Covid-19 Face Mask Detection Using TensorFlow, Keras and OpenCV

LOVELY PROFESSIONAL UNIVERSITY

Transforming Education Transforming India

NAME - ANKIT KUMAR

REGNO - 11806646

SECTION - KM086

ROLL - B76

GITHUB - https://github.com/iamankit555/int248-project

Abstract

Effective strategies to restrain COVID-19 pandemic need high attention to mitigate negatively impacted communal health and global economy, with the brim-full horizon yet to unfold. In the absence of effective antiviral and limited medical resources, many measures are recommended by WHO to control the infection rate and avoid exhausting the limited medical resources. Wearing a mask is among the non-pharmaceutical intervention measures that can be used to cut the primary source of SARS-CoV2 droplets expelled by an infected individual. Regardless of discourse on medical resources and diversities in masks, all countries are mandating coverings over the nose and mouth in public. To contribute towards communal health, this paper aims to devise a highly accurate and real-time technique that can efficiently detect non-mask faces in public and thus, enforcing to wear mask. The proposed technique is ensemble of one-stage and two-stage detectors to achieve low inference time and high accuracy. We start with ResNet50 as a baseline and applied the concept of transfer learning to fuse high-level semantic information in multiple feature maps. In addition, we also propose a bounding box transformation to improve localization performance during mask detection. The experiment is conducted with three popular baseline models viz. ResNet50, AlexNet and MobileNet. We explored the possibility of these models to plug-in with the proposed model so that highly accurate results can be achieved in less inference time. It is observed that the proposed technique achieves high accuracy (98.2%) when implemented with ResNet50. Besides, the proposed model generates 11.07% and 6.44% higher precision and recall in mask detection when compared to the recent public baseline model published as RetinaFaceMask detector. The outstanding performance of the proposed model is highly suitable for video surveillance devices.

INTRODUCTION

The trend of wearing face masks in public is rising due to the COVID- 19 corona virus epidemic all over the world. Before Covid-19, People used to wear masks to protect their health from air pollution. While other people are self-conscious about their looks, they hide their emotions from the public by hiding their faces. Scientists proofed that wearing face masks works on impeding COVID-19 transmission. COVID19 (known as corona virus) is the latest epidemic virus that hit the human health in the last century. In 2020, the rapid spreading of COVID-19 has forced the World Health Organization to declare COVID- 19 as a global pandemic.
More than five million cases were infected by COVID-19 in less than 6 months across 188 countries. The virus spreads through close contact and in crowded and overcrowded areas.

The corona virus epidemic has given rise to an extraordinary degree of worldwide scientific cooperation. Artificial Intelligence (AI) based on Machine learning and Deep Learning can help to fight Covid-19 in many ways. Machine learning allows researchers and clinicians evaluate vast

quantities of data to forecast the distribution of COVID-19, to serve as an early warning mechanism for potential pandemics, and to classify vulnerable populations.The provision of healthcare needs funding for emerging technology such as artificial intelligence, IoT, big data and machine learning to tackle and predict new diseases. In order to better understand infection rates and to trace and quickly detect infections, the AI's power is being exploited to address the Covid-19 pandemic. People are forced by laws to wear face masks in public in many countries. These rules and laws were developed as an action to the exponential growth in cases and deaths in many areas. However, the process of monitoring large groups of people is becoming more difficult. The monitoring process involves the detection of anyone who is not wearing a face mask.

Here we introduce a mask face detection model that is based on computer vision and deep learning. The proposed model can be integrated with surveillance cameras to impede the COVID-19 transmission by allowing the detection of people who are wearing masks not wearing face masks. The model is integration between deep learning and classical machine learning techniques with opencv, tensor flow and keras. We have used deep transfer leering for feature extractions and combined it with three classical machine learning algorithms. We introduced a comparison between them to find the most suitable algorithm that achieved the highest accuracy and consumed the least time in the process of training and detection.

1.1 MACHINE LEARNING

Machine learning (**ML**) is the study of computer algorithms that improve automatically through experience. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so.

Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop conventional algorithms to perform the needed tasks. Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

Machine learning approaches are traditionally divided into three broad categories, depending on the nature of the "signal" or "feedback" available to the learning system:

- Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.
- Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).
- Reinforcement learning: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against

an opponent). As it navigates its problem space, the program is provided feedback that's analogous to rewards, which it tries to maximize.

Other approaches have been developed which don't fit neatly into this three-fold categorization, and sometimes more than one is used by the same machine learning system.

1.2 COMPUTER VISION

Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to understand and automate tasks that the human visual system can do, Computer vision tasks include methods for acquiring, processing, analyzing and understanding digital images, and extraction of highdimensional data from the real world in order to produce numerical or symbolic information, e.g. in the forms of decisions, Understanding in this context means the transformation of visual images (the input of the retina) into descriptions of the world that make sense to thought processes and can elicit appropriate action. This image understanding can be seen as the disentangling of symbolic information from image data using models constructed with the aid of geometry, physics, statistics, and learning theory.

The scientific discipline of computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, multidimensional data from a 3D scanner or medical scanning device. The technological discipline of computer vision seeks to apply its theories and models to the construction of computer vision systems. Computer vision is an interdisciplinary field that deals with how computers can be made to gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do.

extraction, analysis and understanding of useful information from a single image or a sequence of images. It involves the development of a theoretical and algorithmic basis to achieve automatic visual understanding. As a scientific discipline, computer vision is concerned with the theory behind artificial systems that extract information from images.

The image data can take many forms, such as video sequences, views from multiple cameras, or multidimensional data from a medical scanner. As a technological discipline, computer vision seeks to apply its theories and models for the construction of computer vision systems.

1.3 DEEP LEARNING

Deep learning methods aim at learning feature hierarchies with features from higher levels of the hierarchy formed by the composition of lower level features. Automatically learning features at multiple levels of abstraction allow a system to learn complex functions mapping the input to the output directly from data, without depending completely on human-crafted features. Deep learning algorithms seek to exploit the unknown structure in the input distribution in order to discover good representations, often at multiple levels, with higher-level learned features defined in terms of lower-level features.

The hierarchy of concepts allows the computer to learn complicated concepts by building them out of simpler ones. If we draw a graph showing how these concepts are built on top of each other, the graph is deep, with many layers. For this reason, we call this approach to AI deep learning. Deep learning excels on problem domains where the inputs (and even output) are analog. Meaning, they are not a few quantities in a tabular format but instead are **images of pixel data, documents of text data or files of audio data.** Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction.

1.4 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-ofthe-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a

high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, Video Surf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow
Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a template interface that works seamlessly with STL containers.

1.5 TENSORFLOW

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google, TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units).

Tensor Flow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS. Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

The name Tensor Flow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as *tensors*. During the Google I/O Conference in June 2016, Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.Unlike other numerical libraries intended for use in Deep Learning like Theano, TensorFlow was designed for use both in research and development and in production systems, not least RankBrain in Google search and the fun DeepDream project.It can run on single CPU systems, GPUs as well as mobile devices and large scale distributed systems of hundreds of machines.

1.6 KERAS

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides. Keras contains numerous implementations of commonly used neuralnetwork building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel. Keras is a minimalist Python library for deep learning that can run on top of Theano or Tensor Flow. It was developed to make implementing deep learning models as fast and easy as possible for research and development. It runs on Python 2.7 or 3.5 and can seamlessly execute on GPUs and CPUs given the underlying frameworks. It is released under the permissive MIT license.

Keras was developed and maintained by François Chollet, a Google engineer using four guiding principles:

- **Modularity**: A model can be understood as a sequence or a graph alone. All the concerns of a deep learning model are discrete components that can be combined in arbitrary ways.
- **Minimalism**: The library provides just enough to achieve an outcome, no frills and maximizing readability.
- **Extensibility**: New components are intentionally easy to add and use within the framework, intended for researchers to trial and explore new ideas.
- **Python**: No separate model files with custom file formats. Everything is native Python. Keras is designed for minimalism and modularity allowing you to very quickly define deep learning models and run them on top of a Theano or TensorFlow backend.

1.7 PyTorch

PyTorch is an open source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Face book's AI Research lab (FAIR).

It is free and open-source software released under the Modified BSD license. Although the Python interface is more polished and the primary focus of development, PyTorch also has a C++ interface. Tensor computing (like NumPy) with strong acceleration via graphics processing units (GPU).

 Deep neural networks built on a tape-based automatic differentiation system

PyTorch defines a class called Tensor `torch.Tensor` () to store and operate on homogeneous multidimensional rectangular arrays of numbers. PyTorch Tensors are similar to NumPy Arrays, but can also be operated on a CUDA-capable Nvidia GPU. PyTorch supports various sub-types of Tensors.

DATASET

Two datasets have been used for experimenting the current method. Dataset 1 [16] consists of 1376 images in which 690 images with people wearing face masks and the rest 686 images with people who do not wear face masks. Fig. 1 mostly contains front face pose with single face in the frame and with same type of mask having white color only.

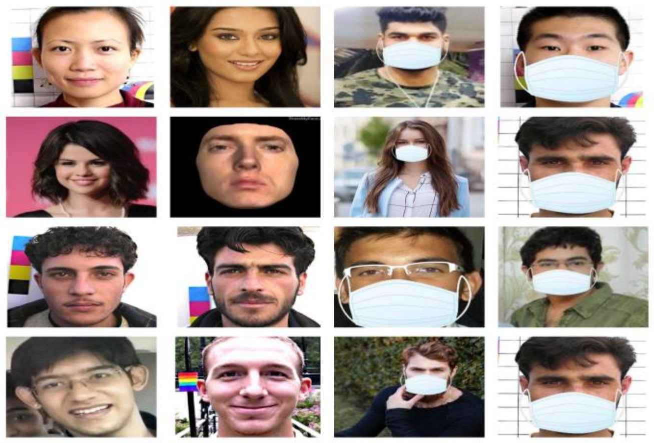


Fig. 1. Samples from Dataset 1 including faces without masks and with masks

Dataset 2 from Kaggle [17] consists of 853 images and its countenances are clarified either with a mask or without a mask. In fig. 2 some face collections are head turn, tilt and slant with multiple faces in the frame and different types of masks having different colors as well.

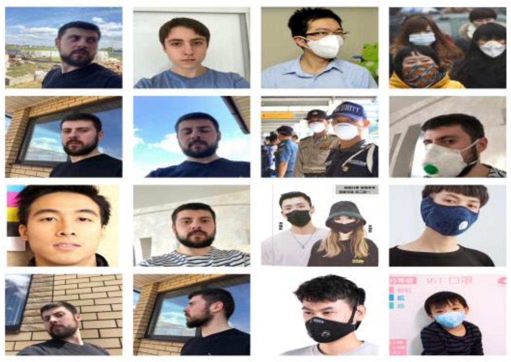


Fig. 2. Samples from Dataset 2 including faces without masks and with masks

Data Processing

Data preprocessing involves conversion of data from a given format to much more user friendly, desired and meaningful format. It can be in any form like tables, images, videos, graphs, etc. These organized information fit in with an information model or composition and captures relationship between different entities [6]. The proposed method deals with image and video data using Numpy and OpenCV.

a) Data Visualization:

Data visualization is the process of transforming abstract data to meaningful representations using knowledge communication and insight discovery through encodings. It is helpful to study a particular pattern in the dataset [7].

The total number of images in the dataset is visualized in both categories – 'with mask' and 'without mask'.

The statement categories=os.listdir(data path) categorizes the list of directories in the specified data path.

The variable categories now looks like: ['with mask', 'without mask'] Then to find the number of labels, we need to distinguish those categories using labels=[i for i in range(len(categories))].

It sets the labels as: [0, 1] Now, each category is mapped to its respective label using label dict=dict(zip(categories,labels))

which at first returns an iterator of tuples in the form of zip object where the items in each passed iterator is paired together consequently.

The mapped variable label dict looks like: {'with mask': 0, 'without mask': 1}

b) Conversion of RGB image to Gray image:

Modern descriptor-based image recognition systems regularly work on grayscale images, without elaborating the method used to convert from color-to-grayscale. This is because the colorto-grayscale method is of little consequence when using robust descriptors. Introducing nonessential information could increase the size of training data required to achieve good performance. As grayscale rationalizes the algorithm and diminishes the computational requisites, it is utilized for extracting descriptors instead of working on color images

We use the function cv2.cvtColor(input image, flag) for changing the color space. Here flag determines the type of conversion [9]. In this case, the flag cv2.COLOR BGR2GRAY is used for gray conversion.

Deep CNNs require a fixed-size input image. Therefore we need a fixed common size for all the images in the dataset. Using cv2.resize() the gray scale image is resized into 100 x 100.

c) Image Reshaping:

The input during relegation of an image is a three-dimensional tensor, where each channel has a prominent unique pixel. All the images must have identically tantamount size corresponding to 3D feature tensor. However, neither images are customarily coextensive nor their corresponding feature tensors [10].

Most CNNs can only accept fine-tuned images. This engenders several problems throughout data collection and implementation of model. However, reconfiguring the input images before augmenting them into the network can help to surmount this constraint. [11].

The images are normalized to converge the pixel range between 0 and 1.

Then they are converted to 4 dimensional arrays using data=np.reshape(data,(data.shape[0], img size,img size,1)) where 1 indicates the Grayscale image.

As, the final layer of the neural network has 2 outputs – with mask and without mask i.e. it has categorical representation, the data is converted to categorical labels.

B. Training of Model

a) Building the model using CNN architecture:
CNN has become ascendant in miscellaneous computer vision tasks [12]. The current method makes use of Sequential CNN. The First Convolution layer is followed by Rectified Linear Unit (ReLU) and MaxPooling layers. The Convolution layer learns from 200 filters. Kernel size is set to 3 x 3 which specifies the height and width of the 2D convolution window. As the model should be aware of the shape of the input expected, the first layer in the model needs to be provided with information about input shape. Following layers can perform instinctive shape reckoning [13]. In this case, input shape is specified as data.shape[1:] which returns the dimensions of the data array from index 1. Default padding is "valid" where the spatial dimensions are sanctioned to truncate and the input volume is non-zero padded. The activation parameter to the Conv2D class is set as "relu". It represents an approximately linear function that possesses all the assets of linear models that can easily be optimized with gradient-descent methods. Considering the performance and generalization in deep learning, it is better compared to other activation functions [14]. Max Pooling is used to reduce the spatial dimensions of the output volume. Pool size is set to 3 x 3 and the resulting output has a shape (number of rows or columns) of: shape of output = (input shape - pool size + 1) / strides), where strides has default value (1,1) [15].

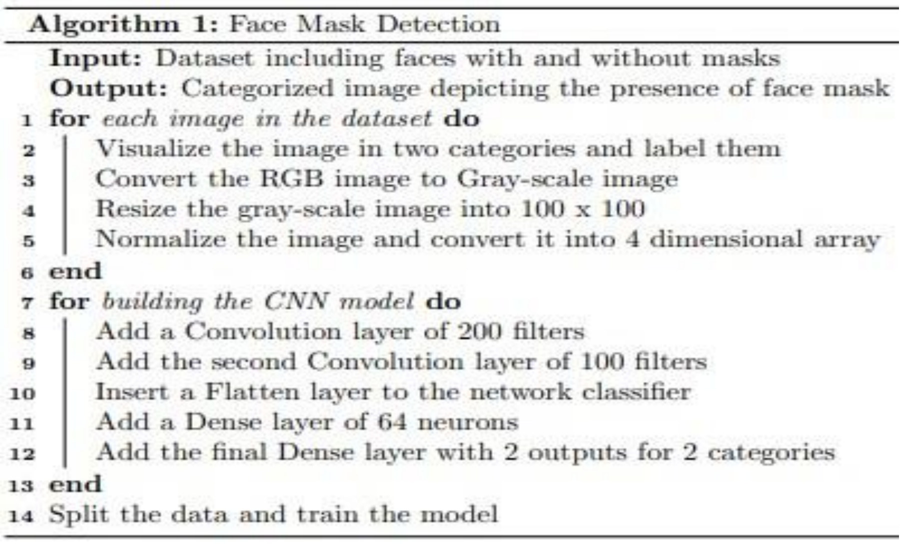
As shown in fig, 4, the second Convolution layer has 100 filters and Kernel size is set to 3 x 3. It is followed by ReLu and MaxPooling layers. To insert the data into CNN, the long vector of input is passed through a Flatten layer which transforms matrix of features into a vector that can be fed into a fully connected neural network classifier. To reduce overfitting a Dropout layer with a 50% chance of setting inputs to zero is added to the model. Then a Dense layer of 64 neurons with a ReLu activation function is added. The final layer (Dense) with two outputs for two categories uses the Softmax activation function.

Fig. 4. Convolutional Neural Network architecture The learning process needs to be configured first with the compile method [13]. Here "adam" optimizer is used. categorical crossentropy which is also known as multiclass log loss is used as a loss function (the objective that the model tries to minimize).
As the problem is a classification problem, metrics is set to "accuracy".

b) Splitting the data and training the CNN model: After setting the blueprint to analyze the data, the model needs to be trained using a specific dataset and then to be tested against a different dataset. A proper model and optimized train test split help to produce accurate results while making a prediction. The test size is set to 0.1 i.e. 90% data of the dataset undergoes training and the rest 10% goes for testing purposes. The validation loss is monitored using ModelCheckpoint. Next, the images in the training set and the test set are fitted to the Sequential model. Here, 20% of the training data is used as validation data. The model is trained for 20 epochs (iterations) which maintains a trade-

c) THE PROPOSED METHOD

d) The proposed method consists of a cascade classifier and a pre-trained CNN which contains two 2D convolution layers connected to layers of dense neurons. The algorithm for face mask detection is as follows:



```
Algorithm 1: Face Mask Detection
  Input: Dataset including faces with and without masks
  Output: Categorized image depicting the presence of face mask
  for each image in the dataset do
      Visualize the image in two categories and label them
      Convert the RGB image to Gray-scale image
      Resize the gray-scale image into 100 x 100
      Normalize the image and convert it into 4 dimensional array
  end
  for building the CNN model do
      Add a Convolution layer of 200 filters
      Add the second Convolution layer of 100 filters
      Insert a Flatten layer to the network classifier
      Add a Dense layer of 64 neurons
      Add the final Dense layer with 2 outputs for 2 categories
  end
  Split the data and train the model
```

e)

A. Data Processing

Data preprocessing involves conversion of data from a given format to much more user friendly, desired and meaningful format. It can be in any form like tables, images, videos, graphs, etc. These organized information fit in with an information model or composition and captures relationship between different entities [6]. The proposed method deals with image and video data using Numpy and OpenCV.

a) Data Visualization: Data visualization is the process of transforming abstract data to meaningful representations using knowledge communication and insight discovery through encodings. It is helpful to study a particular pattern in the dataset [7].

The total number of images in the dataset is visualized in both categories – ‘with mask’ and ‘without mask’.

The statement *categories=os.listdir(data path)* categorizes the list of directories in the specified data path. The variable *categories* now looks like: [‘with mask’, ‘without mask’]

Then to find the number of labels, we need to distinguish those categories using *labels=[i for i in range(len(categories))]*. It sets the labels as: [0, 1]

Now, each category is mapped to its respective label using *label dict=dict(zip(categories,labels))* which at first returns an iterator of tuples in the form of zip object where the items in each passed iterator is paired together consequently. The mapped variable *label dict* looks like: {'with mask': 0,

'without mask': 1}

b) Conversion of RGB image to Gray image: Modern descriptor-based image recognition systems regularly work on grayscale images, without elaborating the method used to convert from color-to-grayscale. This is because the colorto-grayscale method is of little consequence when using robust descriptors. Introducing nonessential information could increase the size of training data required to achieve good performance. As grayscale rationalizes the algorithm and diminishes the computational requisites, it is utilized for extracting descriptors instead of working on color images instantaneously [8].

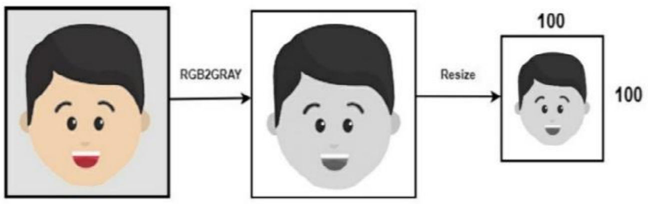


Fig. 3. Conversion of a RGB image to a Gray Scale image of

We use the function *cv2.cvtColor(input image, flag)* for changing the color space. Here flag determines the type of conversion [9]. In this case, the flag *cv2.COLOR BGR2GRAY* is used for gray conversion.

f) Deep CNNs require a fixed-size input image. Therefore we need a fixed common size for all the images in the dataset.

b) Image Reshaping: The input during relegation of an image is a three-dimensional tensor, where each channel has a prominent unique pixel. All the images must have identically tantamount size corresponding to 3D feature tensor. However, neither images are customarily coextensive nor their corresponding feature tensors [10]. Most CNNs can only accept fine-tuned images. This engenders several problems throughout data collection and implementation of model. However, reconfiguring the input images before augmenting them into the network can help to surmount this constraint.
[11].

The images are normalized to converge the pixel range between 0 and 1. Then they are converted to 4 dimensional arrays using *data=np.reshape(data,(data.shape[0], img _size,img size,1))*) where 1 indicates the Grayscale image. As, the final layer of the neural network has 2 outputs – with mask and without mask i.e. it has categorical representation, the data is converted to categorical labels.

B. Training of Model

a) Building the model using CNN architecture: CNN has become ascendant in miscellaneous computer vision tasks [12]. The current method makes use of Sequential CNN.

The First Convolution layer is followed by Rectified Linear Unit (ReLU) and MaxPooling layers. The Convolution layer learns from 200 filters. Kernel size is set to 3 x 3 which specifies the height and width of the 2D convolution window. As the model should be aware of the shape of the input expected, the first layer in the model needs to be provided with information about input shape. Following layers can perform instinctive shape reckoning [13]. In this case, *input shape* is specified as *data.shape[1:]* which returns the dimensions of the data array from index 1. Default padding is "valid" where the spatial dimensions are sanctioned to truncate and the input volume is non-zero padded. The activation parameter to the Conv2D class is set as "relu". It represents an approximately linear function that possesses all the assets of linear models that can easily be optimized with gradient-descent methods. Considering the performance and generalization in deep learning, it is better compared to other activation functions [14]. Max Pooling is used to reduce the spatial dimensions of the output volume. Pool size is set to 3 x 3 and the resulting output has a shape (number of rows or columns) of: shape of _output = (input _shape - pool _size + 1) / strides), where strides has default value (1,1) [15].

As shown in fig, 4, the second Convolution layer has 100 filters and Kernel size is set to 3 x 3. It is followed by ReLu and MaxPooling layers. To insert the data into CNN, the long vector of input is passed through a Flatten layer which transforms matrix of features into a vector that can be fed into a fully connected neural network classifier. To reduce overfitting a Dropout layer with a 50% chance of setting inputs to zero is added to the model. Then a Dense layer of 64 neurons with a ReLu activation function is added. The final layer (Dense) with two outputs for two categories uses the Softmax activation function.

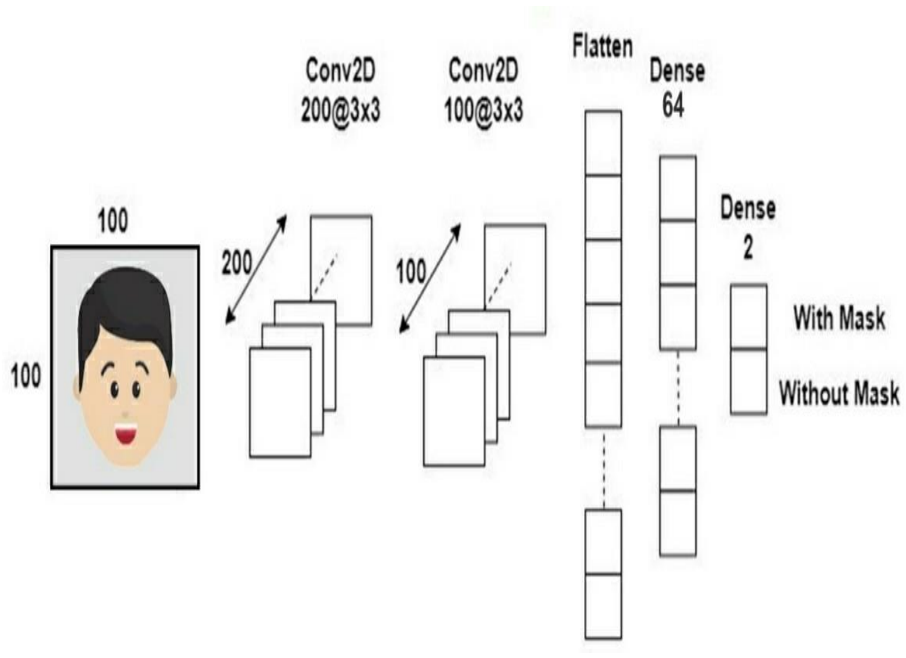
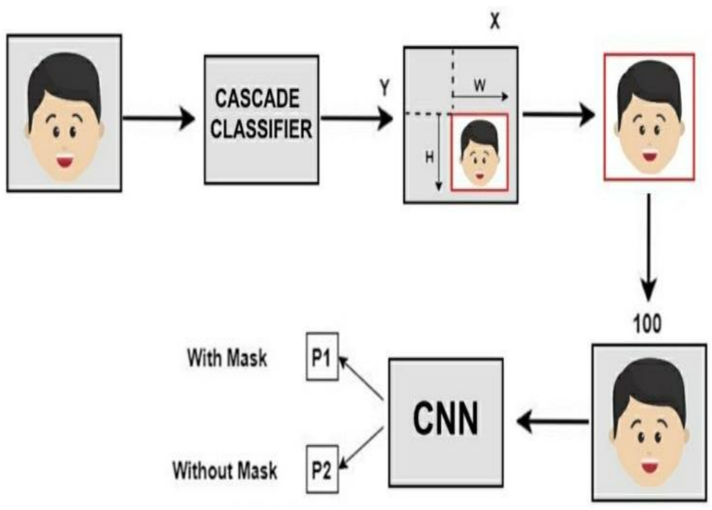


Fig. 4. Convolutional Neural Network architecture

The learning process needs to be configured first with the compile method [13]. Here "adam" optimizer is used. *categorical crossentropy* which is also known as multiclass log loss is used as a loss function (the objective that the model tries to minimize). As the problem is a classification problem, metrics is set to "accuracy".

b) Splitting the data and training the CNN model:

After setting the blueprint to analyze the data, the model needs to be trained using a specific dataset and then to be tested against a different dataset. A proper model and optimized *train test split* help to produce accurate results while making a prediction. The test size is set to 0.1 i.e. 90% data of the dataset undergoes training and the rest 10% goes for testing purposes. The validation loss is monitored using *ModelCheckpoint*. Next, the images in the training set and the test set are fitted to the Sequential model. Here, 20% of the training data is used as validation data. The model is trained for 20 epochs (iterations) which maintains a trade-off between accuracy and chances of overfitting. Fig. 5 depicts visual representation of the proposed model.



VI. RESULT AND ANALYSIS

The model is trained, validated and tested upon two datasets. Corresponding to dataset 1, the method attains accuracy up to 95.77% (shown in fig. 7). Fig. 6 depicts how this optimized accuracy mitigates the cost of error. Dataset 2 is more versatile than dataset 1 as it has multiple faces in the frame and different types of masks having different colors as well. Therefore, the model attains an accuracy of 94.58% on dataset 2 as shown in Fig. 9. Fig. 8 depicts the contrast between training and validation loss corresponding to dataset 2. One of the main reasons behind achieving this accuracy lies in *MaxPooling*. It provides rudimentary translation invariance to the internal representation along with the reduction in the number of parameters the model has to learn. This sample-based discretization process down-samples the input representation consisting of image, by reducing its dimensionality. Number of neurons has the optimized value of 64 which is not too high. A much higher number of neurons and filters can lead to worse performance. The optimized filter values and pool size help to filter out the main portion (face) of the image to detect the existence of mask correctly without causing over-fitting.

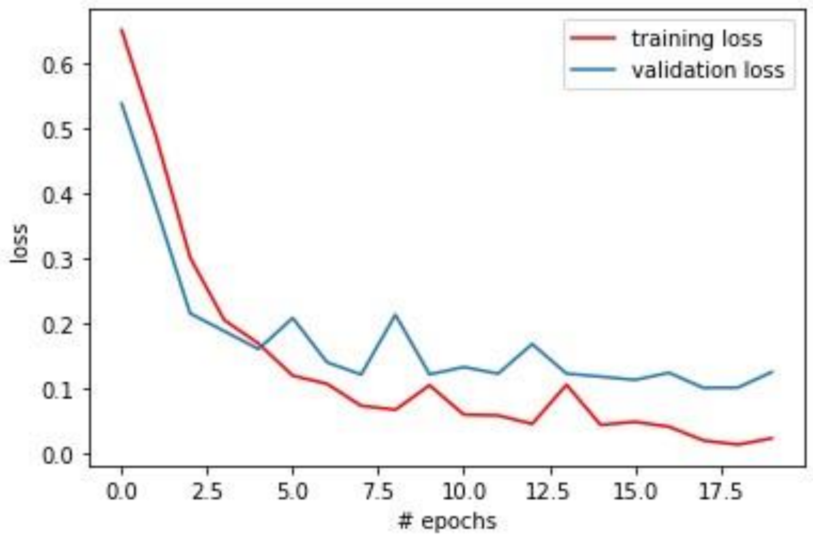


Fig. 6. # epochs vs loss corresponding to dataset 1

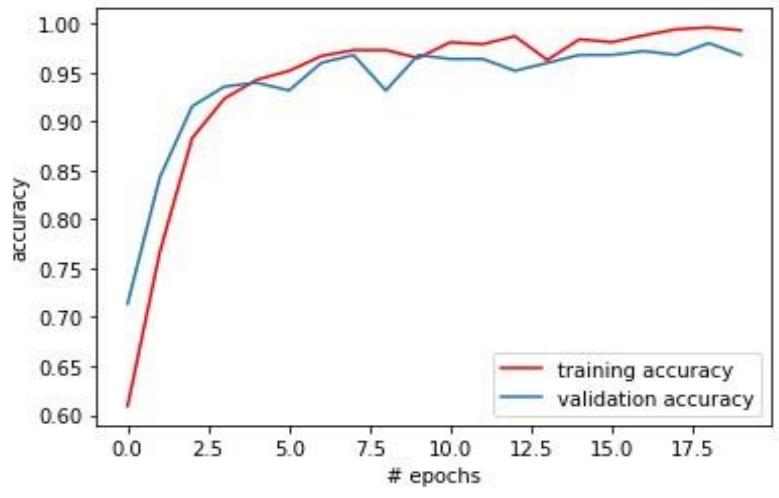


Fig. 7. # epochs vs accuracy corresponding to dataset 1

The system can efficiently detect partially occluded faces either with a mask or hair or hand. It considers the occlusion degree of four regions – nose, mouth, chin and eye to differentiate between annotated mask or face covered by hand. Therefore, a mask covering the face fully including nose and chin will only be treated as “with mask” by the model.

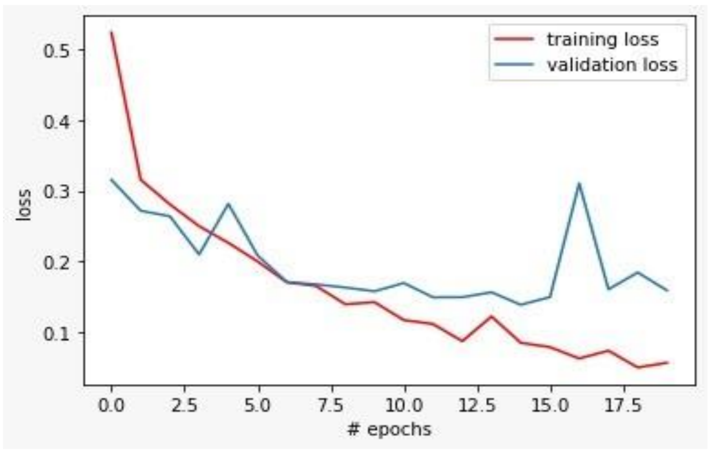


Fig. 8. # epochs vs loss corresponding to dataset 2

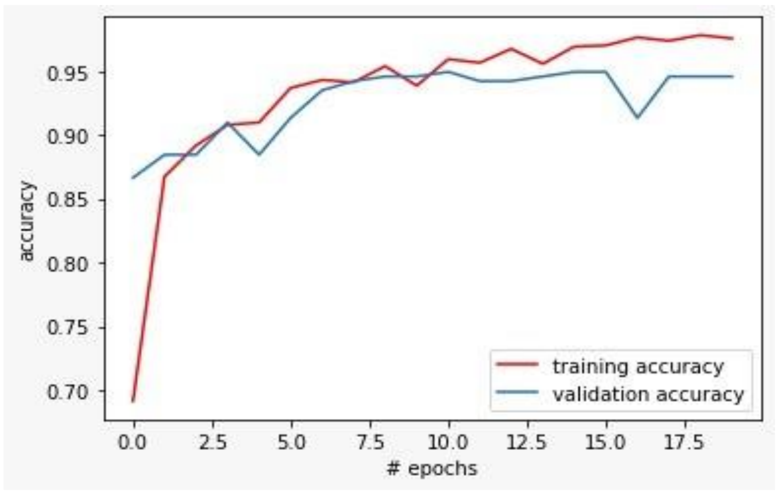


Fig. 9. # epochs vs accuracy corresponding to dataset 2

The main challenges faced by the method mainly comprise of varying angles and lack of clarity. Indistinct moving faces in the video stream make it more difficult. However, following the trajectories of several frames of the video helps to create a better decision – "with mask" or "without mask".

VII. CONCLUSIONS

In this paper, we briefly explained the motivation of the work at first. Then, we illustrated the learning and performance task of the model. Using basic ML tools and simplified techniques the method has achieved reasonably high accuracy. It can be used for a variety of applications.Wearing a mask may be obligatory in the near future, considering the Covid-19 crisis. Many public service providers will ask the customers to wear masks correctly to avail of their services. The deployed model will contribute immensely to the public health care system. In future it can be extended to detect if a person is wearing the mask properly or not. The model can be further improved to detect if the mask is virus prone or not i.e. the type of the mask is surgical, N95 or not.

REFERENCES

[1] W.H.O., "Coronavirus disease 2019 (covid-19): situation report, 205". 2020

[2] "Coronavirus Disease 2019 (COVID-19) – Symptoms", Centers for Disease Control and Prevention, 2020. [Online]. Available: https://www.cdc.gov/coronavirus/2019-ncov/symptomstesting/symptoms.html. 2020.

[3] “Coronavirus — Human Coronavirus Types — CDC”, Cdc.gov, 2020. [Online]. Available: https://www.cdc.gov/coronavirus/types.html. 2020. [4] W.H.O., “Advice on the use of masks in the context of COVID-19: interim guidance”, 2020.

[5] M. Jiang, X. Fan and H. Yan, “RetinaMask: A Face Mask detector”, arXiv.org, 2020. [Online]. Available: https://arxiv.org/abs/2005.03950. 2020.

[6] B. Suvarnamukhi and M. Seshashayee, “Big Data Concepts and Techniques in Data Processing”, International Journal of Computer Sciences and Engineering, vol. 6, no. 10, pp. 712-714, 2018. Available: 10.26438/ijcse/v6i10.712714.

[7] F. Hohman, M. Kahng, R. Pienta and D. H. Chau, “Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers,” in IEEE Transactions on Visualization and Computer Graphics, vol. 25, no. 8, pp. 2674-2693, 1 Aug. 2019, doi: 10.1109/TVCG.2018.2843369.

[8] C. Kanan and G. Cottrell, “Color-to-Grayscale: Does the Method Matter in Image Recognition?”, PLoS ONE, vol. 7, no. 1, p. e29740, 2012. Available: 10.1371/journal.pone.0029740.

[9] Opencv-python-tutroals.readthedocs.io. 2020. Changing Colorspaces — Opencv-Python Tutorials 1 Documentation. [online] Available at:https://opencv-python-tutroals.readthedocs.io/en/latest/py tutorials/ py imgproc/py colorspaces/py colorspaces.html. 2020.

[10] M. Hashemi, “Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation”, Journal of Big Data, vol. 6, no. 1, 2019. Available: 10.1186/s40537-019-0263-7 . 2020.

[11] S. Ghosh, N. Das and M. Nasipuri, “Reshaping inputs for convolutional neural network: Some common and uncommon methods”, Pattern Recognition, vol. 93, pp. 79-94, 2019. Available: 10.1016/j.patcog.2019.04.009.

[12] R. Yamashita, M. Nishio, R. Do and K. Togashi, “Convolutional neural networks: an overview and application in radiology”, Insights into Imaging, vol. 9, no. 4, pp. 611-629, 2018. Available: 10.1007/s13244018-0639-9.

[13] “Guide to the Sequential model - Keras Documentation”, Faroit.com, 2020. [Online]. Available: https://faroit.com/keras-docs/1.0.1/gettingstarted/sequential-model-guide/. 2020.

[14] Nwankpa, C., Ijomah, W., Gachagan, A. and Marshall, S., 2020. Activation Functions: Comparison Of Trends In Practice And Research For Deep Learning. [online] arXiv.org. Available at: https://arxiv.org/abs/1811.03378. 2020.

[15] K. Team, “Keras documentation: MaxPooling2D layer”, Keras.io, 2020. [Online]. Available: https://keras.io/api/layers/pooling layers/ max _pooling2d/. 2020.

[16] “prajnasb/observations”, GitHub, 2020. [Online]. Available: https://github.com/prajnasb/observations/tree/master/experiements/data. 2020.

[17] “Face Mask Detection”, Kaggle.com, 2020. [Online]. Available: https://www.kaggle.com/andrewmvd/face-mask-detection. 2020.

[18] “TensorFlow White Papers”, TensorFlow, 2020. [Online]. Available: https://www.tensorflow.org/about/bib. 2020.

[19] K. Team, “Keras documentation: About Keras”, Keras.io, 2020. [Online]. Available: https://keras.io/about. 2020.

[20] “OpenCV”, Opencv.org, 2020. [Online]. Available: https://opencv.org/. 2020.

[21] D. Meena and R. Sharan, “An approach to face detection and recognition,” 2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE), Jaipur, 2016, pp. 1-6, doi: 10.1109/ICRAIE.2016.7939462.

[22] S. Ge, J. Li, Q. Ye and Z. Luo, “Detecting Masked Faces in the Wild with LLE-CNNs,” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 426-434, doi:
10.1109/CVPR.2017.53.