iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** VI **Month of publication:** June 2021

DOI: https://doi.org/10.22214/ijraset.2021.35511

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VI Jun 2021- Available at www.ijraset.com

Chatbot with Facemask Detection Technique

Lavanya Kesa[1], Shreya Singh Takur[2], Kirti Pasupuleti[3], Mr. T. Sandeep[4]
[1, 2, 3]*U.G.Student,* [4]*Assistant Professor Dept. of ECE, Sreenidhi Institute of Science and Technology, Hyderabad, Telangana-501301 India*

Abstract: ***Artificial intelligence is now in full swing, and chatbots are just a blip on the radar of a massive wave of advancement. Virtual chatbots that simulate human discussions for performing various tasks are becoming increasingly popular as messengers become more widely used. Some chatbots can schedule medical appointments, order a taxi, transfer money to pals, check into a flight, and perform a variety of other functions. Chatbots assist consumers as well, and they are becoming increasingly interested in this technology. Given the current COVID-19 situation, it's critical to be informed about the growing number of cases and the many precautions that must be followed. This information will be obtained with the assistance of the chatbot. Our chatbot not only converses with users, but it also informs them about Covid-19 and the measures that must be taken. We also understand the importance of wearing masks to prevent the infection from spreading. Using image processing, we will determine if people are wearing their masks or not in this section of our research. To determine if people are wearing a face mask or not, we will develop this section of the project using a simple and basic Convolutional Neural Network (CNN) model, TensorFlow with Keras library, and OpenCV. In this project, we will use a dataset that contains two types of images: images with masks and images without masks. Then, using the webcam on your PC, we'll utilize these images to create a CNN model using TensorFlow to detect if you're wearing a face mask.***

Keywords: ***Natural language processing, Face mask, Deep learning, Convolution neural networks.***

I. INTRODUCTION

In place of direct communication with a live human agent, a chatbot is a software application that conducts an online chat conversation using text or text-to-speech. Designed to closely resemble how a human would interact with a conversational partner, Chatbot systems are notoriously difficult to tune and test, and many in production are still unable to talk effectively or pass the industry-standard Turing test. Chatbots are used for a variety of reasons, including customer support, request routing, and information collection. While some chatbots make heavy use of word classification, natural language processing, and advanced AI, others just scan for basic keywords and generate responses using common phrases from a library or database. The majority of chatbots are accessed via website popups or virtual assistants. Commerce (e-commerce via chat), education, entertainment, finance, health, news, and productivity are some of the sectors in which they can be classed. Many businesses use messaging applications or SMS to communicate with chatbots. The bots often appear as one of the user's contacts, but they can also take part in group chats. Chatbots have been employed by a several banks, insurers, media businesses, e-commerce companies, airlines, hotel chains, merchants, health care providers, government bodies, and restaurant chains to answer simple questions and enhance client engagement, for marketing purposes and to provide alternative ordering options. According to a 2017 research, only 4% of businesses use chatbots. According to a 2016 research, 80% of companies said they planned to have one by 2020.COVID-19 is a disease that spreads from person to person and can be prevented by wearing a face mask properly. COVID-19 can be kept at bay if persons maintain tight social distance and use a face mask. Unfortunately, people are not following the guidelines, which is hastening the spread of the infection. Detecting persons who aren't following the guidelines and contacting the appropriate authorities can help to stop the spread of the coronavirus. Face mask detection is a technique for determining whether or not someone is wearing a mask. Object detection systems have been introduced in a variety of ways. In medical applications, deep learning algorithms are widely used. Deep learning architectures have recently demonstrated a significant significance in object detection. These architectures can be used to detect a mask on a person's face. This study aims to create a system that can detect whether a person is wearing a mask and alert the appropriate authorities in a smart city network. To begin, CCTV cameras are utilized to gather real-time video footage of various public locations throughout the city. Facial images are taken from the camera clip, and these images are used to identify the mask on the face. The learning technique Convolutional Neural Network (CNN) is used to extract features from images, and subsequently, numerous hidden layers learn these features. When the architecture detects people without a face mask, the information is sent to the appropriate authority via the city network, who then takes the appropriate action. The proposed system evaluated promising output based on data gathered from various sources. In this pandemic situation, we also represented a system that can ensure proper law enforcement on people who are not following basic health guidelines.

II. LITERATURE SURVEY

In 1950, Alan Turing's famous article "Computing Machinery and Intelligence" was published, which proposed what is now called the Turing test as a criterion of intelligence. This criterion depends on a computer program's ability to impersonate a human in a real-time written conversation with a human judge to the extent that the judge is unable to distinguish reliably—based on the conversational content alone—between the program and a real human. The notoriety of Turing's proposed test stimulated great interest in Joseph Weizenbaum's program ELIZA, published in 1966, which seemed to be able to fool users into believing that they were conversing with a real human. However Weizenbaum himself did not claim that ELIZA was genuinely intelligent, and the introduction to his paper presented it more as a debunking exercise: Artificial intelligence devices are programmed to act in amazing ways, frequently enough to astonish even the most seasoned observer. But once a program's secrets are revealed, once its inner workings are revealed, its allure fades; It is revealed to be nothing more than a set of procedures... "I could have written it," the spectator thinks to himself. With that idea, he moves the application in question from the "clever" shelf to the "curious" shelf. The purpose of this paper is to prompt such a reevaluation of the program that will be "described." Few applications have ever required it as much as this one.

The recognition of hint words or phrases in the input is ELIZA's key way of functioning (which has been adopted by chatbot designers ever since), and the output of the matching pre-prepared or pre-programmed responses that can appear to advance the conversation in a meaningful way (for example, by responding to any input that contains the term 'MOTHER' with 'TELL ME MORE ABOUT YOUR FAMILY'). As a result, even though the processing involved was very superficial, an illusion of understanding is created. Because human judges are so willing to give the benefit of the doubt when conversational responses might be regarded as "intelligent," ELIZA demonstrated that such an illusion is surprisingly easy to create. Humans' willingness to accept computer output as really conversational—even when it is actually based on relatively simple pattern-matching—can be leveraged for productive reasons, according to interface designers. The majority of individuals prefer to interact with human-like systems, As a result, chatbot-style tactics may have a place in interactive systems that require to elicit information from users, as long as the content is simple and falls into predictable categories. Online help systems, for example, can benefit from chatbot techniques to identify the area of assistance that consumers require, A "friendlier" interface than a more formal search or menu system is possible. This type of application has the potential to move chatbot technology from Weizenbaum's "shelf... kept for curiosities" to "really valuable computational procedures.

Jiang et al. offered the Retina Face Mask face mask detection model in combination with a cross-class object removal technique. The created model incorporates a single stage detector that uses a feature pyramid network to provide somewhat better precision and recall than the baseline result. Deep learning is a technology that can be used to analyze large amounts of data and has applications in computer vision, pattern recognition, and speech recognition, among other things. The work of Liu et al. focuses on various widely used deep learning architectures and their applications. The autoencoder, convolutional neural networks, Boltzmann machines, and deep belief networks are all detailed networks. To process unlabeled data, deep learning can be employed in unsupervised learning methods. Li et al. introduced a CNN model for fast face detection that analyses low resolution input images, discards non-facial parts, and accurately processes the regions with higher resolution for exact detection. This project distinguishes between using a face mask and not using one. Matthias et al. have worked on a face mask detection project that focuses on capturing real-time photos that indicate whether or not a person is wearing a mask. The dataset was utilized for both training and implementing the decision-making algorithm to recognize the primary face features (eyes, mouth, and nose). Rigid masks performed better, however inaccurate detections can occur owing to lighting and items seen outside of the face.

III. METHODOLOGY

If you're in charge of customer support and service, the promises of today's chatbots are almost too good to be true. Advanced cognitive technologies, native machine learning, natural language processing, and generation are all compelling features. Such technology can process, analyze, and reply to inputs in massive data volumes, simulating human communication.

The basic criterion for chatbot success is the same as it is for any other technology installation project: ROI. Top explained that "cost savings are key to the initial business plan that justifies intelligent help and chatbots." "However, chatbots are demonstrating their worth by making consumers happy, increasing the productivity of real agents or technicians, and helping to raise the topline."

The following code can be used to implement a general purpose Chat bot:

To install Chatter Bot in Python, type the following command in the terminal or at the command prompt.

pip install chatterbot

A. Trainer for Chatbot

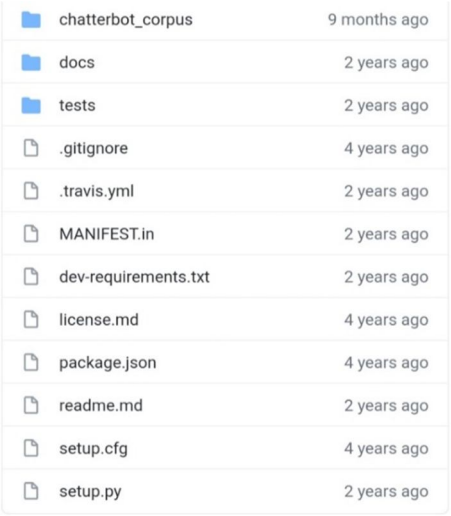


Fig 1.Modules to train chatbot

These modules allow Chatter Bot to be easily trained to respond to a variety of inputs in a variety of languages. Despite the fact that much of Chatter Bot is designed to be language agnostic, these training sets are nevertheless valuable for priming a new database and expanding the range of responses a bot may produce.

1) *Training:* Chatter Bot comes with features that make the process of training a chat bot instance much easier. The training method for Chatter Bot entails loading sample conversations into the chat bot's database. The graph data structure that represents the sets of known assertions and answers is either created or built upon. When a conversation bot trainer receives a data set, it constructs the necessary entries in the chat bot's knowledge graph to accurately reflect the statement inputs and responses.

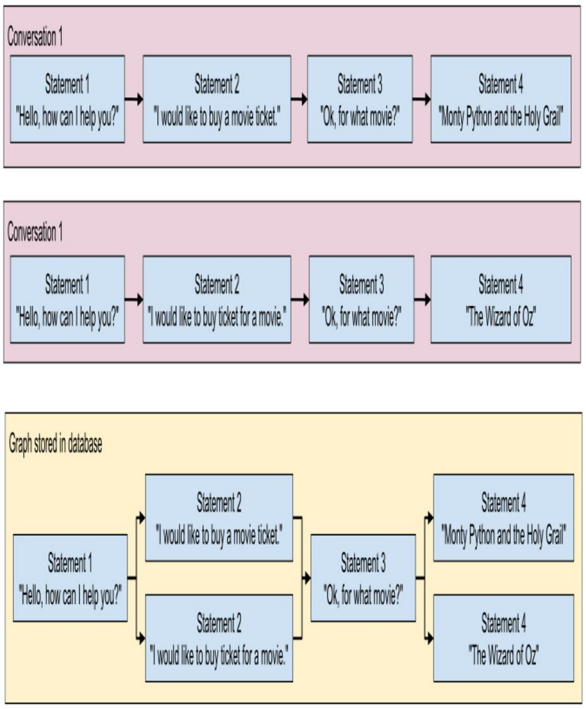


Fig 2.Training of a chatterbot

B. Training of a Chatterbot

Chatter Bot includes a number of training classes. These tools include everything from tools that let you update the chat bot's database knowledge graph based on a list of statements representing a discussion to tools that let you train your bot using a corpus of pre-loaded training data. You can also design your own training programme. If you want to train your bot using data in a format that isn't supported by one of the pre-built classes listed below, this is the way to go.

1) Setting the Training Class: Chatter Bot comes with built-in training classes, but you may also construct your own if necessary. You call train () on an instance that has been initialised with your chat bot to utilise a training class.

C. Training Classes

1) Training via list data

chatterbot.trainers.ListTrainer(*chatbot, **kwargs*)

Allows you to teach a chat bot with a list of strings that resembles a discussion. You will be required to submit a list of statements for the training process, with the order of each statement determined by its placement in a given conversation. For instance, if you were to make one of the following training calls with a bot, The resulting chatterbot would then say "Hello!" in response to both "Hi there!" and "Greetings!" messages.

chatbot.py:

```
chatbot =ChatBot('Training Example')
```

Train.py:

```
fromchatbotimportchatbot
fromchatterbot.trainersimportListTrainer
trainer=ListTrainer(chatbot)
trainer.train([
"Hi there!",
"Hello",
])
trainer.train([
"Greetings!",
"Hello",
])
```

2) Training with corpus data

chatterbot.trainers.ChatterBotCorpusTrainer(*chatbot, **kwargs*)

Allows the Chatter Bot dialogue corpus to be used to train the chat bot. Chatter Bot includes a corpus data and utility module that makes it simple to train your bot to communicate quickly. Simply provide the corpus data components you want to employ to accomplish this.

chatbot.py

```
chatbot =ChatBot('Training Example')
```

train.py

```
fromchatbotimport chatbot
fromchatterbot.trainersimportChatterBotCorpusTrainer

trainer =ChatterBotCorpusTrainer(chatbot)

  trainer.train(
    "chatterbot.corpus.english"
```

3) *Specifying Corpus Scope:* Individual subsets of Chatter Bot's corpus can also be imported at once. If you only want to train with the English greetings and conversations corpora, for example, you may just specify them.

train.py

```
trainer.train(
"chatterbot.corpus.english.greetings",
"chatterbot.corpus.english.conversations"
)
```

When invoking the train method, you can additionally give file paths to corpus files or folders of corpus files.

4) *Training with the Ubuntu dialog corpus*

chatterbot.trainers.UbuntuCorpusTrainer*(chatbot, **kwargs)***[source]**

Allow chatbots to be taught using the Ubuntu Dialog Corpus data. This training class allows you to use the Ubuntu dialogue corpus to train your chat bot. Because of the Ubuntu dialogue corpus's large file size, It's possible that the download and training processes will take a long time. The method of downloading and extracting the compressed corpus file will be covered in this training session. If you've already downloaded the file, It is not going to be downloaded again. The file will not be extracted again if it has previously been extracted.

5) *Creating a New Training Class:* You can use your own data files to train your chat bot by creating a new trainer. If you wish to train your chat bot from a data source that isn't directly supported by Chatter Bot, you can do so this way. Your custom trainer should inherit *chatterbot .trainers. Trainer* class. Your trainer will need to have a method named *train*, that can take any parameters you choose. The chatter bot. trainers. Trainer class should be inherited by your custom trainer. Your trainer will require a train method that accepts any parameters you specify.

6) *Install your Training corpus Data:* As part of the Quick Start Guide, you must instal chatterbot. Please navigate to the (Virtual Env)/lib/pythonX.X/site-packages/chatterbot corpus/data/ directory once the installation is complete. This is the same structure as this GitHub repo, and this is where you may add your own directories and discussion files. You can then update the Django setting.py file and locate the chatterbot training section once you've finished with your files. Chatterbot.corpus must be added here.

```
  FILENAME>.DIRECTORY>
'training_data': [
      'chatterbot.corpus.english.greeting',
'chatterbot.corpus.custom.myown',
      'chatterbot.corpus.swedish.food'
  ]
```

D. Unit Testing

“A true professional does not waste the time and money of other people by handing over software that is not reasonably free of obvious bugs; that has not undergone minimal unit testing; that does not meet the specifications and requirements; that is gold-plated with unnecessary features; or that looks like junk.” – Daniel Read

```
pip install -r dev-requirements.txt
```

We proposed an automated smart framework for screening persons who are not using a face mask in this paper. In the smart city, all public places are monitored by CCTV cameras. The cameras are used to capture images from public places; then these images are feed into a system that identifies if any person without face mask appears in the image. If any person without a face mask is detected then this information is sent to the proper authority to take necessary actions.

We implemented this project in jupyter notebook. First we need to install the necessary packages.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VI Jun 2021- Available at www.ijraset.com

E. Incorporated Packages

1) *TensorFlow:* TensorFlow, a programming interface for expressing machine learning algorithms, is used to put machine learning systems into production in a variety of fields, including sentiment analysis, voice recognition, geographic information extraction, computer vision, text summarization, information retrieval, computational drug discovery and flaw detection to pursue research. TensorFlow is used at the backend of the proposed model's Sequential CNN architecture (which consists of numerous layers). It's also used in data processing to restructure data (images).
2) *Keras:* Keras provides fundamental reflections and building units for the design and transfer of machine learning arrangements at high iteration rates. TensorFlow's scalability and cross-platform features are fully utilised. Keras' primary data structures are layers and models.Keras is utilised to implement all of the layers in the CNN model. It aids in the compilation of the overall model, as well as the conversion of the class vector to the binary class matrix in data processing.
3) *OpenCV:* OpenCV (Open Source Computer Vision Library), an open-source computer vision and machine learning software library, is used to detect faces, group movements in recordings, trace progressive modules, follow eye gestures, and track camera operations, Remove red eyes from flash photos, identify comparable photos in an image database, sense the landscape and place marks to overlay it with more realism, and so on . In order to resize and colour convert data images, the proposed technique makes use of OpenCV's characteristics.
4) *Data Processing:* Data preparation entails converting data from one format to another that is more user-friendly, desirable, and meaningful. It can take any shape, including tables, photographs, movies, graphs, and so on. These ordered data are part of an information model or composition that represents relationships between various entities. Numpy and OpenCV are used in the suggested method to deal with image and video data.

a) *Data Visualization:* The technique of translating abstract data into meaningful representations via knowledge exchange and insight finding through encodings is known as data Visualization is a term that refers to the process of Investigating a certain trend in the dataset is beneficial.

Both the ‘with mask' and ‘without mask' categories are used to illustrate the total number of photos in the collection.
The category=os.listdir(data path) command categorises the list of directories in the supplied data path. [‘with mask', ‘without mask'] are the new variable categories.Then, in order to get the number of labels, we must use labels=[i for I in range(len(categories)) to identify those categories. The labels are set to [0, 1].
Now, each category is mapped to its corresponding label with the help of label dict=dict(zip(categories,labels)), which produces an iterator of tuples in the form of a zip object, with the elements in each passed iterator coupled together.{‘with mask': 0, ‘without mask': 1' } is the mapped variable label dict.

b) *Conversion of RGB Image to Gray Image:* Modern descriptor-based image recognition systems frequently work with grayscale images, without expanding on the color-to-grayscale conversion mechanism. This is because when utilizing robust descriptors, the color-to-grayscale approach has little impact. Incorporating non-essential data could increase the amount of training data needed to attain effective results. Grayscale is used for extracting descriptors rather than working on color photos in real time because it rationalizes the approach and reduces computational requirements.

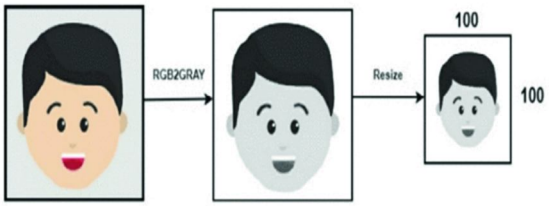


Fig 3.Conversion of a RGB image to a Gray Scale image of 100x100 size

To change the colour space, we utilise the function cv2.cvtColor(input image, flag). The type of conversion is determined by the flag . The flag cv2.COLOR BGR2GRAY is utilised for grey conversion in this situation.
A fixed-size input image is required for deep CNNs.As a result, all of the photos in the collection must have the same size. The grayscale image is enlarged to 100 × 100 pixels using cv2.resize().

c) *Image Reshaping:* A three-dimensional tensor is used as the input during image relegation, with each channel having a prominent unique pixel. All of the photos must be the same size and belong to the 3D feature tensor. However, neither pictures nor their related feature tensors are usually coextensive. Most CNNs can only accept photos that have been fine-tuned. This causes a slew of issues during data gathering and model implementation. This limit can be overcome by rearranging the input images before supplementing them into the network. The photos have been adjusted to bring the pixel range between 0 and 1 closer together. Then, using data=np.reshape(data,(data.shape[0], imgsize,img size,1), they are converted to four-dimensional arrays, with 1 indicating the Grayscale picture. The data is converted to categorical labels since the last layer of the neural network has two outputs – with mask and without mask, i.e. it has categorical .

F. Training of Model

1) Building the model using CNN architecture

In a variety of computer vision tasks, CNN has risen to the top . Sequential CNN is used in the current technique.

The Rectified Linear Unit (ReLU) and MaxPooling levels come after the First Convolution layer. The Convolution layer has 200 filters to learn from. The height and width of the 2D convolution window are specified by the kernel size, which is set to 3 x 3. Because the model must be aware of the shape of the expected input, the initial layer of the model must be provided with input shape information. Following layers can do shape reckoning instinctively . In this scenario, input shape is set to data.shape[1:], which returns the data array's dimensions from index.1. When the input volume is non-zero padded and the spatial dimensions are sanctioned to truncate, the default padding is “valid.” The Conv2D class's activation parameter is set to "relu."It depicts a nearly linear function with all of the advantages of linear models and the ability to be easily optimised using gradient-descent methods. It outperforms other activation functions in terms of performance and generalisation in deep learning. Max Pooling is used to lower the output volume's spatial dimensions. The final output has the form (number of rows or columns) of: shape of output = (input shape – pool size + 1) / strides, where strides is set to the default value (1,1) .

The second Convolution layer contains 100 filters and a Kernel size of 3 × 3 as illustrated in fig. The ReLu and MaxPooling layers come after it. The long vector of input is routed via a Flatten layer, which converts the matrix of features into a vector that can be fed into a fully connected neural network classifier, to insert the data into CNN. A Dropout layer with a 50% chance of setting inputs to zero is added to the model to reduce overfitting. Then a 64-neuron dense layer with a ReLu activation function is introduced. The Softmax activation function is used in the final layer (Dense), which has two outputs for two categories.

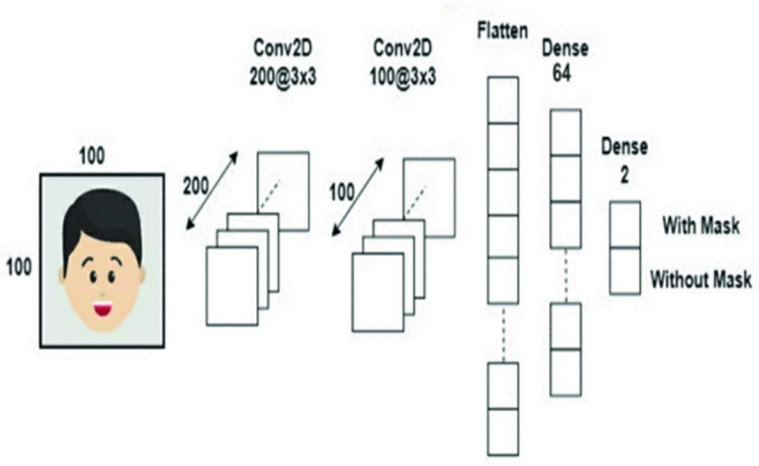


Fig 4.Second convolution layer

The compile method must be used to set up the learning process first. The “adam” optimizer is utilized in this case. As a loss function, categorical cross entropy, commonly known as multiclass log loss, is used (the objective that the model tries to minimize). Because this is a classification task, the metrics are set to "accuracy."

2) *Splitting the Data and Training the CNN Model:* The model must be trained using a certain dataset and then evaluated against a new dataset after the blueprint for data analysis has been established. When constructing a forecast, a good model and an optimized train test split can help you get accurate results. The test size is set to 0.1, which means that 90% of the data in the dataset is trained and 10% is used for testing. ModelCheckpoint is used to track validation loss. The images from both the training and test sets are then fitted to the Sequential model. Validation data is used to replace 20% of the training data. The model is trained for 20 epochs (iterations) to maintain a balance of accuracy and overfitting risk. The visual depiction of the proposed model is shown in the diagram below.

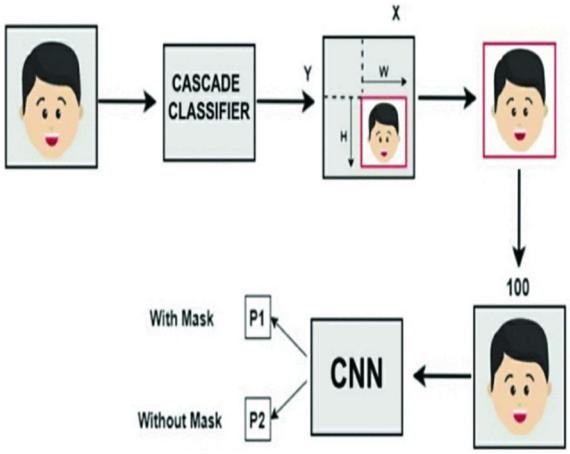


Fig 5.Depiction of classifying model

IV. PROPOSED ALGORITHM

A. User Login.
B. Interaction with bot.
C. Get information regarding COVID 19 updates and precautions (if necessary).
D. Detect the Faces through Camera.
E. Detection of possible faces in a frame.
F. Identification of faces without masks.
G. Marking mask less faces with bordered lines.
H. Capturing and sending this information to required authorities.

V. CONCLUSION & FUTURE SCOPE

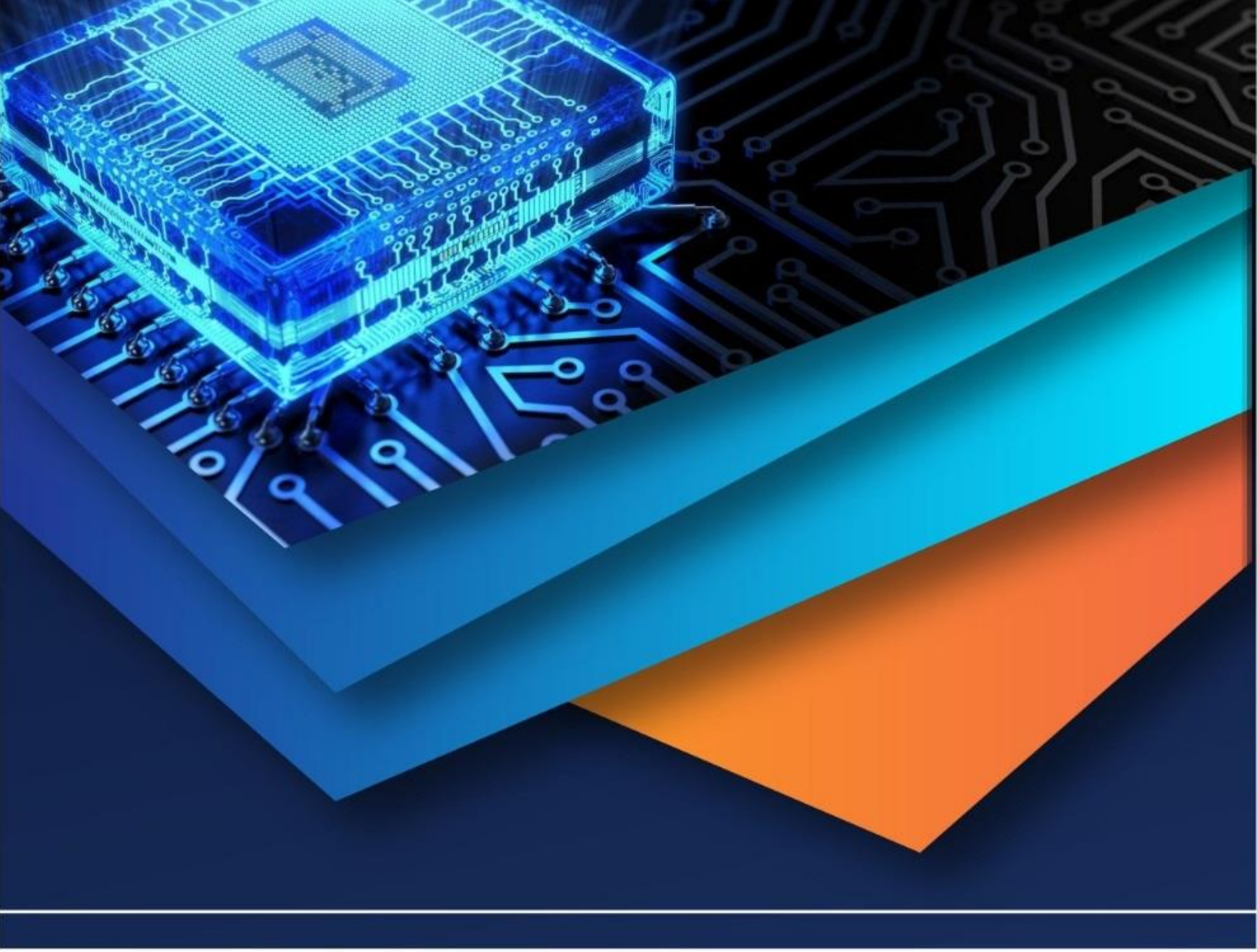
The majority of countries have made wearing a face mask mandatory due to an epidemic of COVID-19. In busy places, manual supervision of the face mask is necessary. As a result, researchers have been inspired to automate the face mask detecting method. Future work will include the integration of chatbot and face mask detection into an application, in which the camera detects whether or not a person is wearing a face mask while also measuring the distance between each person, check and detect whether or not a person is wearing a face mask as additional protection in COVID-19 prevention, and create an alarm if these rules are not followed properly. When numerous CNN models are combined and that each model be evaluated with the highest performance accuracy during training to improve performance in detecting and recognizing people wearing face masks and face shields.

VI. ACKNOWLEDGEMENT

Firstly, we are grateful to Sreenidhi Institute of Science and Technology for giving us the opportunity to work on this project. We are fortunate to have worked under the supervision of our internal guide Mr. T. Sandeep, guide Dr. J. Chattopadhyay. His guidance and ideas have made this project work. We are thankful to Mr. Sateesh Kumar for being the in charge for this project and conduction reviews. We are also thankful to the HoD of Electronics and Communication Engineering, Dr. S.P.V. Subba Rao for giving us access to all the resources that went into building this project.

REFERENCES

[1] M. Z. Islam, M. M. Islam, and A. Asraf, "A Combined Deep CNNLSTM Network for the Detection of Novel Coronavirus (COVID-19) Using X-ray Images," *Informatics in Medicine Unlocked*, vol. 20, pp. 100412, Aug. 2020.
[2] R. Jaiswal, A. Agarwal, and R. NEGI, "Smart Solution for Reducing the COVID-19 Risk using Smart City Technology," *IET Smart Cities*, vol. 2, pp. 82–88, 2020.
[3] Z. Wang, G. Wang, B. Huang, Z. Xiong, Q. Hong, H. Wu, P. Yi, K. Jiang, N. Wang, Y. Pei, et al., Masked face recognition dataset and application, arXiv preprintarXiv:2003.09093 (2020)
[4] Fadhil, A., 2018."Beyond patient monitoring: Conversational agents role in telemedicine & healthcare support for home-living elderly individuals". arXiv preprint arXiv:1803.06000.
[5] Raij, A.B., Johnsen, K., Dickerson, R.F., Lok, B.C., Cohen, M.S., Duerson, M., Pauly, R.R., Stevens, A.O., Wagner, P. and Lind, D.S., 2007. Comparing interpersonal interactions with a virtual human to those with a real human. IEEE transactions on visualization and computer graphics, 13(3),pp.443-457.
[6] H. Li, Z. Lin, X. Shen, J. Brandt, G. Hua, in Proceedings of the IEEE conference on computer vision and pattern recognition (2015), pp. 5325-5334
[7] S.S. Farfade, M.J. Saberian, L.J. Li, in Proceedings of the 5th ACM on International Conference on Multimedia Retrieval (2015), pp. 643-650



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY