WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Face Mask Detection and Alert System Using Deep Learning

Riya Phulewar[1], Vansh Nagardhankar[2], Aniket Hatewar[3], Pratik Bobate[4]

Professor, Department of Computer Science Engineering, Govindrao Wanjari College of Engineering and Technology, Nagpur, Maharashtra, India

ABSTRACT

Pandemic on a global scale COVID-In a worldwide outbreak of hazardous sickness, 19 situations developed. Wearing a face mask can assist to prevent the transmission of infection and will protect the individual from any airborne infectious bacteria. Face Mask Detection Systems can detect whether or not people are wearing masks. For image detection, the HAAR-CASACADE algorithm is utilized. This classifier, when combined with other current algorithms, achieves a high recognition rate even with fluctuating expressions, efficient feature selection, and a low assortment of false positive features. The HAAR feature-based cascade classifier system uses just 200 features out of 6000 features to achieve an 85-95% identification rate. We require mask detection as a distinct public health service system during the worldwide pandemic COVID-19 outbreak. Face mask and non-face mask images are used to train the model. When the program recognizes the mask on the user's face, it will emit a beep alarm sound.

Keywords - COVID-19 epidemic, HAAR-CASACADE algorithm, mask detection, face mask image, non-face mask image

INTRODUCTION

The vaccine that can effectively treat Covid-19 has not yet been developed, and the globe has not yet fully recovered from this epidemic. However, numerous governments have permitted a small number of economic activities to be resumed once the number of new cases of Covid19 has decreased below a specific level to lessen the pandemic's impact on the nation's economy. Concerns about worker safety in the new post-Covid-19 climate have surfaced as these nations carefully resume their economic operations. It is recommended that individuals wear masks and keep a distance of at least one meter between each other to limit the risk of infection. Deep learning has received greater attention in the field of object detection and was utilized to produce a face mask-detecting device that can determine whether or not someone is wearing a mask. Real-time streaming from the Camera may be examined to evaluate the categorization results. We require training data collection for deep learning applications. It is the real dataset that was used to train the model to carry out different tasks.

LITERATURE SURVEY

A Smart City Network Facial Mask Detection Automated System to Reduce COVID-19 The new coronavirus that triggered the COVID-19 pandemic is still spreading around the world today. COVID-19's effects have been felt in practically all development-related fields. There is a problem in the healthcare system. Wearing a mask is one of several precautions that have been done to stop the spread of this disease. In this research, we provide a method that limits the spread of COVID-19 by identifying individuals in a network of smart cities where all public spaces are watched over by Closed-Circuit Television (CCTV) cameras. If a person without a mask is found, the appropriate authority is notified using the city's network. A collection of photos of people wearing and not wearing masks that were gathered from multiple sources was used to train a deep learning architecture. For never-before-seen test data, the trained architecture distinguished between persons wearing facial masks and those wearing none with 98.7% accuracy. Our research might potentially help many nations across the world by limiting the spread of this contagious disease.

Using a Convolutional Neural Network, Masked Face Recognition [2]: A common and important technology in recent years is facial recognition. It's too difficult because of the masks' variety and altered faces. Masking is another typical circumstance in the real world where a person is uncooperative with technology, such as in video surveillance. Current facial recognition technology performs worse with these masks. For recognizing faces in various situations, such as shifting attitude or illumination, damaged photos, etc., several studies have been conducted. However, challenges brought on by masks are typically ignored. The main goal of this research is to improve the recognition accuracy of various masked faces, with a focus on facial masks. A workable strategy has been suggested, consisting of first recognizing the face areas. Multi-Task Cascaded Convolutional Neural Network has been used to tackle the obstructed face identification challenge. (MTCNN). The Google Face Net embedding model is then used to extract face characteristics.

EXISTING SYSTEM

With the help of a Multi-Task Cascaded Convolutional Neural Network, the challenge of face identification has been addressed. (MTCNN). The Google Face Net embedding model is then used to extract face characteristics. 1. Both datasets of people wearing masks and those who aren't can be used to train this system. Following model training, the system can determine if a person is donning a mask or not.

METHODOLOGY

PROPOSED SYSTEM

1.This system is capable to train the dataset of both persons wearing masks and without wearing masks.

2.After training the model the system can predicting whether the person is wearing the mask or not .

3.It also can access the webcam and predict the result.

4.It will produce a beep sound once face without mask detect to produce an alert

TENSORFLOW FRAMEWORK:

An open-source software library called TensorFlow exists. Researchers and engineers were the ones who first created tensor flow. To perform machine learning and deep neural network research, it is working on the Google Brain Team under Google's Machine Intelligence research group. Deep learning workloads as well as other statistical and predictive analytics workloads may be executed on this open-source platform. It is a Python package that supports a variety of deep learning and regression methods in general. A free and open-source software library called TensorFlow is used for differentiable programming and dataflow across a variety of activities. It is a symbolic math library that is also utilized by neural network applications in machine learning. Google uses it for both research and manufacturing. TensorFlow is the second generation of the Google Brain system. On February 11th, version 1.0.0 was made available. TensorFlow can run on several CPUs and GPUs, unlike the reference implementation, which only utilizes a single device. (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). On 64-bit Linux, macOS, Windows, and mobile operating systems like Android and iOS, TensorFlow is accessible. Because of its adaptable design, computing may be easily deployed across a range of platforms (CPUs, GPUs, and TPUs), from desktop computers to server clusters to mobile and edge devices. The operations that these neural networks carry out on multidimensional data arrays, known as tensors, are where the name Tensorflow originates.

OPENCV:

1. We can create cross-platform, real-time computer vision applications utilizing this framework.

2. It primarily focuses on image processing, video recording, and analysis, including tools for object and face identification.

3. Open CV is currently accessible on several platforms, including Windows, Linux, OS X, Android, and iOS, and supports a broad range of programming languages, including C++, Python, and Java.

4. For high-speed GPU tasks, interfaces based on CUDA and OpenCL are also being actively developed. The Python API for Open CV is called Open CV-Python.

5. It combines the greatest features of Python and the Open CV C++ API.

6. The open-source computer vision and machine learning software library OpenCV is available for free. OpenCV was created to offer a standard infrastructure to facilitate the use of machine perception in commercial goods and computer vision applications. OpenCV makes it simple for companies to utilize and alter the code because it is a BSD-licensed product.

7. There are more than 2500 optimized algorithms in the library, including a wide range of both traditional and cutting-edge computer vision and machine learning techniques. 8. Algorithms can be used to find related images from an image database, remove red eyes from flash-taken photos, follow eye movements, classify human actions in videos, detect and recognize faces, identify objects, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce high-resolution images of entire scenes, and identify scenery and set up markers to add augmented reality on top of it, etc.

NUMPY:

Large, multi-dimensional arrays and matrices are supported by NumPy, a library for the Python programming language, along with a substantial number of high-level mathematical operations that may be performed on these arrays. Jim Hugunin originally developed Numeric, the predecessor of NumPy, with assistance from a number of other programmers. Travis Oliphant developed NumPy in 2005 by heavily altering Numeric to incorporate the capabilities of the rival Num array. Numerous people have contributed to the open-source program NumPy. Although the Python programming language

wasn't initially intended for numerical computing, the scientific and engineering community quickly became interested in it, leading to the establishment of the matrix-sig special interest group in 1995 with the goal of developing an array computing framework. Its members included Guido van Rossum, the author, and maintainer of Python, who added enhancements to the language's grammar (namely the indexing syntax) to facilitate array computation. Jim Fulton finished a matrix package implementation, then Jim Hugunin generalized it to become Numeric, often known as Numerical Python extensions or NumPy. Hugunin, a Ph.D. student at MIT, joined the Corporation for National Research Initiatives (CNRI) to work on J Python in 1997, handing the maintainer role to Paul Dubois of Lawrence Livermore National Laboratory (LLNL). Travis Oliphant, a developer of NumPy, translated the functionality of num-array to Numeric in early 2005 in order to unify the community behind a single array package. NumPy 1.0 was then released in 2006. The new endeavor was a component of SciPy. This new package, named NumPy, was created in order to avoid having to install the substantial SciPy package merely to obtain an array object.

MATPLOT:

For the Python computer language and its NumPy numerical mathematics extension, Mat plot is a charting library. Plots may be embedded into programs utilizing general-purpose GUI toolkits such as Tkinter, WX Python, Qt, or GTK+ using this object-oriented API. It is not recommended to utilize the procedural "Pylab" interface, which is built on a state machine (like OpenGL) and is intended to closely mimic the MATLAB interface. Matplotlib is used by SciPy. John D. Hunter initially created Matplotlib, which is available under a BSD-style license and has a thriving development community. Just before John Hunter passed away in August 2012, Thomas Caswell and Michael Droettboom were named matplotlib's primary developers.

IPYTHON What exactly is Python?

You might be curious about it. If you want to study editing but are unfamiliar with editing languages, you could be referring to this book. Alternatively, you could be familiar with "big word" programming languages like C, C++, C#, or Java and want to learn more about Python and how it differs from these.

PYTHON CONCEPTS

If you are not interested in the how and why of Python, you can go to the next chapter. I'll try to explain why Python is, in my opinion, one of the greatest programming languages out there and why it makes for such an excellent starting point in this chapter. Programming language Python has evolved into one that is simple to use. It has less syntax than other languages and utilizes English words rather than punctuation. A very advanced, translated, interactive, and object-oriented language is Python. Python translation: Python is being processed by the interpreter at startup. You don't have to install your program before utilizing it. Editing languages like PERL and PHP are comparable to this. Python interactive - You may use the Python Prompt to interact with the interpreter and create your apps. Python supports the object-oriented programming style or approach, which encapsulates the code in objects. Python is a wonderful language for novices because it enables the building of a wide range of programs, from basic text apps to web browsers and games.

1Python Features

Features of Python include:

1. Python is simple to learn since it has a limited set of keywords, a clear structure, and well-defined syntax. This enables the pupil to pick up the language more quickly.

Python code is easily readable and can be seen with the unaided eye.

3. Python source code is simple to keep up-to-date.

4. Standard General Library – The bulk library in Python is extremely portable and fast-compatible with UNIX, Windows, and Macintosh.

5. Interaction mode - Python has an interaction mode, which enables interaction testing and caption mistake correction.

6. Portable - Python has the same user interface on all computer platforms and operates on a wide range of them.

7. Extensible - The Python interpreter may be expanded using low-level modules. Installing or customizing these modules enables system developers to increase the functionality of their tools.

8. Details - Python approaches to meeting provides all significant commercial facts. 9. GUI Programming - Python supports the development and setup of a user interface for pictures of various program phones, libraries, and programs, including Windows MFC, Macintosh, and Unix's X Window.

10. Scalable - Python development and support are beneficial for large applications, whereas Shell programming is not. In addition to the qualities mentioned above, Python has a wealth of helpful features, some of which are detailed below. –

It supports functional and structured programming techniques in addition to OOP.

It supports automatic garbage collection, and dynamic type verification, and provides very high-level dynamic data types. It may be used as a scripting language or converted into Byte-code for large-scale application development.

ADVANTAGES/BENEFITS OF PYTHON:

programming are:

1. The existence of third-party modules: The Python Package Index (PyPI) is home to a large number of third-party modules that enable Python to communicate with the majority of other systems and languages. 9

2. Comprehensive Support Libraries: Python comes with a sizable standard library that covers topics including operating system interfaces, string operations, web services tools, and internet protocols. The standard library has already scripted many high-use programming operations, which considerably minimizes the amount of code that has to be created.

3. Open Source and Community Development: The Python language is created under an open-source license that has received OSI approval, making it available for free use and distribution—including for commercial endeavors. Additionally, the community works on its code by organizing conferences and mailing groups and providing for its many modules, which in turn drives its development.

4. Python's outstanding readability and uncomplicated, simple-to-learn syntax make it easy for newcomers to learn how to use this programming language. PEP 8's code style standards offer a set of recommendations to help with code formatting. The large user base and active developer community have also produced a wealth of online resources to support further development and language adoption.

5. User-friendly Data Structures: List and dictionary data structures that are built into Python may be utilized to create quick runtime data structures. Additionally, Python offers the choice of dynamic high-level data type, which minimizes the amount of support code required.

6. Productivity and Speed: Python's excellent integration and text processing capabilities, together with its own unit testing framework, as well as its clean object-oriented architecture, all help to raise its speed and productivity. Python is regarded as a practical choice for creating intricate multi-protocol network systems. 10 Python has several benefits for software development, as can be observed from the aforementioned statements. Its support base might expand as the language continues to be improved as well.

Python has five standard data types –

- Numbers
- String
- List
- Tuple
- Dictionary

Python Numbers

Numeric values are stored in number data types. When you give a number a value, it becomes a number object.

Python Strings

A string is defined in this Python use as a collection of characters that are encased in quotation marks. Python lets you use as many quotations as you like in pairs. To extract subsets of strings, use the slice operator ([] and [:]), where the indices start at 0 at the beginning of the text and go all the way to -1 at the conclusion.

Python Lists

Lists are the most varied kind of Python data. The list's items are enclosed in square brackets and separated by commas. ([]). In some aspects, lists and C-order are comparable. One distinction between them is that listings can contain a variety of data kinds. To obtain values from a list, use the slide operator ([] and [:]). The indicators start at 0 at the beginning of the list and go to the end of the list. A plus sign (+) serves as the list's concatenation operator, while an asterisk (*) serves as the repeater.

Python Tuples

A data type called a cone resembles a list of elements in a series. Comma-separated values are grouped as a cone. In contrast to the list, the pods are included in brackets. The items and sizes of lists are changeable and are included in brackets ([]), but lumps are enclosed in brackets (()) and cannot be sorted. Powders are just like reading lists.

Python Dictionary

Python dictionaries function like a hash table in some ways. They consist of two key numbers and resemble Perl's combination schemes or hashes. Any Python type can be used as the dictionary key, however, strings and integers are by far the most popular. Python, on the other hand, lets you set its own

prices. Dictionary entries are enclosed in curly braces (), while values are assigned and accessed using square brackets ([). The many modes in Python The two most fundamental Python modes are normal and interactive. The scripted and completed.py files are run in the standard way by the Python interpreter. A command line shell in interactive mode runs previously given statements in active memory while also responding instantly to each statement. As new lines are fed into the interpreter, the feed program is evaluated in part and in its entirety.

PANDAS

Built on top of the NumPy library is the open-source Pandas library. It is a Python module that provides a number of data structures and actions for working with time series and numerical data. It is mostly used for how much simpler it makes importing and analyze data. Pandas is quick and offers users exceptional performance & productivity. Working with "relational" or "labeled" data is made simple and straightforward with the help of the Python module Pandas, which offers quick, adaptable, and expressive data structures. It seeks to serve as the essential, high-level building block for using Python for actual, practical data analysis. Its overarching objective is to develop into the most potent and adaptable open-source data analysis/manipulation tool attainable in any language. It is already making great progress in this direction. Use Python to learn about data analysis. Pandas Data Frames make it simple to manipulate your data, including choosing or changing columns and indexes as well as reshaping it. Pandas are appropriate for a wide range of data types, including 12 Tabular data having columns of many types, such as those found in a SQL database or an Excel spreadsheet Time series data that are organized and unordered (though not always with a set frequency). arbitrary matrix data with row and column labels, whether it is homogeneously or heterogeneously coded by any other kind of statistical or observational data sets. To be put into a Pandas data structure, the data doesn't need to be labeled in any way.

KERAS is an API made with people in mind, not machines. Best practices for lowering cognitive load are followed by Keras, which provides consistent & simple APIs, reduces the number of user activities necessary for typical use cases, and gives clear & actionable error signals. Additionally, it contains a wealth of development instructions and documentation. As well as a variety of tools to make dealing with picture and text data easier, Keras includes multiple implementations of widely used neural network building blocks including layers, objectives, activation functions, and optimizers. This helps to reduce the coding required to create deep neural networks. The GitHub problems page and a Slack channel serve as community support forums, and the source is available on GitHub. Keras is a lightweight Python deep-learning package that may be used with Theano or TensorFlow. It was created to make deep learning model implementation as quick and simple as feasible for research and development.

FOUR PRINCIPLES:

• Modularity: A model may be thought of as a sequence or a graph on its own. A deep learning model's concerns are all separate components that may be merged in any way. • Minimalism: The library delivers only what is necessary to achieve a goal, with minimal frills and a focus on readability.

• Extensibility: New components are designed to be simple to install and use inside the framework, allowing researchers to test and explore new ideas.

• Python: There are no different model files with unique file formats. Everything is written in Python. Keras is built with minimalism and modularity in mind, allowing you to easily construct deep learning models and execute them on top of a Theano or TensorFlow backend.

Machine Learning approaches:

1. Object identification framework based on HAAR features developed by Viola-Jones

2. Feature transform with scale invariance (SIFT)

3. Features of the histogram of oriented gradients (HOG) Machine learning (ML) is the study of computer algorithms that improve themselves automatically over time. It is thought to be a subset of artificial intelligence. Machine learning algorithms construct a mathematical model using sample data, referred to as "training data," in order to make predictions or judgments without being expressly programmed to do so. Machine learning algorithms are utilized in a broad range of applications, including email filtering and computer vision, when developing traditional algorithms to do the required tasks would be difficult or impossible. Machine learning is closely connected to computational statistics, which focuses on utilizing computers to make predictions. Mathematical optimization research provides methodology, theory, and application fields to the subject of machine learning. A similar topic of research is data mining, which focuses on exploratory data analysis via unsupervised learning. Machine learning is sometimes known as predictive analytics when used for commercial concerns.

Approaches for Machine Learning:

Viola-Jones Object detection framework based in HAAR features:

The Viola-Jones method is one of the most widely used for object recognition in images. This study looks at the potential for parametric optimization of the Viola-Jones algorithm in order to attain maximum efficiency under various environmental situations. It is demonstrated that by using further changes, it is feasible to raise the speed of the algorithm in a specific image by 2-5 times while sacrificing just 3-5% of the correctness and completeness of the task.

Scale-invariant feature transform (SIFT):

The scale-invariant feature transform (SIFT) is a computer vision feature identification technique used to find and characterize local features in pictures. Object SIFT key points are first retrieved from a collection of reference pictures and saved in a database. An item in a new picture is recognized by comparing each feature in the new image to this database and discovering possible matching features based on the Euclidean distance of their feature vectors. Subsets of important points that agree on the item and its location, size, and orientation in the new picture are identified from the whole set of matches to filter out excellent matches. Using an efficient hash table implementation of the generalized Hough transform, consistent clusters are determined quickly. Each cluster of three or more features that agree on an object and its 14 poses is then subjected to additional detailed model verification, and outliers are discarded. Finally, the probability that a specific set of features indicates the presence of an object is computed, given the accuracy of fit and the number of probable false matches. Object matches that pass all of these checks are very likely to be accurate.

Histogram of oriented gradients (HOG):

The histogram of oriented gradients (HOG) is a feature descriptor that is used in computer vision and image processing to recognize objects. The approach counts the number of times a gradient orientation appears in a localized section of a picture. This approach is comparable to edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but it is generated on a smaller scale. This suggested system includes an algorithm. Cascade Classifiers Based on HAAR Features 3.11 Cascade Classifiers Based on HAAR Features It is an Object Detection Algorithm that is used to recognize faces in images or real-time video. For enhanced precision, a dense grid of equally spaced cells is used, as well as overlapping local contrast normalization. It is an efficient method of detecting objects. A large number of positive and negative pictures are utilized to train the classifier in this technique. In this experiment, a pre-trained model using frontal traits is constructed and utilized to recognize faces in real time. HAAR Cascade is a machine learning-based strategy that uses a large number of positive and negative pictures to train the classifier. Positive photos: These are the photos that we want our classifier to recognize. Negative Images: Images of everything else that does not include the object we are looking for. WHY IS IT PREFERRED TO USE HAAR FEATURE-BASED CASCADE CLASSIFIERS? A HAAR-like feature has a significant edge over most other features in terms of computation speed. In constant time, a HAAR-like feature of any size may be computed. (approximately 60 microprocessor instructions).

DEEP LEARNING

1. Deep learning is an AI function that replicates the workings of the human brain in processing data for use in object detection, speech recognition, language translation, and decision-making.

2. Deep learning AI can learn without human supervision, using both unstructured and unlabeled data.

3. In this case, face mask identification is accomplished through the use of a Deep Learning technology known as Convolution Neural Networks. (CNN). Deep learning techniques try to learn feature hierarchies composed of lower-level features with features from higher levels of the hierarchy. Automatically learning features at several layers of abstraction enables a system to learn complicated functions mapping input to output directly from data, rather than relying only on human-crafted features. Deep learning methods strive to leverage the unknown structure in the input distribution to identify suitable representations, frequently at many levels, with higher-level learned features described in terms of lower-level features.

NEURAL NETWORKS VERSUS CONVENTIONAL COMPUTERS:

Neural networks learn by doing. They cannot be planned to be squandered, and the network may even be malfunctioning. The downside is that because the network solves the problem on its own, its behavior is unpredictable. Conventional computers, on the other hand, employ a cognitive approach to problem-solving; the method the problem is addressed must be known and specified in concise clear commands. These instructions are then translated into a high-level language program, which is ultimately translated into machine code that the computer understands. These machines are completely predictable; everything that goes wrong is the result of a software or hardware flaw. Neural networks and traditional algorithmic computers do not compete, but rather complement one another. There are some activities that are better suited to an algorithmic approach, such as arithmetic operations, and others that are better suited to neural networks. Furthermore, in order to execute at optimum efficiency, a huge number of jobs necessitate systems that employ a mix of the two methodologies (often, a conventional computer is used to oversee the neural network).

ARCHITECTURE OF NEURAL NETWORKS:

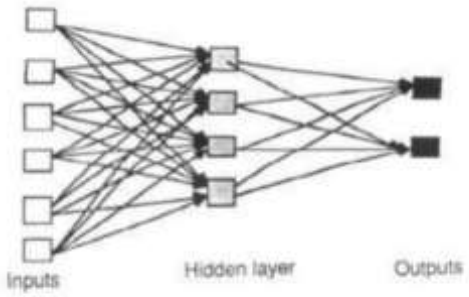
FEED-FORWARD NETWORKS:

Feed-forward ANNs enable signals to move in just one direction: from input to output. There is no feedback (loops), hence the output of one layer does not impact the output of another. Feed-forward ANNs are typically simple networks that connect inputs to outputs. They are often employed in pattern recognition. This organization is also known as bottom-up or top-down. to be simple networks that connect inputs and outputs. They are often employed in pattern recognition. This organization is also known as bottom-up or top-down.

FEEDBACK NETWORKS:

Feedback networks can have signals traveling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; is changing continuously until they reach an equilibrium point. They remain at the equilibrium

point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in a single-layer organization.



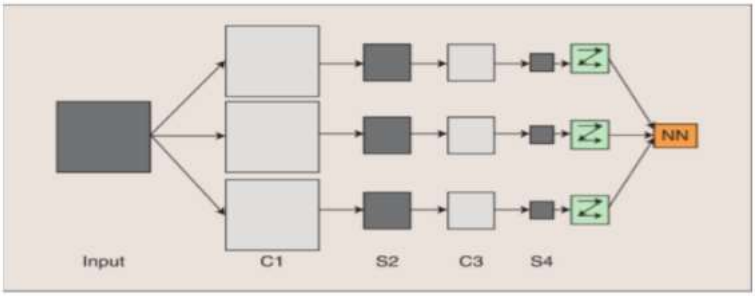
Layers in NN

NETWORK LAYERS:

The most popular sort of artificial neural network is composed of three groups, or layers, of units: an input layer is connected to a hidden layer, which is connected to an output layer. The raw information that is supplied into the network is represented by the activity of the input units. The activities of the input units and the weights on the links between the input and the hidden units determine the activity of each hidden unit. The activity of the hidden units, as well as the weights between the hidden and output units, influence the behavior of the output units. The hidden units in this basic network are allowed to generate their own representations of the input, which makes it intriguing. Because the weights between the input and hidden units govern when each hidden unit is active, a hidden unit may pick what it represents by adjusting these weights. Separate single-layer and multi-layer systems as well. The most general instance is the single-layer organization, in which all units are linked to one another, and it has higher potential computational capacity than hierarchically organized multi-layer organizations. Units in multi-layer networks are frequently numbered per layer rather than globally.

Convolution Neural Network

A convolution neural network is a type of artificial neural network design introduced by Yann Lecun in 1988. Image categorization is one of the architecture's most prominent applications. CNNs have a wide range of applications, including image and video identification, recommender systems, and natural language processing. The example that will be used in this article is connected to computer vision. The essential notion, however, stays the same and may be extended to any other use case! CNNs, like neural networks, are made up of neurons that may be trained to learn weights and biases. Each neuron gets numerous inputs, computes a weighted total, passes it through an activation function, and produces an output. The entire network has a loss function, and all of the neural network tips and tricks still apply to CNNs. More specifically, the picture is processed through a sequence of convolution, nonlinear, pooling, and fully linked layers before being output. 19 A convolutional neural network (CNN, or ConvNet) is a type of deep, feed-forward artificial neural network that is often used to analyze visual images in deep learning. Convolutional networks were motivated by biological processes because the pattern of connection between neurons matches the organization of the visual cortex. When compared to other image classification methods, CNNs require very minimal pre-processing. CNN is a subset of multi-layer NNs that is applied to 2-dimensional arrays (often pictures) and is based on spatially localized neural network input. CNN Make 'patterns of patterns' to aid with pattern identification. Patches from previous levels are combined in each layer. Convolutional Networks are multistage designs that can be trained. Each stage's input and output are sets of arrays known as feature maps. Each feature map at output reflects a specific feature retrieved at all points on input. Each stage is made up of the following layers: a filter bank layer, a non-linearity layer, and a feature pooling layer. A ConvNet is made up of one, two, or three of these three-layer stages, each followed by a classification module.



CONVOLUTIONAL LAYER

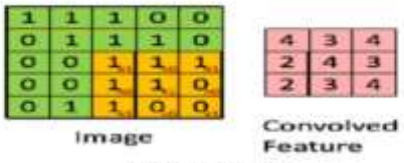
It always comes first. It is fed the picture (a matrix with pixel values). Imagine that the input matrix's reaction starts at the top left of the picture. The program then chooses the smaller matrix there as a filter. The filter then generates convolution that advances along the input picture. The objective of the filter is to multiply its value by the original pixel values. All of these multiplications are added together to provide a single number. Because the filter only read the picture in the upper left corner, it travels one unit right and does a similar action. A matrix is created after passing the filter over all locations,

however, it is smaller than the input matrix. From a human standpoint, this procedure is akin to distinguishing visual borders and basic colors. However, to recognize the fish, the entire network must be used. The network will have numerous convolution layers as well as nonlinear and pooling layers. The first layer to extract features from an input picture is convolution. Convolution features are created by combining tiny squares of input data. It is a mathematical process with two inputs: an image matrix and a filter or kernel.

- An image matrix of dimension (h x w x d)
- A filter (fh x fw x d)
- Outputs a volume dimension (h-fh+1) x (w-fw+1) x1. Consider a 5 x 5 whose image pixel values are 0, 1 and filter matrix 3 x 3 as shown in below

convolution with a filter example

Then the convolution of 5 x 5 image matrix multiplies with 3 x 3 filter matrix which is called "Feature Map" as output shown in below



Output of Convolution layer

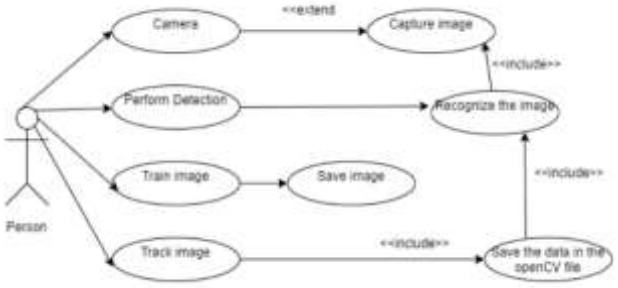
DESIGN AND IMPLEMENTATION

UML Diagrams: It is always first. It is fed the image (a matrix with pixel values). Assume that the reaction of the input matrix begins at the top left of the image. The smaller matrix is subsequently selected as a filter by the program. The filter then produces convolution that moves along the input image. The filter's goal is to multiply its value by the original pixel values. These multiplications are combined together to get a single number. Because the filter only reads the upper left corner of the image, it moves one unit right and performs a similar function. After running the filter over all places, a matrix is formed, but it is less than the input matrix. This method is analogous to recognizing visual boundaries and primary colors in humans. To recognize the fish, however, the full network must be used. There will be several convolution layers, as well as nonlinear and pooling layers, in the network. Convolution is the initial layer used to extract features from an input image. Convolution features are formed by fusing together small squares of input data. It is a mathematical procedure that requires two inputs: an image matrix and a filter or kernel.

Use Case Diagram

In the Unified Modeling Language (UML), a use case diagram can summarise the specifics of your system's users (also known as actors) and their interactions with the system. To construct one, you'll need a collection of specialized symbols and connections. A successful use case diagram may help your team discuss and represent: Scenarios in which your system or application interacts with individuals, organizations, or external systems; and Goals that your system or application assists those entities (known as actors) in achieving.

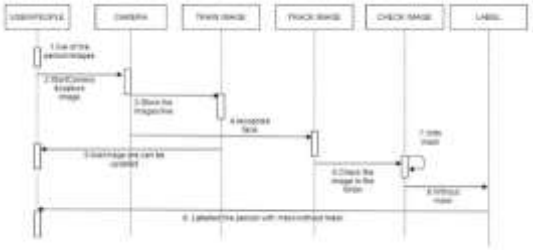
The system's scope



Sequence Diagram

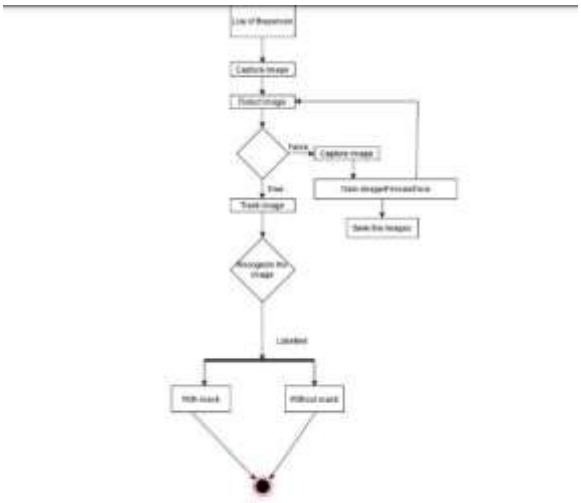
Because it illustrates how and in what order a set of items interacts, a sequence diagram is a sort of interaction diagram. Software engineers and business experts use these diagrams to understand the requirements for a new system or to describe an existing process. Event diagrams and event scenarios are other names for sequence diagrams. Businesses and other organizations can benefit from sequence diagrams. Create a sequence diagram to:

- Represent the details of a UML use case.
- Model the logic of a sophisticated procedure, function, or operation.
- See how objects and components interact with each other to complete a process.
- Plan and understand the detailed functionality of an existing or future scenario.



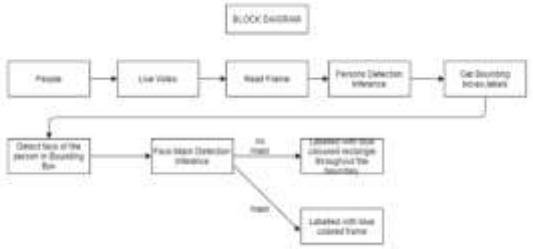
Activity Diagram

An activity diagram is a behavioral diagram, which displays a system's behavior. An activity diagram depicts the control flow from a starting point to a finishing point, highlighting the many decision routes that exist while the activity is being performed.



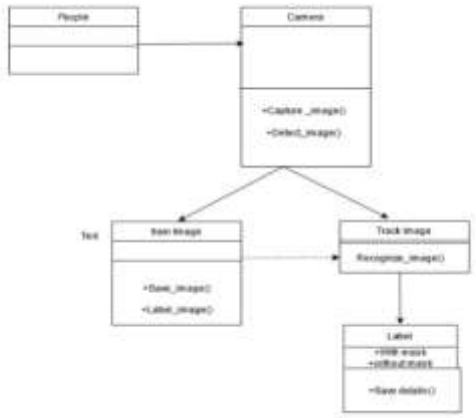
BLOCK DIAGRAM

A block diagram is a graphical depiction of a system that shows how the system works. Block diagrams assist us to comprehend the functioning of a system and aid in the creation of links within it. They are used to depict processes as well as to explain hardware and software systems.



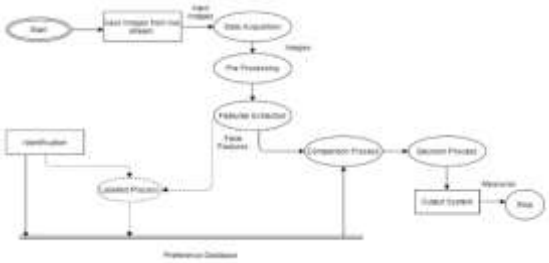
CLASS DIAGRAM

A static diagram is a class diagram. It depicts an application's static view. Class diagrams are the only diagrams that can be directly transferred to object-oriented languages and are thus commonly utilized throughout creation.

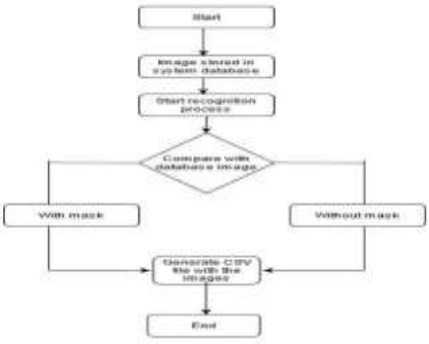


DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a classic visual depiction of a system's information flows. A tidy and clear DFD may graphically display the appropriate quantity of system needs. It can be manual, automatic, or a hybrid of the two. It demonstrates how data enters and exits the system, what alters the data, and where it is kept. A graphical tool used to define and analyze minute data with an active or automated system, such as processes, data storage, and system delays. The flow of Data Data is a critical and fundamental instrument for the construction of all other items. DFD is also known as a bubble-bubble graph or a data flow graph. DFDs are a representation of the proposed system. They should clearly and directly state the requirements upon which the new system should be constructed. During the design phase, this is utilized as a foundation for creating chart structure plans over time. The Basic Notation for Creating DFDs is as follows.



FLOWCHART DIAGRAM



EXPERIMENT ANALYSIS

MODULES

1. Using the experiments folder split, create image datasets and data loaders for train and testing.

● A training dataset is a set of data that we input into our algorithm to train our model.

● Testing Dataset: A dataset that we use to assess our model's correctness but not to train the model. It might be referred to as the validation dataset.

2.Training the model

3. Visualizing images

CODE IMPLEMENTATION

```
# import the necessary packages
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import argparse
import imutils
import time
import cv2
import os
# ### Upload Alarm Sound
# In[2]:
from pygame import mixer
mixer.init()
sound = mixer.Sound('mixkit-security-facility-breach-alarm-994.wav')
# ### Image Pre-Processing
# In[5]:
def mask_detection_prediction(frame, faceNet, maskNet):
    # find the dimension of frame and construct a blob
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),(104.0, 177.0, 123.0))
    # pass the blob through the network and obtain the face detections
    faceNet.setInput(blob)
    detections = faceNet.forward()
    # create a empty list which'll store list of faces,face location and prediction
    faces = []
    locs = []
    preds = []
    # loop over the detections
    for i in range(0, detections.shape[2]):

# find the confidence or probability associated with the detection
confidence = detections[0, 0, i, 2]
        # filter the strong detection [confidence > min confidence(let 0.5)]
        if confidence > 0.5:
```

```
            # find starting and ending coordinates of boundry box
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")
            # make sure bounding boxes fall within the dimensions of the frame
            (startX, startY) = (max(0, startX), max(0, startY))
            (endX, endY) = (min(w - 1, endX), min(h - 1, endY))
            # extract the face ROI, convert it from BGR to RGB channel
            # ordering, resize it to 224x224, and preprocess it
            face = frame[startY:endY, startX:endX]
            face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
            face = cv2.resize(face, (224, 224))
            face = img_to_array(face)
            face = preprocess_input(face)
            # append the face and bounding boxes to their respective lists
            faces.append(face)
            locs.append((startX, startY, endX, endY))
    # only make a predictions if at least one face was detected
    if len(faces) > 0:
        # for faster inference we'll make batch predictions on *all*
        # faces at the same time rather than one-by-one predictions
        # in the above `for` loop
        faces = np.array(faces, dtype="float32")
        preds = maskNet.predict(faces, batch_size=32)
    # return a 2-tuple of the face locations and their corresponding prediction
    return (locs, preds)
# ### Load Caffe Model
# Caffe (Convolutional Architecture for Fast Feature Embedding) is a deep learning framework that allows users to create image classification and image segmentation models. It is a Caffe model which is based on the Single Shot-Multibox Detector (SSD) and uses ResNet-10 architecture as its backbone. It was introduced post OpenCV 3.3 in its deep neural network module.
# In[6]:
# load our serialized face detector model from disk
from os.path import dirname, join
prototxtPath = join("face_detector", "deploy.prototxt")
weightsPath = join("face_detector", "res10_300x300_ssd_iter_140000.caffemodel")
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
# load the face mask detector model from disk
maskNet = load_model("fmd_model.h5")
# ### Face Detection on Live Camera
```

```
# In[10]:
# initialize the video stream
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
# loop over the frames from the video stream
while True:
    # grab the frame from the threaded video stream and resize it
    # to have a maximum width of 400 pixels
    frame = vs.read()
    frame = imutils.resize(frame, width=400)
    # detect faces in the frame and determine if they are wearing a
    # face mask or not
    (locs, preds) = mask_detection_prediction(frame, faceNet, maskNet)
    # loop over the detected face locations and their corresponding
    # locations
    for (box, pred) in zip(locs, preds):
        # unpack the bounding box and predictions
        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred
        if mask>withoutMask:
            label = "Mask"
            color = (0, 255, 0)
            print("Normal")
        else:
            label = "No Mask"
            color = (0, 0, 255)
            sound.play()
            print("Alert!!!")
        # include the probability in the label
        label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)
        # display the label and bounding box rectangle on the output frame
        cv2.putText(frame, label, (startX, startY - 10),cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
    # show the output frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF
    # if the `q` key was pressed, break from the loop
    if key == ord("q"):
        break
```

```
# do a bit of cleanup

cv2.destroyAllWindows()

vs.stop()
```

CONCLUSION

As technology advances and new trends emerge, we now have a revolutionary face mask detector that may contribute to public healthcare. The architecture is built around Mobile Net, which may be utilized for both high and low computation scenarios. To extract more robust features, we use transfer learning to adopt weights from a comparable task face recognition that has been trained on a very large dataset. To determine whether or not people were wearing face masks, we employed OpenCV, Tensor Flow, and NN. Images and real-time video feeds were used to evaluate the models. The model's precision has been attained, and model optimization is a continual process in which we are constructing a very accurate solution by modifying the hyperparameters. This model might be used as an example of edge analytics. Furthermore, the suggested technique produces cutting-edge results on a publicly available face mask dataset. The invention of face mask detection, which can identify whether a person is wearing a face mask and allow them admission, would be extremely beneficial to society.

REFERENCES

1. 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.
2. M. R. Bhuiyan, S. A. Khushbu and M. S. Islam, "A Deep Learning Based Assistive System to Classify COVID-19 Face Mask for Human Safety with YOLOv3," 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)
3. M. M. Rahman, M. M. H. Manik, M. M. Islam, S. Mahmud and J. -H. Kim, "An Automated System to Limit COVID-19 Using Facial Mask Detection in Smart City Network," 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), 2020
4. Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in Advances in neural information processing systems, 2014, pp. 198hy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," 2014.
5. F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on, pp. 138–142, IEEE, 1994.
6. D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," CoRR abs/1411.7923, 2014.
7. X. Cao, D. Wipf, F. Wen, G. Duan, and J. Sun, "A practical transfer learning algorithm for face verification," in Computer Vision (ICCV), 2013 IEEE International Conference on, pp. 3208–3215, IE8–1996. [2] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, "Face recognition: A literature survey," ACM computing surveys (CSUR), vol. 35, no. 4, pp. 399–458, 2003. [3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. KarpatEE, 2013.
8. P. N. Belhumeur, J. P. Hespanha, and D. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," Pattern Analysis and Machine Intelligence, IEEE Transactions on 19(7), pp. 711– 720, 1997.
9.] X. Cao, D. Wipf, F. Wen, G. Duan, and J. Sun, "A practical transfer learning algorithm for face verification," in Computer Vision (ICCV), 2013 IEEE International Conference on, pp. 3208–3215, IEEE, 2013.
10. Y. Sun, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. CoRR, abs/1406.4773, 2014. 1, 2, 3
11. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. Nature, 1986. 2, 4