Mask Detection System Using Convolutional Neural Network

Zubair Iqbal
CS&E Deptt MIT, Moradabad
zubairiqbal17@gmail.com

Prachi Gupta
CS&E Deptt MIT, Moradabad
prachi.g19@gmail.com

Satendra Kumar
CS&E Deptt MIT, Moradabad
satendra04cs41@gmail.com

***Abstract*—** The Covid-19 Pandemic has caused a global economic slowdown, leading to a decrease in consumer spending and a decrease in demand for goods and services. This has resulted in a decrease in production and a decrease in employment. As a result, companies are facing financial pressures and are unable to pay their employees. These measures include social distancing, frequent hand washing, wearing masks, and avoiding large gatherings. Governments around the world are also providing economic stimulus packages to help businesses and people cope with the economic impact of the pandemic. Vaccines are also being developed and distributed to help fight the virus. Furthermore, more research is being conducted in order to improve existing treatments and develop new treatments for COVID-19. Finally, authorities are also taking steps to ensure that the public is aware of the necessary precautionary steps so that the spread of the virus can be contained and the pandemic can be overcome. But then things will never get back to normal until more than 80% of the country's population is being vaccinated, which in itself will take time. Till then Face Mask & Hand Sanitization are the only vaccines we have. With the upliftment of lockdown Offices, Factories, Public Places are going back to the new normal with precautionary measures. But even today a large section of people takes things lightly and do not take precautionary measures like Face Mask seriously. This paper is a small step to automatically detect face masks and provide a precautionary measure because safety is the only cure for this pandemic.

Keywords— Convolutional Neural Network, Face Detection

I. INTRODUCTION

Intelligent Mask Detection & Identification Alert System is a Deep learning project built completely written in Python using various dependencies. The project focuses on providing a Real-Time Face Mask Detection System with self-learning capability. Further, It is also tied with a Real-Time Alert mechanism to guide people to wear a mask and sanitize at the doorstep. With the spread of Covid-19 across the globe, Things will not get normal until 80% of the population will be vaccinated. Till then Face Mask and Hand Sanitization are the only vaccines we can rely on.

II. TECHNOLOGY USED

A. Python

Python syntax is as simple as plain English. This allows a developer to focus on design patterns rather than having a complete concentration on the coding part. Python is open source and python can be easily integrated with web frameworks. It has support for various computer vision libraries and machine learning libraries. It can be simply run with the help of an interpreter and doesn't require its own environment. It can be used on any operating system.

B. OpenCV

OpenCV is an open-source library for computer vision and machine learning, this focus on real-time applications. It provides a wide range of tools and functions for image and video processing, including feature detection, object recognition, and machine learning algorithms. It is also designed to be highly optimized for performance, making it suitable for use in commercial products. The BSD license allows for easy integration and modification of the code in commercial projects.

C. Pillow

Pillow is an open-source Python library built on top of PIL (Python Image Library), which provides a wide range of image processing capabilities. It supports a variety of image file formats, including jpeg, png, bmp, gif, ppm, and tiff. Pillow also provides additional features and functionality over PIL, such as support for Python 3, and a more user-friendly API. With Pillow, you can perform various operations on digital images, such as point operations, filtering, and color space conversions, as well as more advanced image processing tasks like image resizing, cropping, and rotating. It is a powerful library that allows developers to easily manipulate and process images in various ways.

D. Pycharm

PyCharm is a popular IDE for Python development, created by JetBrains. It provides a wide range of features and tools to help developers write, test, and debug Python code. Some of the key features of PyCharm include:

- Code completion and error highlighting: PyCharm can help you write code faster by providing suggestions for code completion and highlighting errors as you type.

- Intelligent code navigation: PyCharm can help you navigate through your codebase quickly and easily, with features like search, go-to-definition, and code refactoring.
- Integrated debugging and testing: PyCharm includes a built-in debugger and test runner, making it easy to find and fix errors in your code.
- Support for popular web frameworks: PyCharm has built-in support for popular web frameworks like Django, Flask, and Pyramid, which can help you to work more efficiently.
- Other features: PyCharm also provides features such as version control integration, integration with databases, and support for scientific libraries like NumPy and Matplotlib.

PyCharm is available in two editions: Community and Professional. The Community edition is free and open-source, while the Professional edition is paid and includes additional features like remote development and web development frameworks

E. Haar Cascade

The Haar Cascade algorithm uses the concept of Haar features, integral images, Adaboost training, and cascading classifiers to detect objects in images and videos. The algorithm is trained on a large dataset of positive and negative images, and the trained cascade function is then used to detect objects in other images. The algorithm is widely used in computer vision applications such as face detection, object tracking, and security systems. It is considered to be an efficient and effective object detection algorithm, but it is not as accurate as some other more recent deep learning-based algorithms.

III. Literature Survey

A. Image Generation

OpenCV is a popular library for computer vision tasks, and it is often used in projects related to face detection and object tracking. It can be used to process live video streams and detect faces in real-time by using pre-trained cascaded classifiers or deep learning-based models.

The OpenCV library can be used to detect faces by using the Haar Cascade algorithm, which is a machine learning-based approach for object detection. This algorithm can be trained on a large dataset of positive and negative images, and the trained cascade function can then be used to detect faces in other images.

It is said that the application transmits a snapshot of the live video to the model for mask detection every second. This is a common approach when working with real-time video streams. By taking periodic snapshots of the video, the application can detect faces and other objects in the images and perform additional processing, such as mask detection.

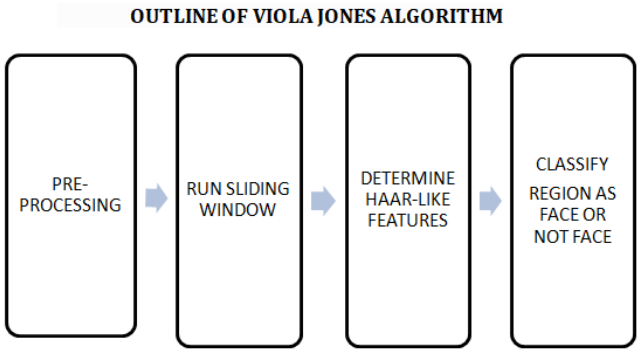
It is important to note that using OpenCV for face detection and mask detection is a good choice for real-time application, however for more accurate results, deep learning-based models like YOLO, RetinaNet, or SSD can be used with OpenCV.

B. Face Detection

Our primary point of emphasis in social situations is the face, which is crucial for expressing identity and emotions. Because they can advance both theoretical understanding and real-world applications, computational models of face identification are intriguing. A wide range of jobs, including criminal identification, security systems, image and film processing, identity verification, tagging for purposes, and human-computer interaction, could be performed by computers that can detect and recognize faces.

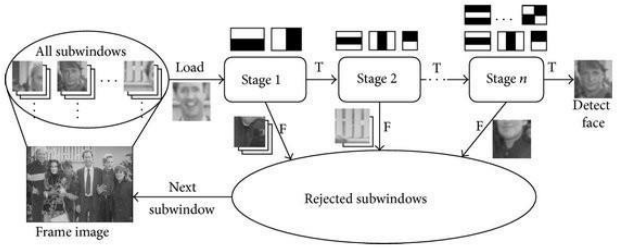
C. Viola-Jones algorithm for Face Detection

Paul Viola and Michael Jones' paper, "Rapid Object Detection with a Boosted Cascade of Basic Features," describes an efficient object recognition technique that uses Haar feature-based cascade classifiers. Using machine learning, a cascade function is trained using a large number of both positive and negative images. The next step is to utilise it to find items in other pictures.

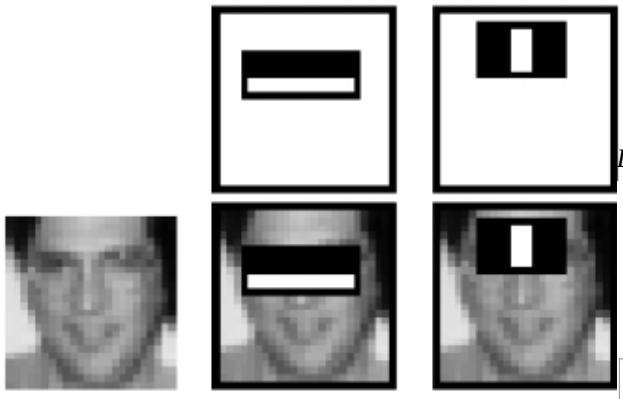


- PRE-PROCESSING: The pre-processing stage prepares an image for the classifiers to run on. At this stage, images are created from a live video feed, converted to grayscale, and then down-sampled in order to detect faces
- RUNNING THE SLIDING WINDOW: The sliding window is a fixed window with a size of 24x24 that moves pixel-by-pixel over the image while applying various filters to the region it covers. These filters show some characteristics that can then be used to categorise the area as a face or not.
- HAAR FILTERS: The region of the image it confines receives numerous filter calls from the sliding window. These filters, known as Haar Filters, make the image's horizontal and vertical details visible. At a certain point in the detection window, a Haar-like feature takes into account adjacent rectangular sections, adds the pixel intensities in each sector, and then determines the difference between these sums. Subsections of an image are then categorised using this difference.
- CASCADE CLASSIFIER: Each sub-window is subjected to the application of a single big classifier that contains a collection of Haar-like features. An individual characteristic is used by the algorithm at

each step to categorise a sub-window. If the sub-window satisfies the requirements, the algorithm continues by applying further characteristics; if not, the sub-window is dismissed. The programme compares the results against tried-and-true classifiers at each level, which makes intuitive sense. Early stages are simpler, whereas later stages require a lot of computation and are more challenging to pass.



To train the classifier, the algorithm first requires a large number of both positive (pictures of faces) and negative (images without faces). After that, features are taken out of it. The haar features depicted in the graphic below are employed for this. They resemble our convolutional kernel exactly. The sum of the pixels under the white rectangle and the sum of the pixels under the black rectangle are subtracted to provide a single value for each feature. Yet, the majority of these estimated attributes are irrelevant. For instance, think about the photo below. There are two excellent characteristics on the top row. The fact that the area around the eyes is frequently darker than the area around the nose and cheeks seems to be the primary emphasis of the first trait chosen.



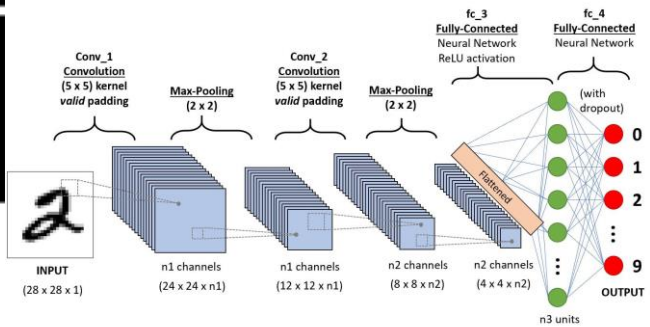
$$f(x,y) = \sum_i p_b(i) - \sum_i p_w(i)$$

Using a concept called Adaboost, which both selects the best features and trains the classifiers that use them, it is possible to choose the best features out of more than 160000 features. By linearly combining weighted, straightforward "weak" classifiers, this approach creates a "strong" classifier. During the detection phase, a window the target size is moved over the input image as each segment and the Haar characteristics are calculated. Then, this difference is contrasted with a learnt threshold that distinguishes between objects and non-objects. Since each Haar feature is simply a "weak classifier" (its detection quality is barely better than random guessing), many Haar features must be combined into cascade classifiers in order to create a strong classifier that can accurately characterise objects..

Classifier Cascade The cascade classifier is made up of a number of steps, each of which is made up of weak learners. Decision stumps are straightforward classifiers that are lousy learners. With the aid of the boosting method, each step is trained. By using a weighted average of the choices made by the weak learners, boosting makes it possible to train a classifier that is incredibly precise. Each classifier step assigns a positive or negative label to the area that is specified by the sliding window's current position. Positive denotes the discovery of an object, whereas negative denotes the absence of any discoveries. If the label is negative, the region has been properly classified, and the detector moves the window to the next spot. The classifier advances the region to the following stage if the label is positive. When the region is classified as positive at the final stage, the detector reports an object found at the current window location. The phases are created to quickly reject negative samples. The vast majority of windows are assumed to be empty of the target object. On the other hand, real positives are uncommon and worth checking.

- When a positive sample is appropriately categorized, a true positive occurs.
- When a negative sample is incorrectly identified as positive, a false positive result.
- When a positive sample is wrongly labelled as negative, a false negative result.

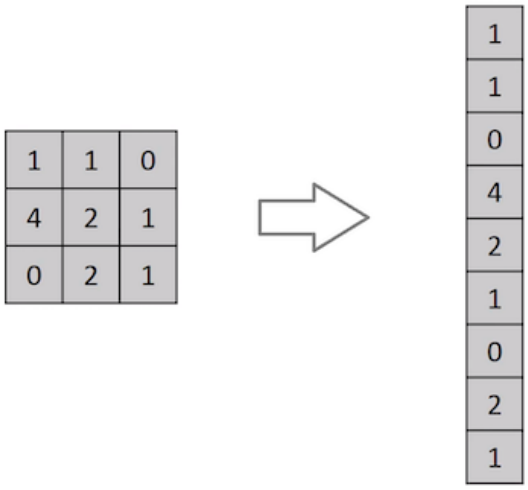
D. Convolutional Neural Network



A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning method that can take in an input image, give various elements and objects in the image importance (learnable weights and biases), and be able to distinguish between them. Comparatively speaking, a ConvNet requires substantially less pre-processing than other classification techniques. ConvNets can learn these filters/characteristics with adequate training, whereas with primitive approaches filters are hand-engineered.

A ConvNet's architecture was influenced by how the Visual Cortex is organised and is similar to the connectivity network of neurons in the human brain. Individual neurons only respond to stimuli in the Receptive Field, which is a little area of the visual field. There are several overlapping fields like this that make up the total visual field.

Why ConvNets over Feed-Forward Neural Nets?

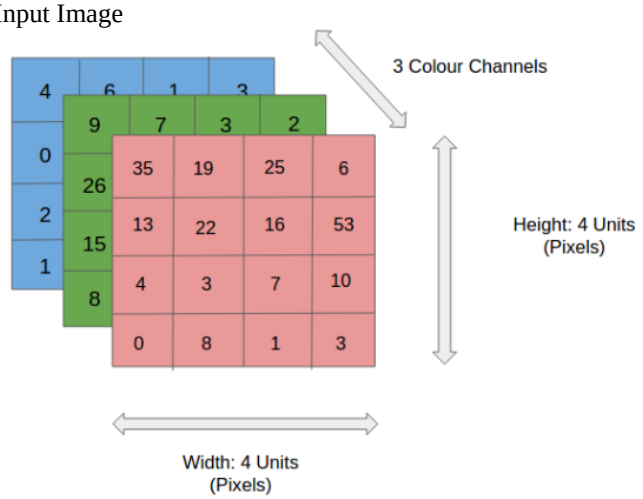


Simply put, a picture is a matrix of pixel data. Why not simply flatten the image (e.g., turn a 3x3 image matrix into a 9x1 vector) for classification purposes? truly not.

For relatively straightforward binary images, the method might perform class prediction with an average precision score, but for complex images with internal pixel dependencies, it would perform with little to no accuracy..

A ConvNet may successfully capture the spatial and temporal dependencies in a picture by employing the appropriate filters. The architecture offers a better fitting to the picture dataset because there are less parameters to take into account and the weights can be reused. In other words, the network might be given instructions to help it understand how complicated the image is.

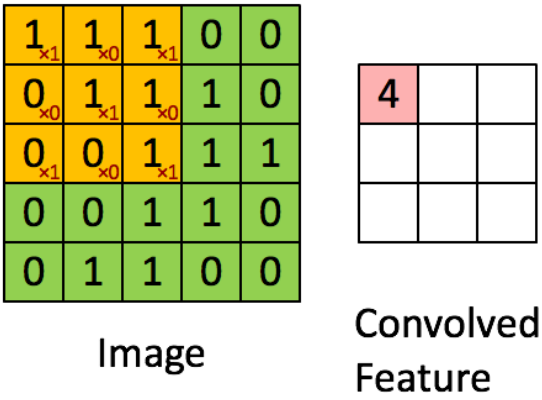
Input Image



Red, Green, and Blue, the three colour planes of the RGB image, have been used to divide it in the image. Several different colour spaces, such as grayscale, RGB, HSV, CMYK, etc., are all used to store images.

You can imagine how computationally intensive things would get once the photographs reach sizes like 8K (7680x4320). The goal of the ConvNet is to compress the images into a more manageable format without losing crucial components needed to produce an accurate forecast. This is critical when developing an architecture that is both efficient at learning features and scalable to huge datasets.

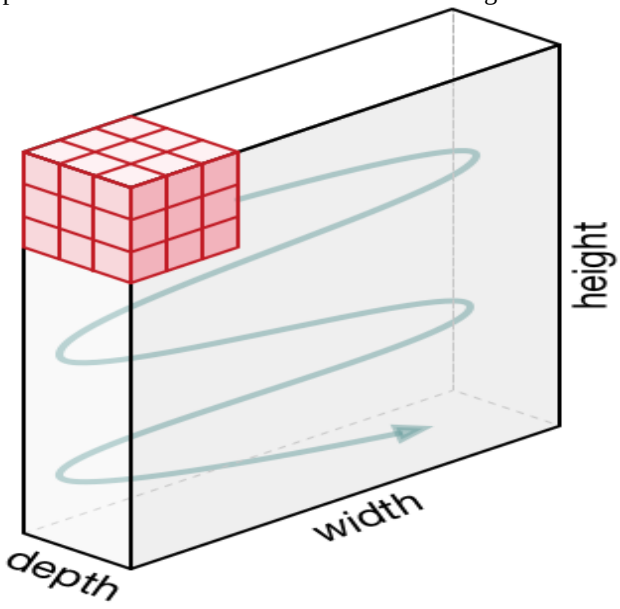
Convolution Layer — The Kernel



Dimensions of the image are 5 (height) x 5 (breadth) x 1 (Number of channels, eg. RGB)

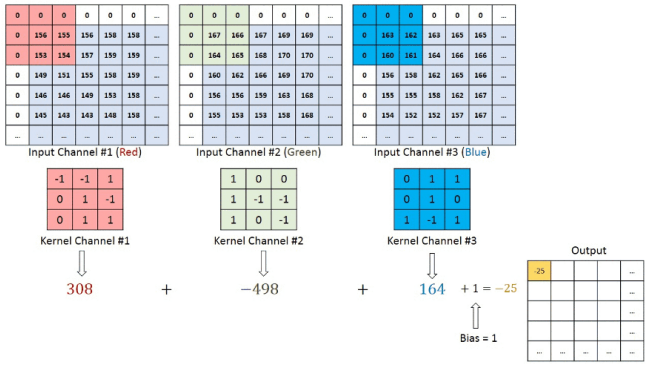
The demonstration above mimics our 5x5x1 input image with a green region. The element that executes the convolution process in the first part of a convolutional layer is the Kernel/Filter, K, which is symbolised by the colour yellow. K is represented as a 3x3x1 matrix..

When the stride length is set to 1, the kernel shifts by one pixel at a time, performing a matrix multiplication operation between the kernel (K) and the portion of the image (P) over which the kernel is currently positioned. This process is repeated 9 times in total to cover the entire image.



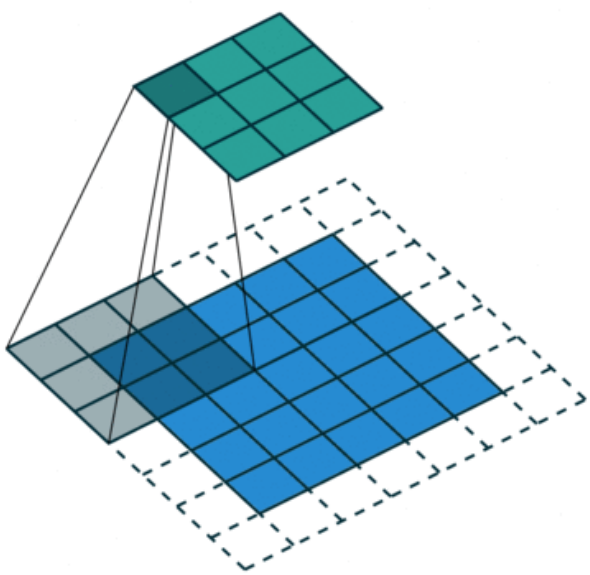
When convolving the filter over the image, it typically starts at the top-left corner of the image and moves to the right

with a certain stride value until it reaches the end of the current row. Then, it "hops" down to the next row, again starting at the left side, and continues the process until the entire image is traversed. This process is done with the stride value set, it could be 1 or more depending on the requirement.



The kernel has the same depth as the input image when dealing with images that have many channels, such as RGB images. This allows for a matrix multiplication operation to be conducted between each channel of the kernel and the corresponding channel of the input image. The results of these matrix multiplications are then added together with a bias term to produce the convoluted feature output, a single output channel. The convolution operation's goal is to extract from the input image high-level features like edges, patterns, and textures.

Convolutional Neural Networks (ConvNets) are not limited to only one convolutional layer. Conventionally, the first convolutional layer is responsible for capturing low-level features such as edges, color, and gradient orientation. As the network progresses, with added layers, the architecture adapts to high-level features such as object shapes and patterns. This provides the network with a comprehensive comprehension of the photos in the collection, just like a human would.



The convolution operation produces two different kinds of results: one where the dimensionality of the convolved feature is decreased in comparison to the input, and the other where the dimensionality is either raised or stays the same.

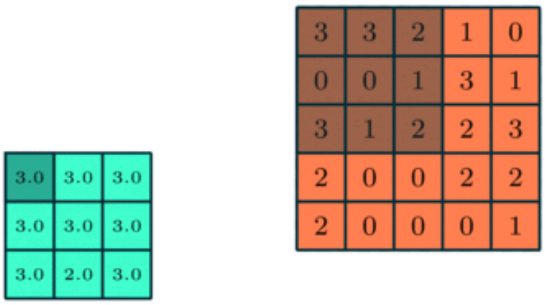
When the convolved feature is less dimensional than the input, valid padding is employed. This is accomplished by not padding the input picture further before applying the kernel. This can be seen in the example you gave, where a 3x3x1 kernel and a 5x5x1 image are convolved to produce a 3x3x1 convolved matrix..

On the other hand, when the dimensionality of the convolved feature is either raised or stays the same as the input, the same padding is applied. This is accomplished by increasing the input image's padding so that the output feature, following convolution, is the same size as the input feature. In the example you gave, a 5x5x1 image is enhanced to a 6x6x1 image and then convolved with a 3x3x1 kernel to get a 5x5x1 convolved matrix, you can see how this is done.

The stride length also plays a role in determining the size of the convolved feature. A larger stride length results in a smaller convolved feature, while a smaller stride length results in a larger convolved feature.

It is important to note that the padding and stride length can be adjusted according to the specific needs of the problem. The repository you mentioned is a great resource for visualizing the effect of different padding and stride length values on the convolution operation.

Pooling Layer



Via dimensionality reduction, the pooling layer is utilised to reduce the computing load needed to process the data by shrinking the spatial size of the convolved feature. The process of efficiently training the model can be maintained by extracting dominant characteristics that are rotationally and positionally invariant.

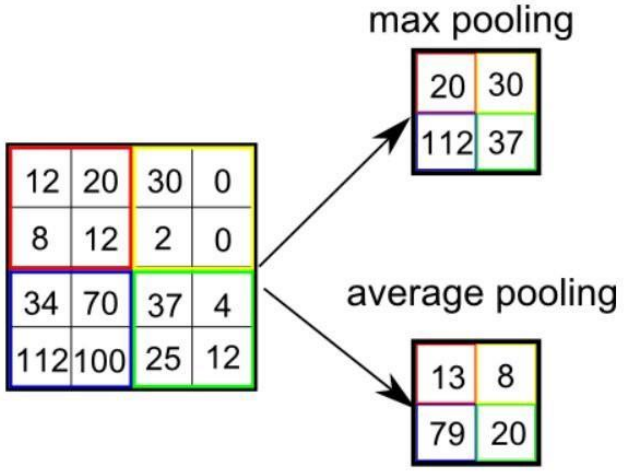
Max pooling and average pooling are the two types of pooling. In contrast to average pooling, which returns the average of all the values from the portion of the picture covered by the kernel, max pooling returns the highest value from the area of the image covered by the kernel.

Max pooling outperforms average pooling in terms of effectiveness because it also reduces noise. It also does de-noising and dimensionality reduction in addition to completely discarding the noisy activations. Average

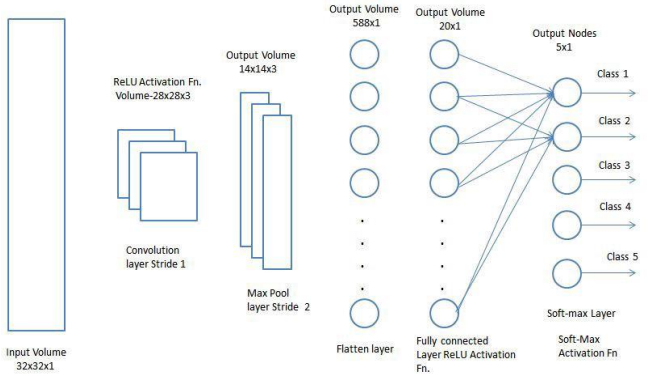
pooling, on the other hand, merely carries out dimensionality reduction as a noise-suppressing strategy.

The i-th layer of a convolutional neural network is made up of the convolutional layer and the pooling layer. The number of these layers may be expanded to capture even more minute details, but doing so will require more computer power depending on how complex the images are.

After going through the approach outlined above, we were able to successfully help the model comprehend the features. Next, we will flatten the output for classification purposes and feed it into a standard neural network. Final predictions are often made by running the output through a completely connected layer, also referred to as a dense layer.



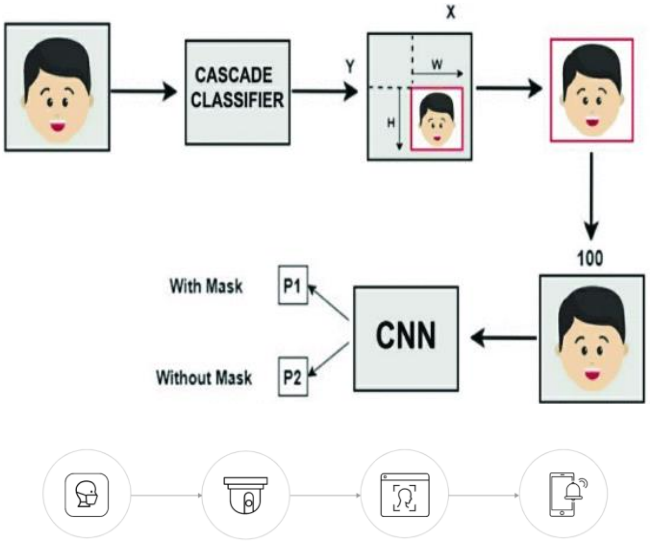
Classification — Fully Connected Layer (FC Layer)



The Fully Connected (FC) layer, also known as a dense layer, is used to make predictions based on the high-level features extracted by the convolutional and pooling layers. The FC layer receives the flattened output of the previous layers and applies non-linear transformations through the use of activation functions, such as ReLU or sigmoid. The output of the FC layer is then passed through a final layer, such as a softmax layer, which performs the classification task using the technique of probability estimation. The model is then trained using backpropagation, and the parameters are adjusted to minimize the error between the predicted and actual outputs. Overall, the combination of convolutional, pooling, and fully connected layers forms a powerful architecture for image classification tasks, known as Convolutional Neural Networks (CNNs).

IV. PROPOSED

The proposed method consists of a cascade classifier and a CNN which contains for face mask detection is as follows



INPUT: Dataset including faces with and without masks.
OUTPUT: Categorized image depicting the presence of face mask.

ALGORITHM:

- REAL-TIME COMPUTER VISION

The OpenCV library proved to be versatile enough for the project, as it can detect and highlight faces in real time by drawing a rectangle around them. The application takes a snapshot of the live footage every second while faces are being detected and sends it to the model for mask detection.

IMAGE GENERATION USING LIVE VIDEO STREAM

```
import cv2
img=cv2.VideoCapture(0) #Id of device passed as
parameter,0 for default
webcam
while True:
ret,frame=img.read()
if ret==False:
continue
gray_frame=cv2.cvtColor(frame,cv2.Color_BGR2GRAY)
cv2.imshow('frame',gray_frame)
key_pressed=cv2.waitKey(1)&0XFF
if(key_pressed==ord('q')):
break
img.release()
cv2.destroyAllWindows()
```

- FACE DETECTION

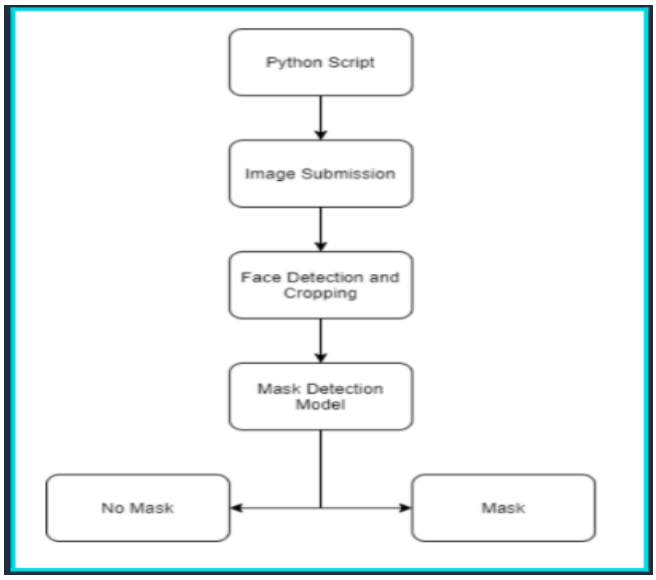
After going through the approach outlined above, we were able to successfully help the model comprehend the features. Next, we will flatten the output for classification purposes and feed it into a standard neural network. Final predictions are often made by running the output through a completely connected layer, also referred to as a dense layer.

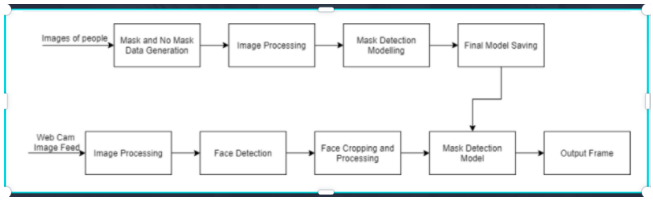
- MASK DETECTION

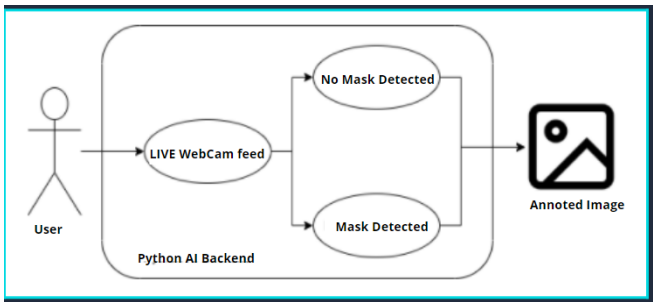
Convolutional Neural Networks, a Deep Learning technique, are used to build the Face Mask detection system (CNN). The Keras library's sequential API is used.

- NOTIFICATION

Once the user is identified without a mask for a particular duration, Admin will be notified via email and then the user can be directed to wear a mask & sanitize their hands.







V. CONCLUSION

In conclusion, this research paper presents a Mask Detection System (MDS) based on Convolutional Neural Network (CNN) architecture. The objective of the MDS is to automatically identify whether individuals in images or videos are wearing masks or not. The system aims to assist in enforcing mask-wearing policies in public spaces, thereby contributing to the prevention and control of infectious diseases. The research utilized a CNN model due to its effectiveness in image classification tasks. The architecture consisted of multiple convolutional layers followed by pooling and fully connected layers. The model was trained on a large dataset containing images of individuals with and without masks, ensuring a diverse range of scenarios and variations in appearance. To train the CNN model, a combination of data augmentation techniques, such as rotation, scaling, and horizontal flipping, were applied to increase the robustness and generalization of the model. The dataset was split into training, validation, and testing sets, ensuring a fair evaluation of the model's performance. The experimental results demonstrated the effectiveness of the proposed MDS. The model achieved a high accuracy in mask detection, indicating its ability to differentiate between masked and unmasked individuals accurately. The precision, recall, and F1-score metrics were also evaluated, showing a balanced performance across different evaluation criteria. Moreover, the MDS showed promising results in real-world scenarios, handling various challenges, such as different types of masks, diverse facial appearances, and different lighting conditions. This suggests the system's potential for practical implementation in public spaces, where real-time mask detection is essential for enforcing preventive measures. The research also discussed the limitations and future directions for improvement. Despite achieving high accuracy, the model may still encounter challenges in cases where individuals are wearing unconventional masks or partially covering their faces. Future work could focus on addressing these challenges and further enhancing the system's performance. In summary, this research paper presents a Mask Detection System based on a Convolutional Neural Network, which demonstrates high accuracy and robustness in identifying whether individuals are wearing masks. The system has potential applications in public health, safety, and compliance monitoring, contributing to the prevention and control of infectious diseases in various settings.

REFERENCES

[1] L. Haoxiang and Lin, "A Convolutional Neural Network Cascade for Face Detection", Proc. CVPR, pp. 5325-5334, 2015.

[2] H. Hatem, Z. Beiji and R. Majeed, "A Survey of Feature Base Methods for Human Face Detection", *International Journal of Control and Automation*, vol. 8, no. 5, pp. 61-78, 2015.

[3] M. Loey, G. Manogaran, M. Hamed and N. Taha, "A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic", *Journal of the International Measurement Confederation (IMEKO)*, January 2021.

[4] R. P. Sidik and E. Contessa Djamal, "Face Mask Detection using Convolutional Neural Network," *2021 4th International Conference of Computer and Informatics Engineering (IC2IE)*, 2021, pp. 85-89, doi: 10.1109/IC2IE53219.2021.9649065.

[5] G. Howard, M. Zhu, B. Chen et al., "Mobilenets:efficient convolutional neural networks for mobilevisionapplications," 2017, https://arxiv.org/abs/1704.04861.

[6] M. S. Ejaz and M. R. Islam, "Masked Face Recognition Using Convolutional Neural Network," 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI), 2019, pp. 1-6, doi: 10.1109/STI47673.2019.9068044

[7] Iqbal, Zubair; Gupta, Prachi; Gola, Kamal Kumar; " Visualization of COVID-19 Data using Jupyter Notebook", Dogo Rangsang Research Journal UGC Care Group I Journal, Vol-10 Issue-07 No. 1 July 2020, ISSN : 2347-7180.

[8] K. Suresh, M. Palangappa and S. Bhuvan, "Face Mask Detection by using Optimistic Convolutional Neural Network," *2021 6th International Conference on Inventive Computation Technologies (ICICT)*, 2021, pp. 1084-1089, doi: 10.1109/ICICT50816.2021.9358653.

[9] S. Yadav "Deep learning based safe social distancing and facemask detection in public areas for COVID-19 safety guidelines adherence" Int J Res Appl Sci Eng Technol. vol. 8 no. 7 pp. 1368-1375 2020.

[10] Tsega, Tajebe & Kumar, Deepak. (2021). Covid-19 Face Mask Detection Using Convolutional Neural Network and Image Processing. 1-7. 10.1109/INCET51464.2021.9456288.