphical image annotation tool. It is written in Python and uses Qt for its graphical interface. Annotations are saved as XML files in PASCAL VOC format, the format used by Image Net. Besides, it also supports YOLO format. Image Net: is a large dataset of annotated photographs intended for computer vision research. The goal of developing the dataset was to provide a resource to promote the research and development of improved methods for computer vision.

2.3 Yolo

YOLO is a clever convolutional neural network (CNN) for doing object detection in real-time. [6] The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

3. Object Detection and its Algorithms

a) **Fast R-CNN**

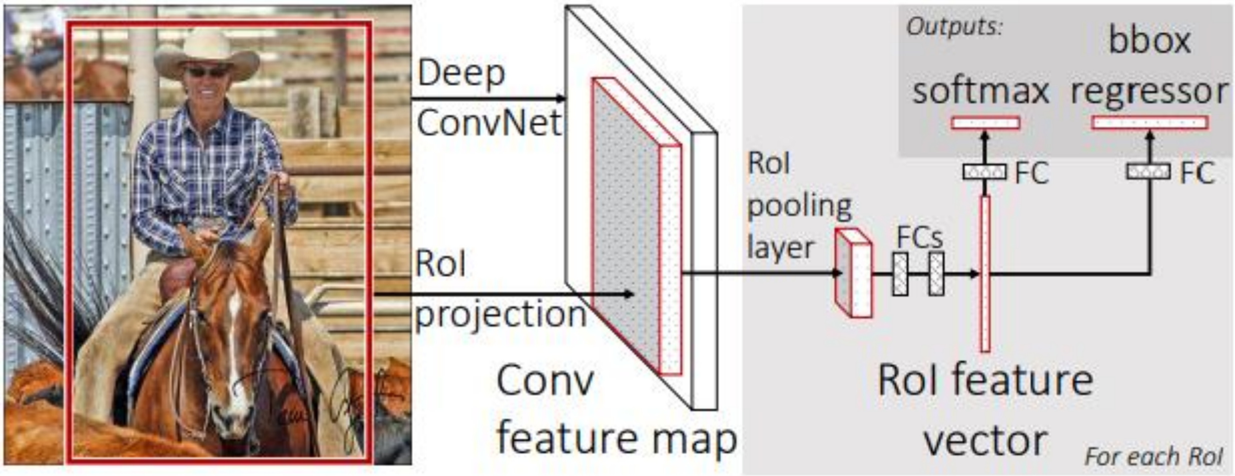


Figure 7. Fast R-CNN

Figure 7 shows the working of the fast r-CNN object detection example by taking the example picture of a man on a horse.

Written in the language of Python and in the language of C++ (Caffe), Fast Region-Based Convolutional Network process (also referred as Fast R CNN) is a training algorithm for the detection of objects. This algorithm mainly fixes the disadvantages of R CNN and SPP net, with upgrading on its speed and correctness.

Advantages of Fast R-CNN: –

- It provides a high detection quality (mAP), more compared to the RCNN or SPP net.
- It performs it training in a single stage while also utilizing a multitask loss.
- The training performed could enhance all of the network layer.
- No type of disk storage is needed for feature caching.

b) Faster R-CNN

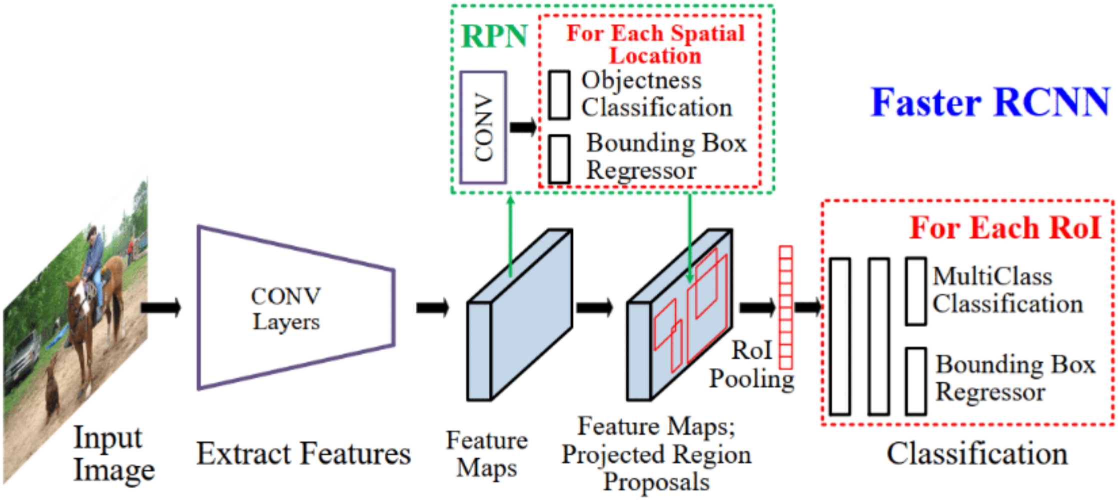


Figure 8. Faster R-CNN

Figure 8 shows us the working of the faster R-CNN object detection algorithm.

Faster R-CNN is an object detection algorithm that is similar to R-CNN. This algorithm utilizes the Region Proposal Network (RPN) that shares full-image convolutional features with the detection network in a cost-effective manner than R-CNN and Fast R-CNN.[7] A Region Proposal Network is basically a fully convolutional network that simultaneously predicts the object bounds as well as objectness scores at each position of the object and is trained end-to-end to generate high-quality region proposals, which are then used by Fast R-CNN for detection of objects.

c) Histogram of Oriented Gradients (HOG)

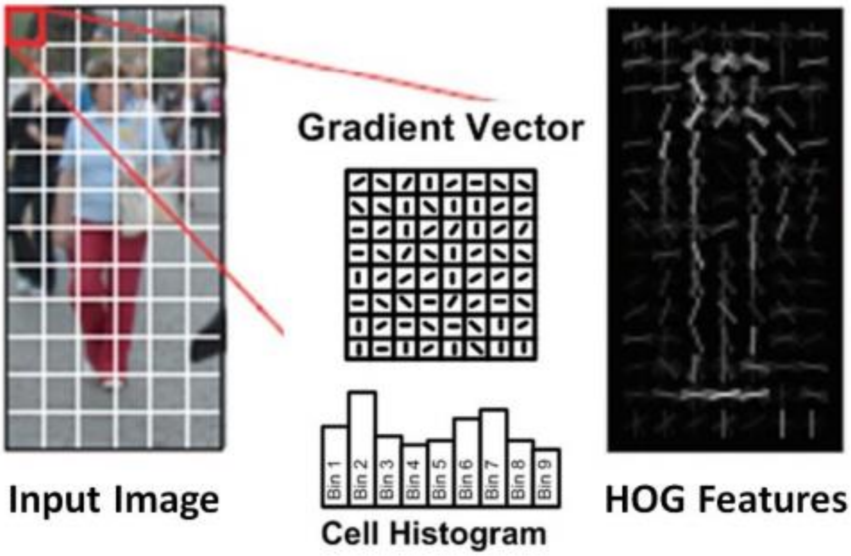


Figure 9. Histogram of oriented gradients (HOG)

Figure 9 shows us the working of the histogram of gradient oriented detection of object algorithm by taking a image as the input data. Then its HOG features are shown through an image depiction. Histogram of oriented gradients (HOG) is basically a feature descriptor that is utilized to detect objects in image processing and other computer vision techniques. The Histogram of oriented gradients descriptor technique includes occurrences of gradient orientation in localized portions of an image, such as detection window, the region of interest (ROI), among others. One advantage of HOG-like features is their simplicity, and it is easier to understand the information they carry.

d) Region-based Convolutional Neural Networks (R-CNN)

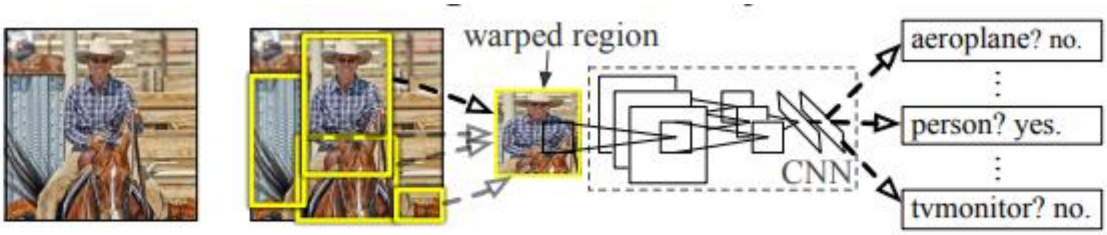


Figure 10. R-CNN

Figure 10 shows us the working stages of the Region based convolutional network-based algorithm used in object detection. The Region based Convolutional Network process (RCNN) is a combination of region proposals with Convolution Neural Networks (CNNs). [8] R-CNN helps in localizing objects with a deep network and training a high-capacity model with only a small quantity of annotated detection data. It achieves excellent object detection correctness by utilizing a deep ConvNet in order to classify the object proposals. R-CNN has the capability to scale to thousands of object classes without resorting to approximate techniques, including hashing.

e) Region-based Fully Convolutional Network (R-FCN)

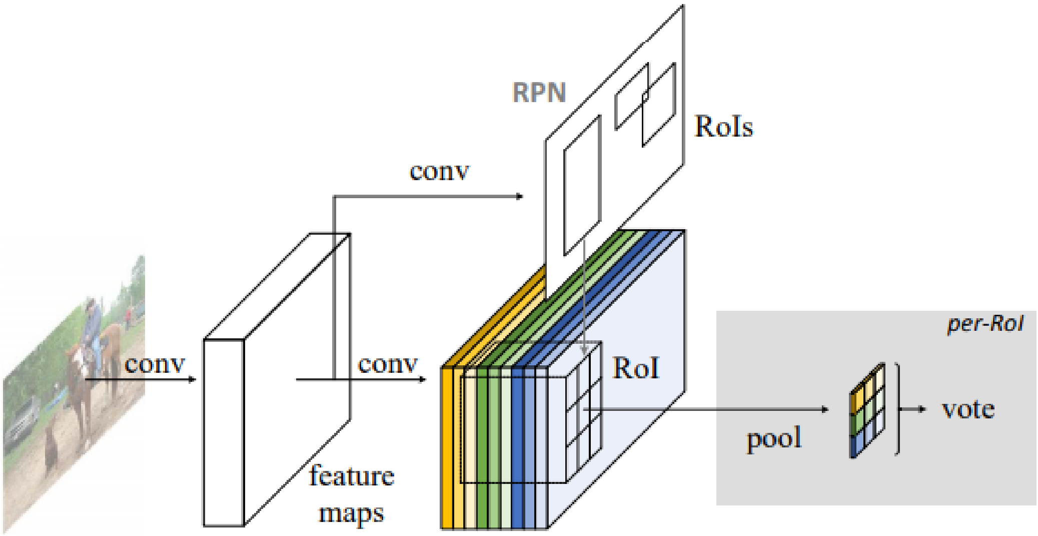


Figure 11. R-FCN

Figure 11 shows us the internal working of the region based fully convolutional network-based algorithm which is popularly used in object detection cases. Region-based Fully Convolutional Networks or R-FCN is a region-based detector for object detection. Unlike other region-based detectors that apply a costly per-region subnetwork such as Fast R-CNN or Faster R-CNN, this region-based detector is fully convolutional with almost all computation shared on the entire image. R-FCN consists of shared, fully convolutional architectures as is the case of FCN that is known to yield a better result than the Faster R-CNN. In this algorithm, all learnable weight layers are convolutional and are designed to classify the ROIs into object categories and backgrounds.

f) Single Shot Detector (SSD)

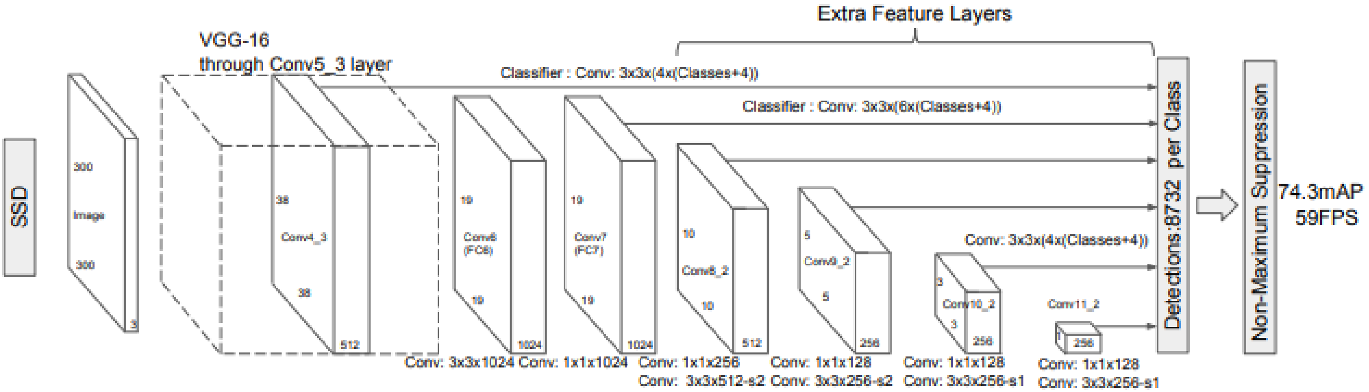


Figure 12. SSD

Figure 12 shows us the step by step working of the single shot detector-based object detection algorithm. Single Shot Detector (SSD) is a method for detecting objects in images using a single deep neural network. The SSD approach discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios. After discretizing, the method scales per feature map location. [5] The Single Shot Detector network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes.

Advantages of SSD: –

- SSD completely eliminates proposal generation and subsequent pixel or feature resampling stages and encapsulates all computation in a single network.
- Easy to train and straightforward to integrate into systems that require a detection component.
- SSD has competitive accuracy to methods that utilize an additional object proposal step, and it is much faster while providing a unified framework for both training and inference.

g) Spatial Pyramid Pooling (SPP-net)

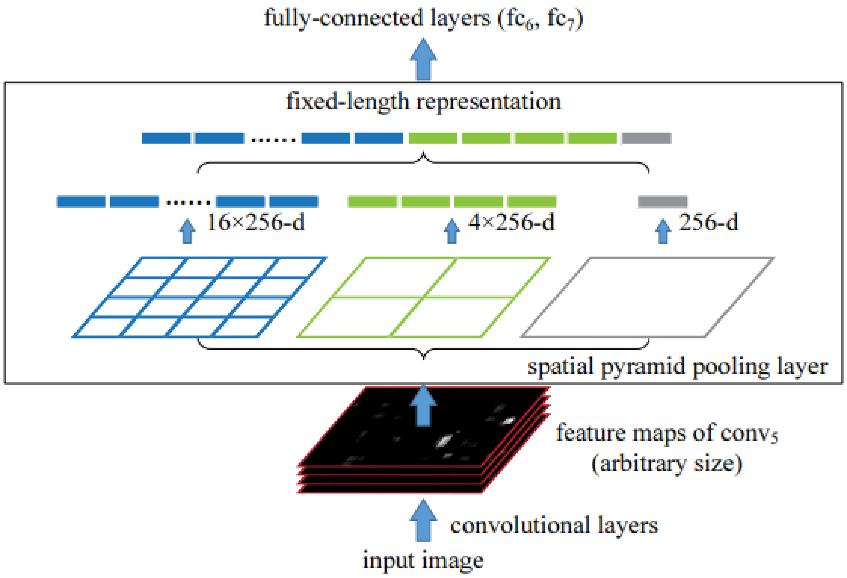

fully-connected layers (fc_6, fc_7)
fixed-length representation
16×256-d
4×256-d
256-d
spatial pyramid pooling layer
feature maps of $conv_5$
(arbitrary size)
convolutional layers
input image

Figure 13. SPP-net

Figure 13 shows us the working of the spatial pyramid pooling network structure used in object detection problems. Spatial Pyramid Pooling also known as the SPP net is a network structure that could build a length of fixed representation notwithstanding the image size/scale. Pyramid pooling is said to be robust to object deformations, and SPP-net improves all CNN-based image classification methods. [9] Using SPP-net, researchers can compute the feature maps from a full image only once. Then the pool features present in the random areas, also called sub-images, are utilized to create the length representations which are of fixed length for the training of the detectors. This process steer clears of the constantly computing the convolutional features.

h) YOLO (You Only Look Once)

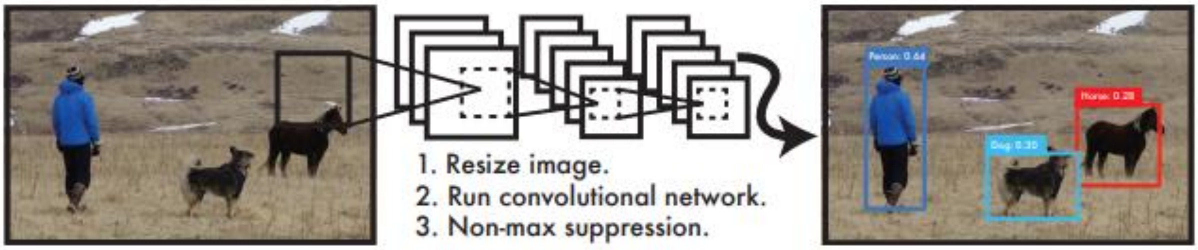


Figure 14. YOLO

Figure 14 shows us the working of the you only look once object detection algorithm. You Only Look Once or YOLO is one of the popular algorithms in object detection used by the researchers around the globe. According to the researchers at Facebook AI Research, the unified architecture of YOLO is extremely fast in manner. The base YOLO model processes images in real-time at 45 frames per second, while the smaller version of the network, Fast YOLO processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. This algorithm outperforms the other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

4. Requirement and Analysis

4.1 Problem definition

In this project, we are building a custom object detection model that can be used to make detections in real time. In our case, we will be using the object detection model to make real time detections of a face mask. This model will be able to detect if a person is wearing a mask or not.

4.2 Planning and Scheduling

We will install labeling and label images from scratch for object detection, train and run a custom detector for face mask detection, use transfer learning to train on top of existing models and then make detections in real time using the trained model. Using this method, based on deep learning techniques, which is also established on the grounds of machine learning, need a lot of mathematical and deep learning frameworks comprehending by using dependencies such as Tensor Flow, Open CV, etc., with which we can detect each and every object present in the image by the area object in a highlighted rectangular box and can then identify each and every object and give a tag to the object for its identification.

5. System Design

5.1 The Dataset Design

For the working project of Real time mask detection to be created, a dataset would be required. The purpose of the dataset would be to serve as input to the model which would then learn from the input data and then use the gather knowledge to detect the output, which in our case, would be to detect masks. There are many available datasets for this purpose; however, a self-made dataset has been used in this project. Figure 15 show the images used to create the dataset. Pictures with and without masks were taken to build the dataset.

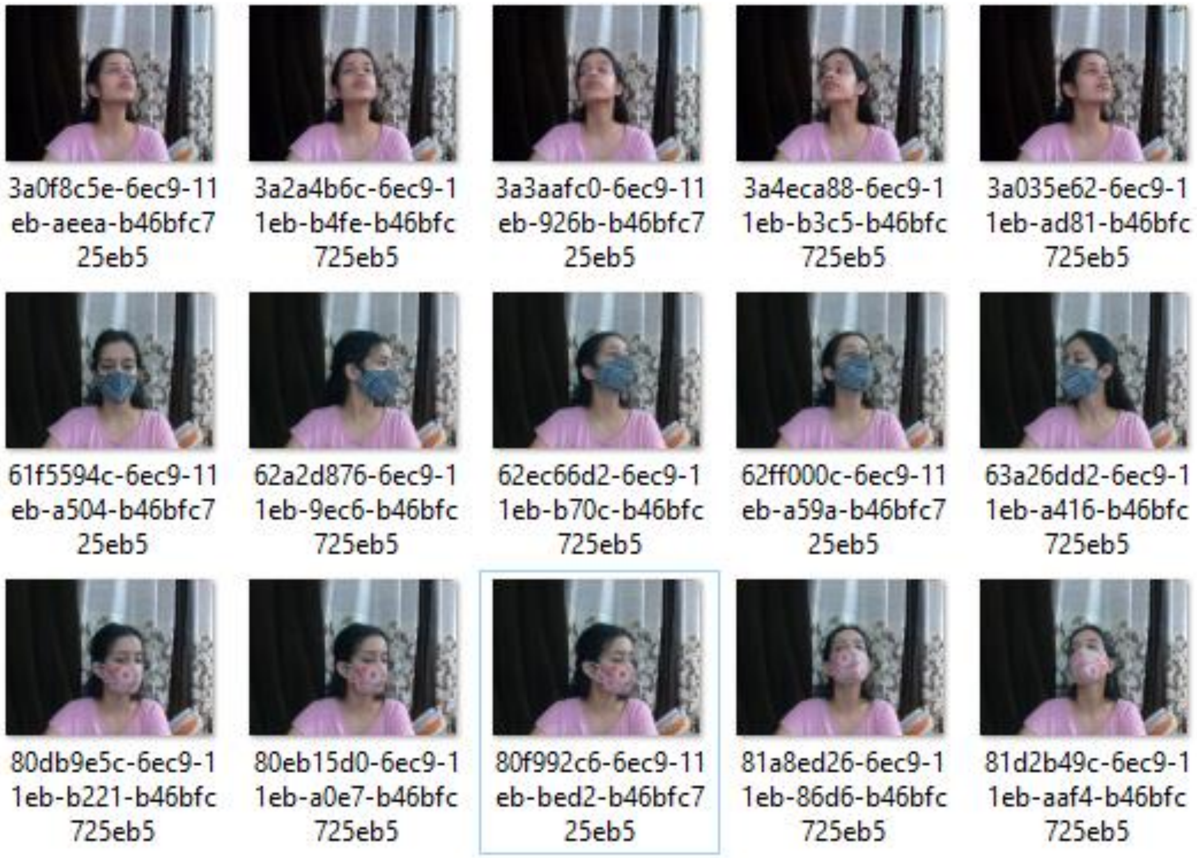

3a0f8c5e-6ec9-11eb-aeea-b46bfc725eb5
3a2a4b6c-6ec9-11eb-b4fe-b46bfc725eb5
3a3aafc0-6ec9-11eb-926b-b46bfc725eb5
3a4eca88-6ec9-11eb-b3c5-b46bfc725eb5
3a035e62-6ec9-11eb-ad81-b46bfc725eb5
61f5594c-6ec9-11eb-a504-b46bfc725eb5
62a2d876-6ec9-11eb-9ec6-b46bfc725eb5
62ec66d2-6ec9-11eb-b70c-b46bfc725eb5
62ff000c-6ec9-11eb-a59a-b46bfc725eb5
63a26dd2-6ec9-11eb-a416-b46bfc725eb5
80db9e5c-6ec9-11eb-b221-b46bfc725eb5
80eb15d0-6ec9-11eb-a0e7-b46bfc725eb5
80f992c6-6ec9-11eb-bed2-b46bfc725eb5
81a8ed26-6ec9-11eb-86d6-b46bfc725eb5
81d2b49c-6ec9-11eb-aaf4-b46bfc725eb5

Figure 15. Images used to create Dataset

6. Implementation and Testing

6.1 Implementation approaches

The process for object Detection occurs in 4 important steps-

a. **Gathering of data**- the first and foremost step is to feed some data to the computer through which our model learns. The data can be in any kind of format such as text files or excel sheets.

b. **Data preparation**- this step involves taking only the useful information from the whole data set which is fed into the computer. The data which is really useful for the model for processing is only taken into consideration and the rest is discarded. This step also involves removing the unwanted data, checking for values which are missing and treatment of the outliers.

c. **Training set and testing set**- after that data is filtered in the second step, that data is then divided into 2 sets. [10] One of the sets is called training set which is used for creating the models and the second set, i.e., test sets are then used to check the accuracy of the model created.

d. **Evaluation**- this step involves Evaluating the model. To verify the accuracy of the created model, the model is tested on some data which was not present in the data used for its creation. Here, the test data is used.

6.2 Modes and types of Object Detection

Without regards to specific details, object detection is spilt down in to two modes or types, approaches based on machine learning and approaches based on deep learning.

a) Machine-learning based approaches

In a more conventional machine learning based approaches, the techniques arising from the field of computer vision can be used to see different types of features present in an image, for example, the technique may focus on the color histogram or certain edges, which can be used to detect groups of pixels which may belong to an object present in the image. Such features are then used as input into a model of regression which can predict the object location of the object in the image and along with its location, it can show the label.

b) Deep learning-based approaches

However, on the other hand, approaches based on deep learning use the concept of convolutional neural networks, in short, CNN. [11-16] Such approaches use convolutional neural network to perform end to end, unsupervised detection of objects. In such cases, there is no need for the features to be defined and extracted individually unlike approaches based on machine learning.

6.3 Coding details and Code efficiency

To begin with the project, the first and foremost task was the creation of the dataset. A data set consisting of images of some person with a mask and without a mask. The dataset would have to be large enough to prove effective for the model to make accurate predictions. OpenCV and Labelimg was used to accomplish this task of building a self-made dataset for the purpose of serving as an effective input data for the soon-to-be created model, which in turn, would perform the task of object detection. For the installation of LabelImg, the below code was run in the command prompt after installing python. Pyqt5 and lxml.

```
pyrcc4 -o libs/resources.py resources.qrc
```

Then the below command was used to run the python file, LabelImg, in the directory it was installed.

```
python labelImg.py
```

To begin with the task of building the dataset for the model, we first import two dependencies, cv2 and uuid.

Uuid: UUID that stands for Universal Unique Identifier, is a python library which helps in generating random objects of 128 bits as ids. It provides the uniqueness as it generates ids on the basis of time, Computer hardware (MAC etc.). After importing the dependencies, a video capture is to be created. This would be used to take the pictures from the webcam installed on a PC. Internal or external webcams may be used in the process.

```
cap=cv2.VideoCapture(0)
```

Using the Video capture class from the cv2 module we pass the value of number zero. It is the video capture that is our webcam in our device. The output of this code was stored int the variable 'cap'.

```
width=int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height=int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
```

The parameters CAP_PROP_FRAME_WIDTH and the parameter CAP_PROP_FRAME_HEIGHT from the cv2 module, is used to extract the frame width and height that will allow us to capture the dimensions of our specific webcam.

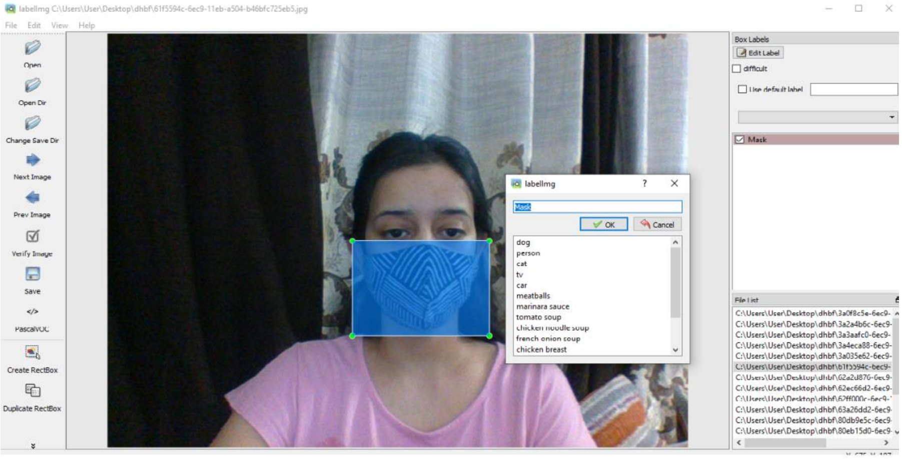
```
while True:
    ret, frame = cap.read()
    imgname = './Images/mask/{}.jpg'.format(str(uuid.uuid1()))
    cv2.imwrite(imgname, frame)
    cv2.imshow('frame', frame)

    if cv2.waitKey(1) & 0xFF==ord('q'):
        break
```

The first line allows us to capture an image from our video capture device. It allows us to grab our current frame. Then we store the image, that is, our captured frame and also define its path. With the uuid module, along with the images, we are also creating a unique id for each of the image stored. Then we use the imwrite and imshow functions from the cv2 module to write and show the frame that was captured. The waitkey function from the cv2 module is used to quit out of our cv2 capture device and to check for a key pressed event and break the loop.

```
cap.release()
cv2.destroyAllWindows()
```

The above code is used to release our video capture device. We use the destroyAllWindows() function from the opencv module to quit from the window. On running the above code, the webcam associated with our device opens up in a new window. With every frame, an image is captured and stored in the folder ‘images’ in the current working directory where the code is running. This folder has another two folders inside it. In one of them we store the images taken with a mask and the other stores the images we take without a mask. So, we ran this code three times. One time without wearing any mask and the other times while wearing two different masks to increase the variety of our dataset. On pressing any key while the webcam is open, there is a keyboard interrupt which forces the webcam to stop and then is further released.



labelImg C:\Users\User\Desktop\dhbf\61f5594c-6ec9-11eb-a504-b46bfc725eb5.jpg

Mask

dog
person
cat
tv
car
meatballs
marinara sauce
tomato soup
chicken noodle soup
french onion soup
chicken breast

☐ difficult

☐ Use default label

☑ Mask

File List

C:\Users\User\Desktop\dhbf\3a0f8c5e-6ec9-
C:\Users\User\Desktop\dhbf\3a2a4b6c-6ec9-
C:\Users\User\Desktop\dhbf\3a3aafc0-6ec9-
C:\Users\User\Desktop\dhbf\3a4eca88-6ec9-
C:\Users\User\Desktop\dhbf\3a035e62-6ec9-
C:\Users\User\Desktop\dhbf\61f5594c-6ec9-
C:\Users\User\Desktop\dhbf\62a2d876-6ec9-
C:\Users\User\Desktop\dhbf\62ec66d2-6ec9-
C:\Users\User\Desktop\dhbf\62ff000c-6ec9-
C:\Users\User\Desktop\dhbf\63a26dd2-6ec9-
C:\Users\User\Desktop\dhbf\80db9e5c-6ec9-
C:\Users\User\Desktop\dhbf\80eb15d0-6ec9-

Figure 16. LabelImg

Our next step is to use the LabelImg annotation tool to label our captured images. To do this, we open the labelImg python file. We open the directory where are images were stored and one by one start labeling them. In the images taken without a mask, we select the mouth area where the mask should be. A bounding box appears, and we label it as ‘No mask’. Then, using the images taken while wearing the different masks, we label the area covered by the mask as ‘Mask’. The labelimg process is shown in Figure 16. We do this for all the images and in the end, we get double the number of files. This is because with our image we now also have our xml annotation file. Each image has its own annotation xml file with it. After this, we keep some of the images and their annotations files separate which would serve as the train set for our model to learn from, whereas the rest of the images and their annotation files will serve as our test set. Our dataset is created.

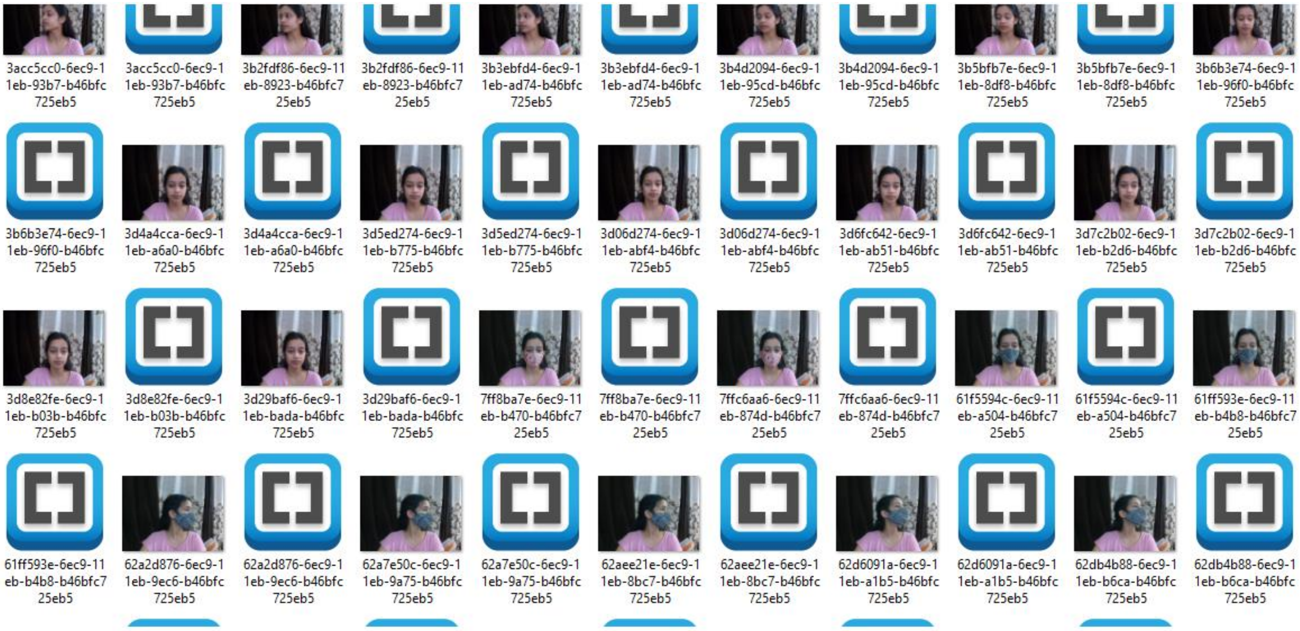

3acc5cc0-6ec9-11eb-93b7-b46bfc725eb5
3acc5cc0-6ec9-11eb-93b7-b46bfc725eb5
3b2fdf86-6ec9-11eb-8923-b46bfc725eb5
3b2fdf86-6ec9-11eb-8923-b46bfc725eb5
3b3ebfd4-6ec9-11eb-ad74-b46bfc725eb5
3b3ebfd4-6ec9-11eb-ad74-b46bfc725eb5
3b4d2094-6ec9-11eb-95cd-b46bfc725eb5
3b4d2094-6ec9-11eb-95cd-b46bfc725eb5
3b5bfb7e-6ec9-11eb-8df8-b46bfc725eb5
3b5bfb7e-6ec9-11eb-8df8-b46bfc725eb5
3b6b3e74-6ec9-11eb-96f0-b46bfc725eb5
3b6b3e74-6ec9-11eb-96f0-b46bfc725eb5
3d4a4cca-6ec9-11eb-a6a0-b46bfc725eb5
3d4a4cca-6ec9-11eb-a6a0-b46bfc725eb5
3d5ed274-6ec9-11eb-b775-b46bfc725eb5
3d5ed274-6ec9-11eb-b775-b46bfc725eb5
3d06d274-6ec9-11eb-abf4-b46bfc725eb5
3d06d274-6ec9-11eb-abf4-b46bfc725eb5
3d6fc642-6ec9-11eb-ab51-b46bfc725eb5
3d6fc642-6ec9-11eb-ab51-b46bfc725eb5
3d7c2b02-6ec9-11eb-b2d6-b46bfc725eb5
3d7c2b02-6ec9-11eb-b2d6-b46bfc725eb5
3d8e82fe-6ec9-11eb-b03b-b46bfc725eb5
3d8e82fe-6ec9-11eb-b03b-b46bfc725eb5
3d29baf6-6ec9-11eb-bada-b46bfc725eb5
3d29baf6-6ec9-11eb-bada-b46bfc725eb5
7ff8ba7e-6ec9-11eb-b470-b46bfc725eb5
7ff8ba7e-6ec9-11eb-b470-b46bfc725eb5
7ffc6aa6-6ec9-11eb-874d-b46bfc725eb5
7ffc6aa6-6ec9-11eb-874d-b46bfc725eb5
61f5594c-6ec9-11eb-a504-b46bfc725eb5
61f5594c-6ec9-11eb-a504-b46bfc725eb5
61ff593e-6ec9-11eb-b4b8-b46bfc725eb5
61ff593e-6ec9-11eb-b4b8-b46bfc725eb5
62a2d876-6ec9-11eb-9ec6-b46bfc725eb5
62a2d876-6ec9-11eb-9ec6-b46bfc725eb5
62a7e50c-6ec9-11eb-9a75-b46bfc725eb5
62a7e50c-6ec9-11eb-9a75-b46bfc725eb5
62aee21e-6ec9-11eb-8bc7-b46bfc725eb5
62aee21e-6ec9-11eb-8bc7-b46bfc725eb5
62d6091a-6ec9-11eb-a1b5-b46bfc725eb5
62d6091a-6ec9-11eb-a1b5-b46bfc725eb5
62db4b88-6ec9-11eb-b6ca-b46bfc725eb5
62db4b88-6ec9-11eb-b6ca-b46bfc725eb5

Figure 17. Dataset With annotations

Figure 17 shows all the images stored with their annotation files in xml format. We start with setting up with our paths. We have a number of folders in our directory which is to be used so to make it easier when referring to these folders, we set up the paths. The next step is to create a label map. Label map is just a representation of all the different objects we can expect to find within our trained annotations. We create a dictionary called ‘labels’ and output it to a labelmap. Since we are working with the tensorflow object detection library, this needs to be in the format of pb texts, that is, protobuff file. We go into our annotation file and we create a new file called labelmap.pb text and we set it to write because we will write to it. Then we loop through the two labels we have, that is, mask and no mask. Using the f.write function, we write the ‘item’, ‘id’ and ‘name’ to the labelmap file.

After creating our label map, we create the tf records. Tf records stands for tensor flow records. You can use the generate_tfrecord file from this official object detection api. The generate_tfrecord file is critical because it will make sure that we get out data in the right format. Also, since we are working with a large dataset, using a binary file format like tf record for storage of our data can have an impact on the performance of our import pipeline and as a result on the training time of your model. This is because data in the form of binary takes up much less space on the disk and hence takes less time to copy and so is read more efficiently from the disk. We use the python command inside of oue notebook to run the generate_tfrecord file to create the tf records. We do it for our training and testing images. Our next step is to download the tensorflow models from the tensorflow model zoo. We will be using the SSD_mobilenet v2 model from the tensorflow model zoo. We then need to import certain dependencies. We import tensorflow, connfig_util from the object detection.utils module, pipeline_pb2 from the object detection.protos module and the text_format from the google.protobuf module. We then need to open up our config file. So, we set up a config path to that file. We use the get _configs _from _pipeline _file method from the config util to grab that config file. We then make changes to the config file according to our requirememnts.

Our next step is to create a template pipeline config to make the changes. The changes that we make are first to change the number of classes that we want to predict and the batch size (which depends on our hardware capacity). In the presence of a gpu, we make take a larger batch size or in the absenece of it, we can bump it down to a lower value. We then use the checkpoint from pre-trained model to tell the system to start training from that point. We get the checkpoint from the tensorflow model zoo. We specify that our model type is of object detection, the path to our label map, and the path to our training records. The current pipeline config file is in the memory of our notebook, so we then store it to the directory. We use the text_format utility from the google. protobuf module to convert our protobuf message to a string and then using the f.write function we write it out. Then to train our model, we used the below command and ran it in the command terminal.

```
python Tensorflow/models/research/object_detection/model_main_tf2.py --model_dir= Tensorflow/workspace/models/my_ssd_mobnet --pipeline_config_path= Tensorflow/workspace/models/my_ssd_mobnet/pipeline.config --num_train_steps= 5000
```

What we have done is used string formatting to pass through our different paths. The baseline script to go and train the model is python and we use the model main_t2f.pi. we are using tensorflow 2 for this particular model. We then pass through where our model directory is and then we pass through where our pipeline config file is. We also then specify the number of our training steps that our model will go through. This meand that we are telling the model how many times we want it to go through the training set. We set it to 5000 number of training steps in our case. We can change the number of training steps as per requirements. It depends on how accurate we want our model to be. On running the python command in the command prompt terminal in the working directory, we can see the loss metric appearing in the terminal. We can also see the time the model takes to train per step and as well as our current loss. After the training of our model, we need to load our model from the checkpoint. Before this we need to import some more dependencies. We imported the OS module. We also import three things from the object detection module. We import the label_ map.util utility from the object detection utils library which will allow us to work with our label map, visualization_ utils from the object _detection library, which will allow us to transform our detection to an overlay on our image so we can actually see the bounding boxes appear during the real time detection and the model _builder utility from the object _detection. builder's library which will allow us to build our model from our checkpoint and our config file. Our next step is to grab our pipeline config file and build our model using the model config and we specify that it is not training at the moment. We then restore our checkpoint. In this case, we have gone and restored the checkpoint 6, which is our latest checkpoint, from the model we trained.

After restoring the latest checkpoint from our trained model, we have to write the code to detect in real time. What we need to do is create a function to which we pass our image to and then we get our detections based on the image passed to our function created. To do this, we specify a TensorFlow function. We name the specified TensorFlow function as 'detect _function' using the standard python formatting by using the def function and passing the image as the argument to the function. Then we tell the function to pre-process the image by resizing it down to 320 by 320. Then we will use our detection model which we defined earlier to go and make the detections. Once model detection is done, the function will pre- process it again. Then our TensorFlow function, detect _function will return the detections made by the model on the image passed to it as an argument. Now to detect in real time, we will be using our webcam to detect in real time so we will be able to see through the webcam whether the person is wearing a mask or whether the person is not wearing a mask.

We import the dependencies, cv2 which is open cv and then import NumPy as np. This will help us work with our images. We then need to create a category index which would work as a representation of our label map and finally set up our capture from the webcam. So, in order to create the category index, we use the label map utilities, and we use the function create_ category_ index_ from _label _map from the label map utilities. To that function we pass the path to our label map. We can then output this to see that it is similar to the label map we created but in the case of this category index, we have got a key per label map. So key 1 will be assigned to the label mask whereas on the other hand, key 2 will be assigned to the label no mask. Now to set up our video capture device, we use video capture function from the open cv module which will allow us to access our webcam. Then using the capture width and height function from the open cv module we will get the dimensions of our webcam, that is, the width and height of the webcam. The next step is to be loop through, set up the webcam, make the detections in real time and render the results to the screen.

6.4 Testing Approach

We set up a loop which will keep running while true. We get our frame from the video capture we set up earlier. Then we have to convert that frame into a NumPy array. The reason we have to convert our frame to NumPy is so that it can work with TensorFlow. We then convert the NumPy array to a TensorFlow tensor and to that we pass np. expand (dim) which will just put it inside of another array. We need to put it inside of another array because TensorFlow or our model expects multiple images to come through. We set up the d type to float 32. Then we actually go and make our detections. To do this, we use the detect function we created and defined earlier. To our detect function, we pass our input tensor as an argument. Then we are grab our number of detections. We need to grab our number of detections because we will then need to perform some preprocessing on them. We then add those preprocessed number of detections back. We then go ahead and grab our detection classes. Then we specify a constant for our label id offset. We do this so when we get our detections, they start at zero, but we want them to start at one because the category index we created earlier will start from the number one.

Then a copy of our detections is to be made. To make the copy of our detections we need to grab the image np and we name it as image np with detections. This is because when we pass it to this function visualize_ boxes_ and_ images_ array, the function will actually transform the array. We ideally, we need to keep a backup in case we ever need to make any changes to them. So, to the visualize_ boxes_ and_ images_ array function we pass image_ np_ with_ detections, which is the raw image from our webcam and then to that function we pass the detection boxes, detection classes and as well as the detection scores. We pass all the mentioned arguments so that when we go ahead with the real time detection, we are able to see the bounding boxes telling us if there is a mask or not on the face of the person. Addition to the bounding boxes, we also pass the scores and label as arguments to the function, so that with the bounding boxes we can see the label name and the score in the top left-hand corner of the bounding box. We then pass our category index, whether or not we want to use the normalized coordinates which in our case we do want to use because it will make sure that the bounding box is accurately positioned on the face. We set the parameters max_ boxes_to _draw to 1, min_ score_ threshold to 0.3 and we set the agnostic_ mode value to false as we do not want it to differentiate between mask or no mask.

We then display this back to the screen. We grab the transformed array and show it using the imshow function from the cv2 library. To break out of it we set up the regular wait key from the cv2 module and use the release function to release the webcam. On running the above commands, a pop-up window opens up. This pop- up window is our webcam ready to make the real time mask detections.

7. Limitations in Object Detection

a) Variable number of objects

There are a number of hurdles while performing an object detection task. One of them is the existence of a number of variable objects. To understand why that is a problem, we need to think of it as the following way. While training a machine learning model, we usually require representing the data into a size of fixed vectors. Now because the number of objects in the given image is not known to us from previous, we will not get the accurate number of outputs. So, because of this issue, some type of post processing is needed, which would add a certain complexity to our model.

b) Sizing

In addition to the above challenge, a bigger one is the various conceivable sizes of object in the given input image. While performing basic classification tasks, we expect and want to classify object which could cover most of the image. On the other hand, certain objects we may need to find could be so small like a dozen pixel (or maybe even a tiny percentage of the original image). Traditionally this problem has been solved by utilizing the sliding window of various sizes, which proves to be simple but however also proves to be inefficient.

c) Dual priorities: the classification and the localization of objects

One of the major challenges faced in the process of object detection is that it of its additional goals, that is, not only do we require to classify the images objects, but we also need to know the position of the objects, typically referred to as the task of localization of objects.

d) Speed for real-time detection

The algorithms pertaining to the detection of objects require not only to correctly classify also localize critical objects, but it also requires to be extremely quick at predicting the results in time to meet the real time needs of video processing.

8. Future Scope of Object Detection

The task of finding objects belonging to classes of interest in images has long been a focus of Computer Vision research. The ability to localize objects is useful in many applications: from self-driving cars, where it allows the car to detect pedestrians, bicyclists, road signs, and other vehicles, to security, where intruding persons can be detected. Though a lot of progress has been made since the conception of the field of Computer Vision more than five decades ago, as always, there is scope for further improvement. This is especially true in the case of object detection where a myriad of factors including variation in object instances through pose and appearance, along with other environmental factors such as the degree of occlusion, and lighting tend to create certain failures. In this work we focus on improving the task of detecting objects through the use of more representative features and better models. We propose new features that are not only more powerful, but also more robust and capture more information than the currently popular features. Further, we propose scalable models which can leverage large amounts of training data to improve performance.

9. Conclusion

Object detection is still an active research area. Although the general landscape of this field is well shaped by a two-stage detector like R-CNN and one-stage detector such as YOLO, our best detector is still far from saturating the benchmark metrics, and also misses many targets in complicated background. A huge amount of technological break throughs was promised to us, in the year 2020. Although not much has been achieved as claimed, yet our advancement and the implementation of many revolutionary technologies in the realm of Artificial Intelligence, Deep Learning, and Machine Learning has actively contributed to re-fining object detection technology. The dawn of driver-less car, impressive facial-recognition mechanic has already been implemented in different countries, and the faster detect of objects in real time with good accuracy were made possible through the use of object detection algorithms, aided by machine learning. Computer Vision is rapidly developing field in Computer Engineering. It has applications in about each other field of study and is as of now being actualized industrially in light of the fact that object detection can take care of issues excessively troublesome or tedious for people to explain.

Acknowledgement

First of all, I would like to thank Dr. Pawan Singh for steering me in the right direction in my beginnings. When I look at the amount of knowledge, I have gained till date in my B. Tech. journey, I realize how much time and effort spent toward the completion of this work; not only by me but also by key individuals whom I feel very indebted to. I gratefully acknowledge the support and guidance of my supervisor Prof. Dr. Pawan Singh. I would not imagine completing this work without all his advice. I acknowledge the time he spent in our weekly meetings, even with his busy schedule and other commitments. I am deeply indebted to him for his patience. He assisted in all aspects of this work from discussing new ideas to writing and completing this project. I would like to thank Amity University, Lucknow Campus for providing me a wonderful platform for this work. I would also like to thank the library for the knowledge I acquired with its help. I am highly obliged to my HOD, Dr. Deepak Arora and my HOI, Wg. Cdr. Anil Tiwari for helping me in the completion of this report. Lastly, I want to thank my family for supporting me entirely. I would like to express my deep gratitude and appropriation to my family and friends who were very cooperative and kind in getting me on the road.

References

[1] S. Srinivas, R. K. Sarvadevabhatla, K. R.Mopuri, et.al, “A taxonomy of deep convolutional neural nets for computer vision,” Frontiers in Robotics and AI, vol. 2, no.36, pp. 1-13, 2016.

[2] A. Krizhevsky, & G.Hinton “Learning multiple layers of features from tiny images” April 2009.

[3] Y. Zhang, Y. Zhang, and X. Qian, “Deep neural networks for free-hand sketch recognition,” in Lecture Notes in Computer Science, Cham: Springer International Publishing, vol.9917, pp. 689–696, 2016.

[4] M. Swamynathan, “Conclusion,” in Mastering Machine Learning with Python in Six Steps, Berkeley, CA: Apress, pp. 443–448,2019.

[5] J. Zhang, F. Hu, L. Li, et al,” An adaptive mechanism to achieve learning rate dynamically,” Neural Comput. Appl., vol. 31, no. 10, pp. 6685–6698, 2019.

[6] Z. Zou, Z. Shi, Y. Guo, et al, “Object detection in 20 years: A survey,” arXiv [cs.CV], May 2019.

[7] H. Zhang and X. Hong, “Recent progresses on object detection: a brief review,” Multimed. Tools Appl., vol. 78, no. 19, pp. 27809–27847, 2019.

[8] K. Tong, Y. Wu, and F. Zhou, “Recent advances in small object detection based on deep learning: A review,” Image Vis. Comput., vol. 97, May 2020.

[9] P. Chaturvedi “Classification and stage prediction of lung cancer using convolutional neural networks,” Regular issue, vol. 8, no. 10, pp. 993–998, 2019.

[10] J. Zhang, F. Hu, L. Li, et al,” An adaptive mechanism to achieve learning rate dynamically,” Neural Comput. Appl., vol. 31, no. 10, pp. 6685–6698, 2019.

[11] H. A. Shiddieqy, F. I. Hariadi, and T. Adiono, “Implementation of deep-learning based image classification on single board computer,” in International Symposium on Electronics and Smart Devices (ISESD), 2017.

[12] A. Singh, P. Singh “Image Classification: A Survey,” Journal of Informatics Electrical and Electronics Engineering, vol. 1 no. 2, pp. 1-9, 2020.

[13] P. Singhal, P. Singh, and A. Vidyarthi, “Interpretation and localization of Thorax diseases using DCNN in Chest X-Ray,” Journal of Informatics Electrical and Electronics Engineering, vol. 1, no. 1, pp. 1–7, 2020.

[14] A. Singh and P. Singh “A Comprehensive Survey on Machine Learning”. Journal of Management and Service Science,” vol. 1 no. 1, pp. 1-17,2021.

[15] A. Singh and P. Singh” License Plate Recognition,” Journal of Management and Service Science, vol.1 no.2, pp. 1-14, 2021.

[16] S. Yadav, P. Singh “Social Distancing Detector,” Journal of Management and Service Science, vol. 1 no. 2, pp. 1-10, 2021.