Computer Networks and Communications
https://ojs.wiserpub.com/index.php/CNC

Article

FaceLite: A Real-Time Light-Weight Facemask Detection Using Deep Learning: A Comprehensive Analysis, Opportunities, and Challenges for Edge Computing

Anup Kumar Paul*

Department of Computer Science and Engineering, East West University, Dhaka, Bangladesh
E-mail: anuppaul@ewubd.edu

Received: 8 February 2024; **Revised:** 24 April 2024; **Accepted:** 28 April 2024

Abstract: The edge computing devices running models based on deep learning have drawn a lot of interest as a prominent way of handling various applications based on AI. Due to limited memory and computing resources, it is still difficult to deploy deep learning models on edge devices in a production context with effective inference. This study examines the deployment of a lightweight facemask detection model on edge devices with real-time inference. The proposed framework uses a dual-stage convolutional neural network (CNN) architecture with two main modules that use Caffe-DNN for face detection and a proposed model based on CNN architecture or customized models based on transfer learning (e.g., MobileNet-v2, resNet50, denseNet121, NASNetMobile, Inception-v3, and XceptionNet) for facemask classification. The study does numerous analyses based on the models' performance in terms of accuracy, precision, recall, and F1-score and compares all models with low disk size and good accuracy as the main priorities for memory-constrained edge devices. The proposed CNN model for facemask detection outperforms other state-of-the-art models in terms of accuracy, achieving 99%, 99%, and 99% on the training, validation, and testing, respectively, with the facemask detection ~12K image datasets available on Kaggle. This accuracy is comparable to other transfer learning-based models, and it also achieves the smallest number of total trainable parameters and, thus, the smallest disk size of all models.

Keywords: facemask, CNN, deep learning, light-weight, transfer learning

1. Introduction

Airborne infections are transmitted through coughing, sneezing, laughing, and close human contact. When a person with an infectious disease comes into close physical contact with someone who does not, the microorganism is transferred from one to the other. This is how diseases are spread. The microorganisms cling to dust particles, water droplets, and respiratory particles in the atmosphere. When a microorganism is breathed, comes into contact with mucous membranes, or is touched and its secretions remain on a surface, illness results [1].

The majority of respiratory infections in human-beings are caused by viral infectious agents. Adenoviruses, coronaviruses (e.g., types 229E, NL63, OC43, and HKU1), the cause of coronavirus disease 2019 [COVID-19], and common human coronaviruses (e.g., influenza virus, measles, mumps, parainfluenza virus, respiratory syncytial virus, and rhinoviruses) are among the causative agents. Middle East respiratory syndrome (MERS) coronavirus and highly pathogenic avian influenza viruses are additional viruses that should be particularly concerned. Human-beings experiencing a new-onset respiratory illness, including those needing hospitalization, should be evaluated for these viruses if no other cause has been found [1].

DOI: https://doi.org/10.37256/cnc.2120244439

COVID-19, also known as SARSCoV-2 and detected in 2019 has a higher reproductive number than SARS-CoV, and because of its contagious nature, the WHO has declared COVID-19 a pandemic in March 2020 [2,3]. The virus has spread more rapidly in a short period of time than any other virus. Several reasons are there for this viral transmission, such as lack of maintaining appropriate physical distance in the society, a lack of awareness about the nature of the novel virus, and, most importantly, lack of awareness to wear a facemask in public places. Different researchers have found solid evidences that tells us to wear a facemask can protect a person from corona viruses and other respiratory diseases spread by other people's droplets, as well as protect others from him [4–7] Although researchers have invented vaccines for most of these infectious viruses, they can only reduce the mortality rate and complications caused by the virus rather than exterminating it. Hence, wearing a facemask is the only way to protect an individual from getting infected by the virus and thus reducing the probability of spreading the virus to others [6]. It is highly recommended by WHO to wear a facemask in public places (inside or outside) where there is a gathering because, by wearing a facemask, one can prevent the transmission of viruses through the oral or nasal cavity [8]. Various facemasks made of cotton fabrics, surgical materials, and N95 can provide 50–95% protection against various viruses [9]. In light of the researcher's findings, many countries' governments have issued the slogan "no mask, no service" in every public service facility and have made wearing a facemask mandatory for all people. Considering the above situations, it is of immense interest to the researchers to find an effective and lightweight solution for automatic detection of a facemask. Traditional solutions such as security personnel forcefully stops other peoples who don't wear facemask to enter any premises in different situations will not work effectively. Thus, automatic detection using deep learning or machine learning is desirable [10,11].

Deep learning has been proven to be superior to other traditional machine learning algorithms in different applications, e.g., image processing, big data analysis, natural language processing (NLP), etc. [12–18]. For example, in the ISLVRC computer vision competition, a deep learning model has outperformed several machine learning models in classification, detection, and localization tasks since 2012 [19]. However, high accuracy can only be achieved during the training and testing phases at the expense of a high memory requirement and high processing costs. Training a deep learning model requires a large amount of data, and to extract useful features from that input data, the model requires a huge amount of model parameters that need to be tuned using the backpropagation algorithm along with gradient descent optimization rules. Thus, training a deep learning model is computationally expensive, and a high memory requirement is needed to store those millions of trainable parameters. High accuracy and a high memory requirement are the key characteristics of deep learning. However, a lightweight deep learning model could play a vital role in facemask detection to protect people from the viral infectious diseases. In practice, a deep learning-based facemask detection model can be deployed in surveillance systems, Internet of Things (IoT) systems, smart cities, university entrance gates, supermarkets, etc. places to ensure that all people are wearing a facemask to avoid spreading the virus to others [11].

The deployment strategy of deep learning-based applications are divided into two categories: distributed edge-based deployment and centralized cloud-based deployment. Cloud-based method is a centralized method with high processing power and resources. On the other hand, the decentralized approach with low computation power and limited resources is edge-based. Data must be moved from source edge devices (e.g., sensors, IoT, smart phones, etc.) to the cloud in order to use cloud-based resources. This imposes several challenges.

1. Scalability: With the increasing number of edge devices, sending data from edge devices to the cloud results in a bottleneck in the network's access to the cloud. Furthermore, in terms of using network resources, it is quite inefficient to upload all the data to the cloud if the deep learning model does not need all the data from all the sources. It is of great concern if the data sources are bandwidth-intensive, such as streaming video.
2. Latency: Despite the superior performance of cloud-based solutions, many real-time applications based on deep learning cannot compromise high latency for end-to-end data transfer. For many applications, real-time inference is of utmost important factor, such as rapidly processing camera frames from an autonomous vehicle to make a real-time decision about avoiding an obstacle or applications that depend on voice commands that need to be processed in real-time to understand the user's query. However, low latency requirements of these applications cannot be met using cloud-based solutions; for example, it takes more than 200 ms for an Amazon web server to process an uploaded camera frame for a computer vision task [20].
3. Privacy: Users are concerned about uploading private data such as faces, speech, and so on to the cloud because it is unknown how these data will be treated in the cloud. Also, the user's behavior can be captured in the data.

Edge computing is a feasible alternative to cloud-based solutions in terms of scalability, latency, and privacy [21–23]. Deep learning models deployed at edge devices have recently been in high demand in a variety of applications [24–26]. It is reported that the companies that have deployed edge solutions in production will increase their deployment from 1% in 2018 to 40% in 2024 [27]. According to [28], spending on edge computing will reach $250 billion in 2024. Edge computing can address scalability issues by adopting a hierarchical network architecture of edge devices, edge computing servers, and finally, the cloud server that can provide computing resources and scale as needed, avoiding a network bottleneck in the cloud data center as shown in Figure 1. Scalability issues can be addressed by putting an edge computing server in close proximity to the edge devices, which reduces the end-to-end delay and meets the real-time requirements. Privacy issues can also be addressed, as data can be processed close to edge devices without sending it via the public internet, avoiding exposure to privacy and security threats.

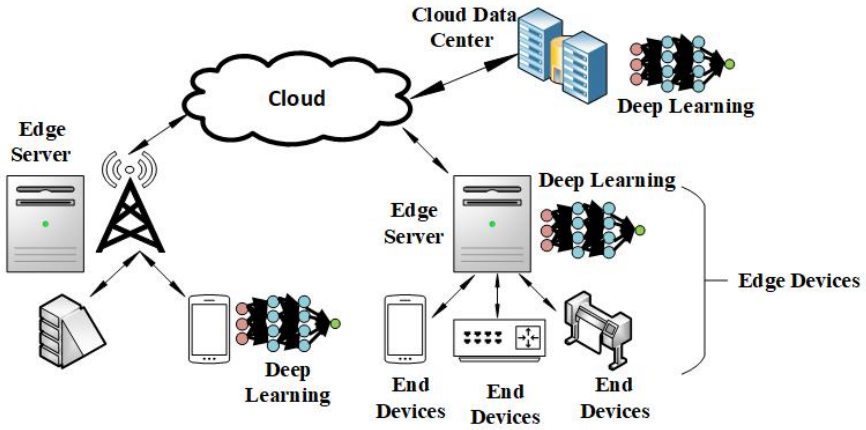


Figure 1. Application scenarios of deployment of deep learning models in edge devices, edge servers, and cloud servers.

An algorithm that detects facemask is a combination of object detection and classification that localizes and classifies the facemask in a static image or in a video stream by drawing a bounding box surrounding its extent. The algorithm of image classification detects the specific category of an image from a possible number of categories, e.g., for facemask detection, it produces whether an image belongs to the with-mask or without-mask class. The object localization algorithm finds the region of interest in the image and draws a bounding box with a certain probability around its extent to identify the masked or unmasked faces. When deep learning is applied to detect face masks, it gives the best result as compared to a traditional machine learning algorithm. To detect a facemask from an image, the algorithm must extract many useful features from the image, and the accuracy of facemask detection is based on those useful features as well as the huge amount of labeled data. A deep learning algorithm can automatically extract those useful features from an image; however, a machine learning algorithm needs handcrafted features and human supervision. The Convolutional Neural Network (CNN), which is a type of deep learning architecture specifically designed to perform image classification and detection tasks. The CNN architecture has basically two operations: the convolution operation on an image extracts useful features from it along with a pooling operation to reduce the dimensionality of the image. In CNN, these two operations constitute a layer. A CNN can have anywhere from 7 to 152 layers. The feature extractor part, also known as convolutional layers, is stacked with the classifier part, also known as dense layers (fully connected neural network), to perform an image classification task as shown in Figure 2. Some popular CNN based architectures are VGG-16 [29], ResNet-50 [30], GoogLeNet [31], MobileNetV2 [32], and so on. Among all the CNN architectures, MobileNetV2 is immensely used in facemask detection tasks as it is a lightweight classifier specifically designed for mobile devices [33,34]. It is used along with other object detectors such as "You Only Look Once" (YOLO) and "Single Shot Detector" (SSD) [35,36].

This paper focuses on facemask detection techniques and provides a narrative analysis in the framework of edge computing. The reader can clearly understand the benefits and drawbacks of each facemask detection algorithm, as well as their implementation options in resource-constrained edge devices. They can choose the suitable facemask detection algorithm according to their application needs. The key major contributions of the work are as follows:

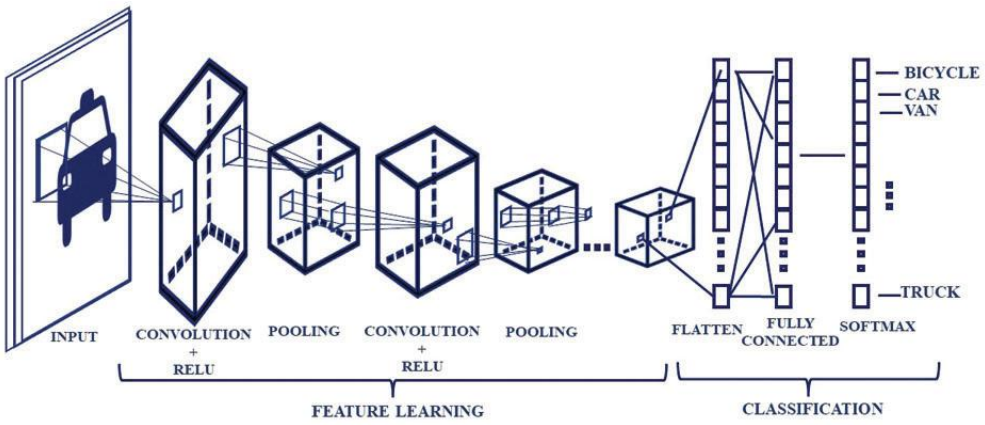


Figure 2. A typical CNN architecture for the image classification task.

- This paper provides a detailed review of deep learning-based object detection techniques and their implementation details in the context of edge computing.
- This paper also focuses on various recent literature on algorithms based on facemask detection and provides a comparison considering performance metrics and implementation possibilities in edge computing so that the researcher can find the best possibilities, identify gaps if any, and further conduct research to overcome those gaps.
- Transfer learning is applied to some of the most popular models, such as MobileNetV2, ResNet50, DenseNet121, InceptionV3, NasNetMobile, and Xception, and finetune them for the task of facemask classification.
- A lightweight CNN-based architecture (FaceLite) is proposed, which can be comparable to a lightweight transfer learning-based MobileNetV2 architecture, to classify whether a person is wearing a facemask or not. The proposed model is trained from scratch. The main advantage of the proposed model is its high accuracy and significantly reduced complexity, which makes it suitable for deployment on resource-constrained edge devices.

The rest of the paper is organized as follows: Beginning with a review of the literature on deep learning, edge computing, and edge computing optimization for various deep learning frameworks, followed by a discussion of recent research papers on face mask detection. Then different methods for fast inference for deep learning models were discussed. After that, proposed lightweight CNN model for facemask detection and different transfer- learning models were discussed and made a comparative analysis among them. Then detailed experimental evaluation results and their analysis were provided, after that the performance comparison with other state-of-the-art algorithms are discussed and finally, concluded the paper in the conclusion section.

2. Background and Literature Survey

2.1 *Deep Learning Background*

Since many of the algorithms covered in this paper are connected to the fundamental workings of deep learning, this section provides background information on deep learning. Readers interested about more comprehensive details of deep learning can refer to [37]. Deep learning architecture is based on the concept of an artificial neural network (ANN), which is inspired by and loosely connected to the biological neurons of the human brain. Although it is designed to function similarly to the human brain, an ANN consists of three layers, namely, an input layer, one or more hidden layers, and an output layer, as shown in Figure 3. Between the input layer and the output layer, one can use many hidden layers. When the number of hidden layers is large, the depth from the input to the output layer becomes deep, hence the name "deep neural network," and the learning achieved through this network is called "deep learning." When the output prediction is continuous, it is called a regression task; on the other hand, when the output prediction is categorical, it is called a classification task. For both tasks, the input data is presented to the input layer, and then a series of matrix multiplications is performed in different hidden layers of the ANN. The output of one layer is the input for the next layer. Thus, a series of forward computations is performed on the input data with the parameter matrix (also known as the weight and bias matrix) to finally get a prediction at the output layer. For supervised learning, a labelled dataset is available.

As a result, at the output layer, the predicted value is compared to the true label of the input data to compute a loss between the predicted and actual values. Depending on the loss, a backpropagation algorithm along with a gradient descent algorithm is applied from the output layer to the input layer to compute the gradient matrix, and finally, the individual learnable parameters are fine-tuned using the gradient matrix by applying a suitable optimization rule (e.g., ADAM, RMS-Prop, etc.). One forward pass using all the input data and subsequently one backward pass to tune all the trainable parameters are known as an "epoch." Using several epochs, a deep learning model can be trained and then ready for inference (testing with unseen data). A deep learning model works well when the input data has many input and output samples (supervised learning). It can also work on input data without any labelling (unsupervised learning). Different deep learning architectures exist for processing different types of input. For image and video data, CNN is the default choice. For sequence data such as natural language processing, speech signal processing, etc., a recurrent neural network (RNN) is used. In RNN, a feedback loop exists between hidden states to capture the dependencies among sequence data. A feedforward fully connected neural network architecture, as shown in Figure 3, is used for other types of data.

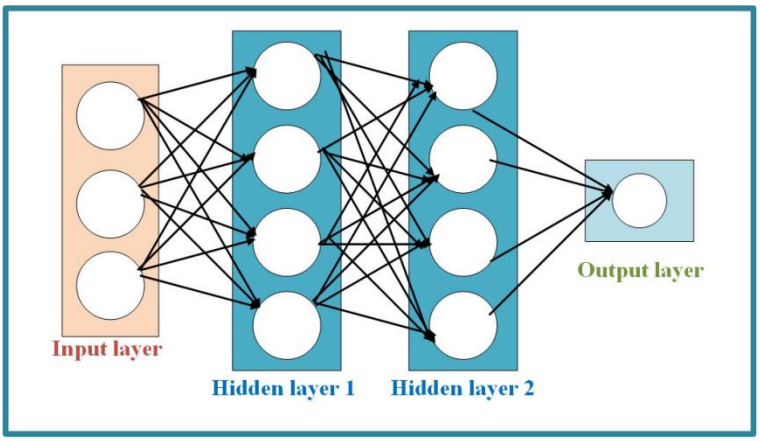


Figure 3. Artificial neural network architecture with one input layer, multiple hidden layers, and one output layer.

A key point that needs to be understood for most of the deep learning model is that a large number of trainable parameters make the model slow as it needs to compute a large matrix multiplication in every layer of the deep neural network, which eventually raises the issue of latency in edge computing. The second key point is that, in a deep neural network, careful choices of hyperparameters on how to design a deep neural architecture (e.g., number of hidden layers, number of neurons per hidden layer, different optimization parameters, etc.) that eventually make this an art rather than a science. Various design decisions lead to trade-offs among performance metrics. For example, a complex design with a large number of trainable parameters can have higher accuracy but require higher latency to compute those parameters and more disk space to store those parameters. On the other hand, a simple design will have a smaller number of parameters, which will solve the latency issues and require less disk space, but the accuracy will not be high enough for different application needs. Many research works discussed these trade-offs issues, that will be further discussed in the later sections.

2.2 *Deep Learning Performance Measurements on Edge Devices*

Selecting or designing the appropriate deep learning model for a specific application is critical as well as difficult because there are so many hyperparameter choices. A very good understanding of the trade-offs among accuracy, speed, memory requirements, energy, and other resource limitations of edge computing devices is helpful for the designer of the deep learning application. Many research papers provide a comparative performance analysis of these metrics [38]. In the context of edge computing, an important performance measurement consideration is the testbed on which the model's performance is tested. Most of the machine learning and deep learning research's primary focus is the accuracy metric, and the reported system performance is from a powerful graphics processing unit (GPU) equipped server. The author also reported the speed and accuracy trade-offs reported by the Nvidia Titan X gaming GPU in his paper [38]. Another deep learning model, YOLO, which is a real time object detection model, provides measurements on timing using the same server GPU [39].

In [40], the author specifically targeted mobile devices and measured the performance of several popular deep learning models on mobile CPUs and GPUs. Another study performed an accuracy/latency, measuring analysis on mobile devices using input data dimensionality reduction and discovered that latency is reduced at

the expense of accuracy [41]. Another deep learning model, MobileNets, which is specifically designed for mobile devices, provides system-level performance in terms of several multiply-add operations that are used for measuring latency and other performance metrics on different processing capabilities of mobile hardware [42].

Upon understanding the system-level performance, the application developer can choose the right deep learning model for a particular application. Also, some recent studies show that the automatic choice of hyperparameters for a deep learning model can be achieved by another machine learning model. For example, the authors in [43,44] combined reinforcement learning and traditional machine learning algorithms to find a way of automatically selecting the right hyperparameters for a deep learning model applicable to mobile devices that is suitable for edge computing devices.

2.3 *Common Frameworks for Training and Testing of Deep Learning Models*

As deep learning has advanced, it is reasonable and no surprise that researchers are looking for open-source software libraries and hardware that support the increasing demand for computational workloads. A number of open-source software libraries and hardware tailored to certain applications are available for deep learning algorithm testing and training on edge devices. These include Google's Tensor Processing Unit (TPU), NVIDIA's Graphical Processing Unit (GPU), Intel's Application Specific Integrated Circuits (ASICs), TinyML (a microcontroller unit for interfacing), cloud computing for training and deployment, etc., that have been specifically designed to support the heavy computational need for deep learning. Some open-source software, such as Google's TensorFlow [45], is an implementation framework that may be used on heterogeneous platforms and an interface for training and inferring deep learning algorithms. The Raspberry Pi and other edge devices can execute a deep learning algorithm that was trained with TensorFlow. Another optimized version of TensorFlow used for IoT devices, known as TensorFlow Lite [46], was proposed in 2017. To improve performance, mobile GPU support was added later in 2019. However, training a network from scratch using TensorFlow Lite is not possible. Only on-device inference is possible. It can achieve low latency by compressing the pretrained deep learning model. To expedite deep learning workloads, Google leverages GPU and TPU in their data center as part of their cloud infrastructure. Google introduced TPU support for edge applications in 2016, and it later evolved into the latest edge TPU [47], which is now extremely popular. It can provide significant power for cutting-edge machine learning while yet being energy efficient. As an example, it is stated by Google that edge TPU enables users to execute mobile versions of deep learning models such as MobileNetV2 at nearly 400 frames per second.

Another deep learning framework, Caffe [48], was created with speed, expression, and modularity in mind by Yangqing Jia's Berkeley AI research group. For industry deployment and research experiments, Caffe's speed is ideal for deep learning models. For example, with a single NVIDIA K40 GPU, Caffe can process up to 60 million images per day. That is, 1 millisecond (ms) per image for inference and 4 ms per image for training. The latest version, Caffe2, with more recent libraries and faster hardware, is maintained by Facebook. Caffe2 is recently merged with PyTorch [49], another deep learning framework that focuses on the integration of research prototypes with production deployment. The PyTorch Mobile [50] runtime beta release allows researchers to seamlessly go from training a model to deploying it on edge devices while staying completely within the PyTorch ecosystem. Some of the key features that PyTorch Mobile possesses are the support for different operating systems, such as IOS, Android, and Linux. Also, it provides APIs that cover the pre-processing and integration tasks needed for deep learning algorithms to be incorporated into mobile applications.

For efficient and faster deep learning training and inference, the GPU plays a significant role. NVIDIA's GPUs have long played a significant role in the AI sector. A GPU contains more logical cores (also known as arithmetic logic units) than a CPU, allowing it to process multiple instructions at once. NVIDIA offers an amazing range of GPUs for desktop, enterprise, and edge-class computers in addition to developer kits that facilitate edge deep learning. Most of them have characteristics like tensor cores to speed up deep learning performance, CUDA (Compute Unified Device Architecture) cores [51], and CuDNN (CUDA Deep Neural Network) libraries [52]. With the help of NVIDIA's CUDA-enabled GPU, CUDA cores are used to enable general purpose computing along with parallel processing capabilities. Alternatively, tensor cores can provide 125 TFLOPS (trillion floating-point operations per second) of optimum performance for matrix calculations in neural network training and inference, hence scaling the performance of a deep learning model. An interface for using Tensor cores in deep learning applications is provided by CuDNN libraries.

While the CUDA and CuDNN libraries are useful for training and inferring deep learning models on desktop computers, they are not designed to meet the needs of current mobile devices with limited processing capabilities, such as smartphones. To provide experimental GPU capabilities on smartphones, Android developers are currently utilizing Tensorflow Lite. Researchers are concentrating on edge-specific development

kits like the NVIDIA Jetson Nano and the NVIDIA Jetson Xavier NX instead of smartphones. Both of them work with the NVIDIA JetPack SDK, which comes with TensorRT, NVIDIA's high-performance deep learning inference engine, the bootloader, the Linux kernel, firmware and drivers, the CuDNN libraries, the CUDA toolkits, and VisionWorks (a software development package for computer vision) [53]. For edge development, the Jetson Xavier NX is the most powerful platform available, featuring a 384 CUDA and 48 Tensor core NVIDIA GPU. It can compute up to 6 TFLOPS (FP16) and 21 TOPS (INT8). Intel's Edison kit, designed specifically for IoT experimentation, is available for use with IoT devices [54].

2.4 *Background on Facemask Detection Algorithm*

The facemask detection algorithms have been thoroughly discussed along with their features, performances, and limitations in this section. For facemask detection, CNN is used by most of the researchers because CNN architecture has been shown to be superior in image classification tasks. Other researchers went for hybrid approaches that used a combination of machine learning (ML) and deep learning (DL)-based algorithms. Figure 4 depicts a general flowchart of the working principle of the facemask detection algorithm. From the figure, it can be seen that the process of facemask detection is a combination of object detection inside an image or video frame and then classifying the image as being either with or without a mask.

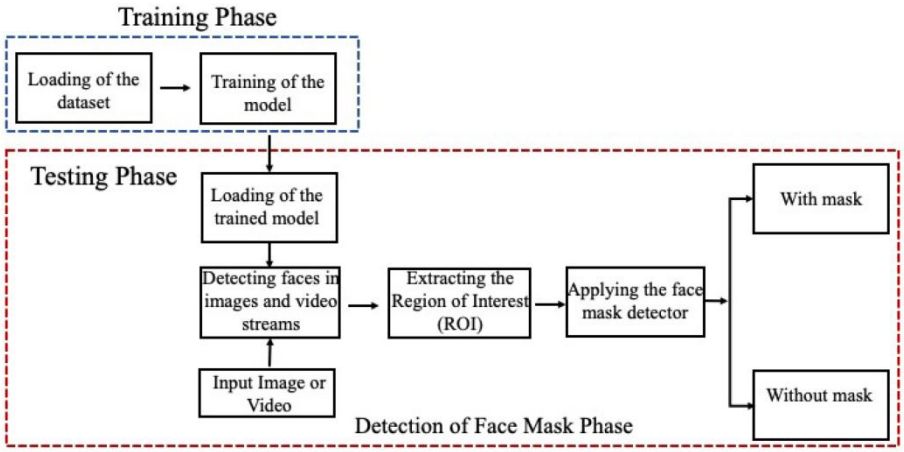


Figure 4. Fundamental framework for facemask detection algorithms.

2.4.1 Object Detection

Object detection is an algorithm for detecting an object of interest from an image or from a video frame using computer vision. It has brought a revolutionary change to the image processing domain. It can detect multiple objects inside an image and categorize them as different objects, such as a human face, cats, cars, a brain tumor, etc., within a split second, and its application domain is expanding day by day. There are two basic learning algorithms such as object classification and localization exist in deep learning, and are the key components of an object detection algorithm. Object localization determines an object's location inside an image by creating a bounding box surrounding the object. On the other hand, object classification attaches a label to an image. Face detection is a highly researched object detection category by researchers due to various application demands. Traditional object detection methods such as the Viola-Jones detector [55], the Histogram of Oriented (HOG) detector [54], the Scale Invariant Feature Transform (SIFT) detector [56] and the Deformable Part-Based Model (DPM) [57] detector can detect multiple objects from an image, but these algorithms have the major drawback that they do not consider the occlusion and have poor generalization ability and robustness. Because of these limitations, DL-based algorithms have become the mainstream research method for object detection. DL algorithms are capable of learning contextual information and complex features as well as efficiently handling occlusion. Compared to object detection, facemask detection is a bit of a complicated process because it is more difficult to extract features from a masked face than it is to do so without a mask. Every face mask detection algorithm maintains two stages for face mask detection. In the first stage, faces are detected from the image, and then in the second stage, the feature is extracted to classify between with-mask and without-mask faces. Since CNN architecture has been proven to be superior at extracting features from an image, it is widely used for face mask detection.

Object detection algorithms using the CNN architecture have been categorized into two-stage and one-stage detectors. A two-stage detector first determines and processes a region of interest (ROI) from the image

using a search algorithm. Then, from each ROI, a CNN feature vector is extracted individually. Some popular two-stage detectors are region-based CNN (RCNN) [58] fast RCNN [59], and faster RCNN [60]. A two-stage detector can achieve high precision but suffers from poor real-time performance. On the other hand, a one-stage detector is based on the concept of regression, directly predicts the coordinates of the ROI, and categorizes the target in a single step. They bypass the stage of region detection consumed in a two-stage detector. This type of detector is fast enough to meet the real-time requirement at the expense of low precision. Some examples of one-stage detectors are You Only Look Once (YOLO) [61] and its variants [39,62]. Single Shot Multibox Detector (SSD) [63], RetinaNet [64], etc. YOLO can achieve the best performance almost in real-time. SSD typically yields excellent results when utilized for object detection. The foundation of RetinaNet is a feature pyramid network.

2.4.2 Facemask Detection

The CNN architecture was used by the majority of existing algorithms for facemask detection because CNN extracts complicated features with exceptional performance. from images using its convolution and pooling operations to help classify the image correctly. Some popular CNN models that are used for facemask detection are MobileNetV2, ResNet50, DenseNet, VGG-16, NASNet-Mobile, etc.

Fan et al. [11] proposed a DL-based single-shot lightweight facemask detector for the embedded devices. They have used a depth-wise separable convolutional network based on MobileNet as its backbone network and a feature pyramid network to fuse low-level layers with high level information. To cope with the problem of using a feature pyramid network to extract complex features, they have also proposed a residual context attention module (RCAM) and a synthesized Gaussian heatmap regression to focus on extracting the complex features related to the facemask region. Evaluations on two publicly available datasets reveal that their proposed model achieved a mean average precision (mAP) that is higher by 1.7% on the Moxa3K [65] dataset and 10.43% on the AIZOO dataset [66] as compared to the YOLOv3-tiny [62] model. However, this method requires extra computation to generate the Gaussian heatmap, and due to the limitations of the dataset, the model is unable to discriminate between masks worn appropriately and incorrectly. In order to differentiate between wearing a mask, not wearing a mask, and wearing a mask incorrectly, Kocacinar et al. [34] devised a two-stage deep mobile system. They have used the MobineNet, VGG-16, and ResNet models as the base models and the transfer learning technique to classify among the three classes.

Albalas et al. [67] proposed a model that fused graphs and CNN architecture. A geographical similarity of facial nodes is measured using a distant graph, and then to compute the correlation between any two facial nodes, the correlation graph is formulated. Transfer learning is also employed and finally, discriminant graph convolutions are constructed by fusing the distant and correlation graphs. Experimental results show that their proposed model can achieve 98% accuracy on two-class classification and 86% accuracy on three class classification using MAFA dataset [68]. Jiang et al. [69] proposed a model called RetinaFaceMask. The RetinaFaceMask model is divided into three parts. The backbone network, the neck network, and the head network are all interconnected. In the backbone network, they used ResNet as the backbone network to extract features from the image. For the neck network, they employed a feature pyramid network for extreme accuracy. A detector or classifier makes up the head network in which they have employed a context attention module to increase the precision of classification. Because of the limited dataset, they also utilized the transfer learning strategy. However, because of this complex CNN network architecture, the computation overhead is large.

YOLOv2 and ResNet-50 were combined to create a facemask detection model that was proposed by Loey et al. [70] YOLOv2 is an updated version of YOLO and is a feature extraction and classification algorithm. ResNet-50 is a lightweight residual network of the original ResNet with only 50 layers. They have used two datasets, namely the facemask dataset [71] and the medical mask dataset [72], to train and test their model. They also used a data augmentation strategy and to improve the model estimation of the anchor boxes are utilized. There were two optimizers used: SGDM and ADAM in their research to obtain a comparative result between these two optimizers. The average accuracy is only 81%, and their model cannot be used to identify masked faces from videos.

Face detection using only a traditional ML-based algorithm is not a feasible solution. Therefore, some researchers combine the DL-based algorithms withML-based classifiers (support vector machines, decision trees, etc.) to obtain better results. Ristea et al. [73] proposed a model to detect a facemask from a speech signal. Their suggested model is split into two main sections: (i) using cycle consistency loss to train generative adversarial networks (GANs) to distinguish utterances between two classes (mask- and mask-free); and (ii) for each transformed pronunciation, they have assigned two opposite levels using cycle consistency GANs. A ResNet network with varying layer depths received the original and altered accents as input in the form of spectra. All

the outputs of different ResNet networks were combined and fed to the SVM classifier to predict the final result. The model requires a long processing time due to its complex architecture. They compared their results with and without data augmentation and reported an accuracy of only 74.6%.

A facemask detecting system was proposed by Nieto-Rodriguez et al. [74] to sound an alert when medical staff members enter the operating room or medical room without wearing a surgical mask. For face mask detection, they used two detectors (one for face detection and the other for mask detection) and two color filters (one for each detector). They have used the traditional Viola-Jones detector as the face detector and a variant of AdaBoost called LogitBoost [75] for detecting face masks. However, it has some problems. For instance, because it relies on two color filters, any clothes in close proximity to the mask area may produce inaccurate results. By applying synthetic rotation [76], this problem is solved. Because this model is only trained using surgical masks and the traditional AdaBoost algorithm, its performance is not as good as that of other DNN models.

Snyder et al. [77] proposed a facemask detection system that can detect unmasked personnel in the public area using a robot called Thor. They have utilized three deep learning modules in the robot to achieve this task. For feature extraction from the images, they have combined the ResNet model and feature pyramid network in their first module. They used multi-task CNN (MT-CNN) to detect faces from human subjects in the second module. MT-CNN is used as the face extractor from the video frame introduced by Zhang et. al. [78] and is widely used in face detection-based research [79]. Then in their third module, they have constructed a neural network-based model to classify between masked and unmasked faces. They have evaluated their proposed model using a dataset collected by the robot from public places. The images were taken from distant locations and at various angles. Because of these challenging scenarios, they reported an F1-score of 87.7%.

A summary of the facemask detection techniques including the models, datasets, results, and areas for future study is depicted in Table 1.

Table 1. A summary of the facemask detection techniques including the models, datasets, results, and areas for future study.

Literature	Used Model	Dataset	Advantage	Disadvantage
[11]	RCAM, MobileNet-V2	AIZOO [66] Moxa3K [65]	Single Shot Lightweight Model Applicable to embedded device.	Extra computation to generate Gaussian Heatmap. Identification between correctly and incorrectly worn mask is low due to limitation of dataset.
[69]	RetinaFaceMask	MAFA-FMD [69]	Accuracy is high.	Computational Overhead is large.
[70]	YOLO-v2, ResNet-50	MAFA-FMD [71], MMD [72]	Data Augmentation is used to enhance the quality of dataset. Two optimizers are used for comparative results.	Relatively small dataset. Accuracy is low. Cannot be used for video stream.
[34]	MobileNet, VGG-16, ResNet.	Custom made dataset from 5 datasets.	Real time facemask detection. Applicable to edge devices.	Accuracy is low. Dataset is small.
[67]	Graphs, CNN	MAFA [68]	High accuracy on two class classification.	Low accuracy on three class classification.
[77]	ResNet, MT-CNN	Custom made dataset collected by robot.	Incorporate challenging scenarios.	Accuracy is low.
[33]	SRCNet	MMD [72]	Increased the resolution of the images using the SR network. High accuracy. Light-weight model.	Small dataset. Cannot be used for video stream.
[68]	LLE-CNN	MAFA [68]	Large and diverse dataset. Robust classification.	False detection of occlusion regardless of facemask. Sideface orientation affects the accuracy.

3. Materials and Methods

The research work uses a dataset (facemask detection ~12K dataset) from Kaggle that has been released in the public domain under the Creative Commons Zero (CC0) license, allowing the use of the dataset without any copyright restrictions. For testing and demonstration purposes, the corresponding author himself was the participant and used his own pictures.

3.1 *Dataset*

The model is trained with a facemask dataset [80] from Kaggle. The dataset includes 12000 images of people wearing or not wearing masks in indoor and outdoor settings. All the images with face masks (6000 images) are scraped from Google, and all the images without face masks (6000 images) are pre-processed with the CelebFace dataset [81]. Therefore, the dataset is balanced. The dataset is divided into train, test and validation sets. The data distribution graph is shown in Figure 5. In the dataset, there are various types of images. The characteristics differ greatly in terms of illumination, occlusion, scale, and poses. For instance, the face can take up a significant amount of space in certain photos while taking up very little space in others. Faces are also occluded by some objects and lie in different locations in the image (e.g., the center, left, right, corner, etc.).

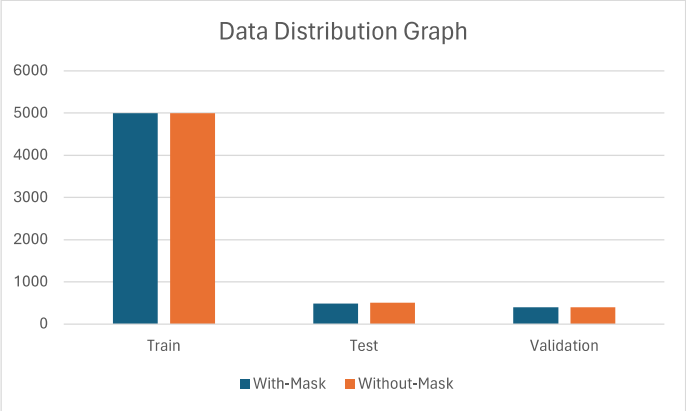


Figure 5. Data Distribution

3.2 *Pre-processing the Dataset*

Before training the model, a set of pre-processing steps are applied to the image data. The mobile device input requirement is (224 x 224) pixels. Thus, the images have been down sampled to (224 x 224) pixels. In order to guarantee that the extracted features have an equal distribution range, the images have also been rescaled to limit the pixels in the range of [0, 1]. Normalization and standardization are then applied using mean subtraction and division by standard deviation. The dataset has been augmented (rotated, resized, zoomed, flipped, etc.) to ensure that the model can distinguish the masks and identities of persons in a variety of situations without being overfitting or underfitting on the dataset. By preprocessing the facial images in the recommended manner, the trained model can more accurately identify between different angles and perspectives of similar faces.

3.3 *Software Specifications*

To implement and deploy the model, TensorFlow and Keras [82] have been used as the main tools. TensorFlow is a deep learning framework supporting both low-level and high-level APIs. It supports CUDA, Python, and C++. Google Brain created TensorFlow to handle many machine learning tasks on a single platform. and provide various high-speed processing units such as a normal CPU, a high-speed Graphical Processing Unit (GPU), and Tensor Processing Unit (TPU) support. Keras is a high-level neural network API written in Python that runs on top of TensorFlow. TensorFlow and Keras provide a high-level API to support custom-made models as well as pretrained transfer learning models to solve a specific problem.

3.4 *Light-Weight Facemask Detection Model*

Here, proposed framework's fundamental architecture for face and mask recognition has been discussed. Two key modules make up the architecture of the two-stage framework. The first module uses the Caffe-DNN module to do face detection at the initial stage, and the second module uses the proposed lightweight CNN model (FaceLite) as shown in Figure 6 to accomplish face mask recognition. Six unique CNN models based on transfer learning are also included in the second stage.

- Face Detection Module: Caffe model [48] which is based on the Single Shot-Multibox Detector (SSD) and uses ResNet-10 architecture [30] as its backbone to detect faces from the image has been used. For

conducting object identification and recognition, Caffe models were developed as an efficient and rapid replacement for previous frameworks.

- Facemask classification Module: The proposed CNN model for classifying face masks has been used in the second stage, which involves face mask identification. Additionally, in order to create customized models, transfer learning has been used to develop six pretrained CNNs: MobileNet-v2 [32], Inception-v3 [83], ResNet-50 [30], DenseNet-121 [84], Xception [85], and NASNet-Mobile [86]. The input for each of these models is the Region of Interest (RoI) obtained from the face detection module, and their classification-based results include face mask recognition. The architecture of the suggested CNN model and six customized models based on transfer learning are covered in the section that follows.

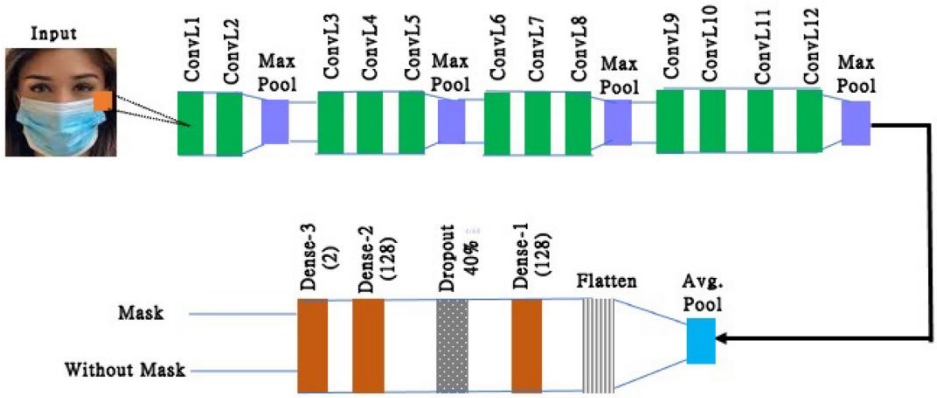


Figure 6. Proposed CNN-based Light-weight Facemask Detection Model.

3.4.1 Proposed Light-Weight CNN Model

An image classification CNN for the face mask identification job has been developed, allowing for the categorization of input images without reliance on pre-trained models. The goal is to create a simple, lightweight model that will help find patterns connected to every group of images and lower the dimensionality of the images.

Because Keras' functional API is more flexible than Sequential API, the custom model in TensorFlow has been constructed. It is a method for creating layer graphs that enables the addition of new layers to directed acyclic graph (DAG). The initial size of the input image to the network in the model definition is 224 x 224 pixels with three channels. A color image that has been scaled to 224 x 224 x 3 pixels for training and testing purposes is the input for the proposed model. Figure 6 shows the general architecture of the model.

The model's architecture is made up of ten convolutional blocks, each of which has a convolutional layer and a ReLU layer as the activation function. Max-pooling layers have been placed carefully after 2, 3, or 4 successive convolutional layers. 16 filters of size 3 by 3 are used in the first two convolutional blocks. Each filter moves across the entire image and, using the ReLU activation function, outputs a unique 2D activation or feature map. The resulting volume's spatial dimensions are subsequently reduced by applying size 2 by 2 max pooling. Convolutional blocks number two, three, and four were added; each of these blocks had 32 filters layered on top of it. After that, the input dimension was reduced once again by using the max-pooling layer. Then another three convolutional layers were followed by another max-pooling layer, and finally four successive convolutional layers were followed by a max pooling layer. Just before connecting the output of the convolutional layer to the fully connected layer, an average-pooling layer was deployed to reduce the dimension of the feature map further while retaining the important features extracted by the filters. To extract more features, an increasing number of filters were used as the model went deeper into the architecture.

The output feature maps of the last convolutional layer must be transformed into one-dimensional array to produce the prediction. In the model, there is a flatten layer that takes a multidimensional output and linearizes it such that the dense layer may accept it as input. Then two dense layers were included, each having 128 neurons are called fully connected (FC) layers. To prevent the network from over-fitting, a dropout layer (dropout ratio value of 0.4) was included after the initial FC layer. More FC layers provide larger coverage for the entire spatial dimension of the image and improve interpretation between the features retrieved by the convolutional blocks and the predictions.

Finally, a dense layer with two output neurons was created with SoftMax as the activation function. As a result, the number of output nodes in our final dense layer equals the class numbers. The target class probabilities are produced, with a range of values from 0 to 1, and the aggregate of all values being 1. The

detailed architecture with number of trainable parameters along with the filter size, stride, padding, output image size after every convolutional and maxpooling operation are shown in Table 2.

Table 2. Details of the proposed FaceLite model architecture.

Layer Type	Output Shape	Number of Parameters
Conv2D	(222, 222, 16)	448
Conv2D	(220, 220, 16)	2320
MaxPool2D	(110, 110, 16)	0
Conv2D	(108, 108, 32)	4640
Conv2D	(106, 106, 32)	9248
Conv2D	(104, 104, 32)	9248
MaxPool2D	(52, 52, 32)	0
Conv2D	(50, 50, 64)	18496
Conv2D	(48, 48, 64)	36928
Conv2D	(46, 46, 128)	73856
MaxPool2D	(23, 23, 128)	0
Conv2D	(21, 21, 128)	147584
Conv2D	(19, 19, 128)	147584
Conv2D	(17, 17, 128)	147584
Conv2D	(15, 15, 256)	295168
MaxPool2D	(7, 7, 256)	0
AvgPool2D	(1, 1, 256)	0
Flatten	256	0
Dense	128	32896
Dropout (40%)	128	0
Dense	128	16512
Dense	2	258
Total Trainable Parameters		942770

Overall system architecture of the proposed system is shown in Figure 7. The proposed system is a two step system where first the proposed facemask model is trained using preprocessed image data. Once training is complete, the model is saved to the disk for later use. In real time face mask detection from video or image data, first the face is detected using a face detector with the help of OpenCV and resNet-10-ssd-Caffe model. The region-of-interest (ROI) that is face is extracted and detected from the image and then fed into the facemask detector module as an input. Then the proposed trained model faceLite is applied on the detected ROI to classify between with-mask and without-mask image.

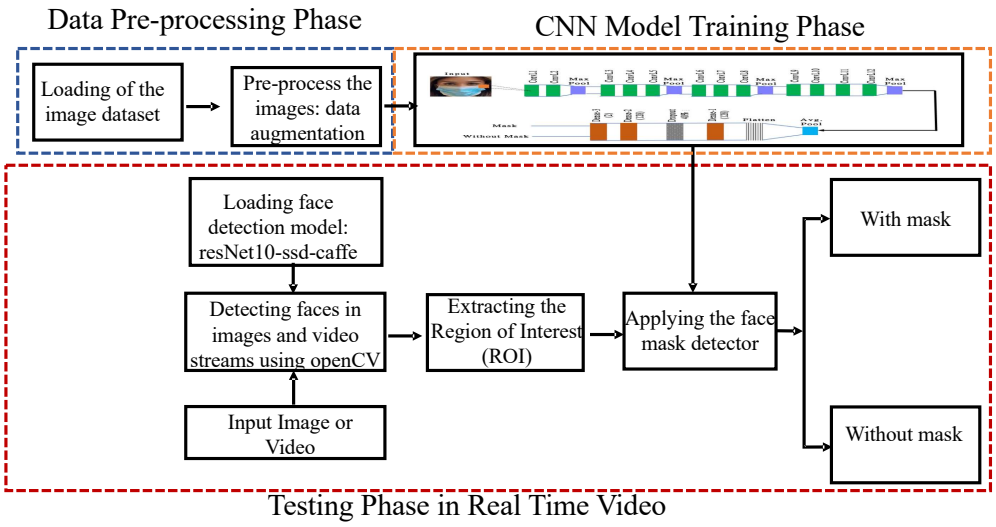


Figure 7. Proposed System Overview of Facemask Detection Model.

3.4.2 Customized Model Based on Transfer Learning

In the transfer learning methodology, a previously trained model on a large dataset is reused as the base model on a new related task. The main advantage of using transfer learning in classifying medical image data is that if a model is trained on a large dataset, one can achieve high accuracy with the same model (after fine tuning) even on a small dataset. In transfer learning, there is no need to train the entire model from the beginning to fit the model on a dataset. Rather, the base convolutional layer already contains optimized parameters that are useful for image classification. However, the top layers of the model are specific to the image classification category and subsequently specific to the old classification task. Thus, it is needed to unfreeze a few top layers of the model, add a few dense layers to match the new classification category, and finally train the newly added dense layers along with the classification layer at the output of the model. In this way, fine-tuning the higher-order feature representations of the image of the base model to make them more closely connected to the classification of the new task. In this paper, six different base models were used such as MobileNet-v2, Inceptionv3, Xception, ResNet-50, NASNet-Mobile, and DenseNet-12 and applied transfer learning methodology to classify faces with and without masks.

- **MobileNet-V2**: The lightweight architecture of MobileNetv2 is designed to function well on embedded mobile devices. Its foundation is an inverted residual structure with thin layers serving as bottlenecks at the residual block's input and output. It applies depth-wise separable convolutions to each color channel to extract the features. By reducing the non-linearities in n-array layers, it keeps the representational power. There are two basic building blocks in its architecture. For shrinking, the first and the second residual block has a stride of 1 and 2 respectively. The two blocks have three layers each. A depthwise convolution is used for the first layer, a 1x1 convolution using ReLU6 [87] for the second, and a linear 1 x 1 convolution for the third. The fundamental idea is to employ bottlenecks to encode inputs and outputs while using the inner layer to encapsulate the transformation from pixels to image categories.
- **ResNet-50**: ResNet-50 is a significantly more complex CNN-based architecture, but instead of using FC layers, it uses global average pooling and that's why its size is noticeably reduced. Without impacting the model's performance, it trains deep layers using skip connections and batch normalization. The issue of vanishing gradients makes it challenging to train deep CNNs. ResNet-50, on the other hand, offers a solution by offering skip connections, also referred to as gated units. It is feasible to train the 152-layer model with less complexity than the VGG-16 thanks to these gated recurrent connections [26]. Rather than learning the straight mapping, it applies residual functions for a small number of stacked layers. It can be trained more quickly as compared to VGG-16 because it has 23 million parameters. With one significant exception, Resnet-34 is the foundation of the ResNet-50 architecture. That is a modified bottleneck design building block that uses a stack of three levels rather than two. The skip connections don't have additional parameters, which reduces computing complexity and enables the transfer of important information from one layer to the next.
- **Inception-V3**: Inception-V3 concentrates on the less powerful computing resources by introducing the concept of factorized convolutions. It seeks to cut back on parameters without adversely compromising the network's effectiveness. It folds the data into convolutions by substituting big convolutions for smaller ones in order to do this. For instance, replacing one 5 x 5 convolution (5 x 5 = 25) with two 3 x 3 convolutions (3 x 3 + 3 x 3 = 9 + 9 = 18) reduces the parameter from 25 to 18. It also performed factorization into asymmetric convolutions in a comparable way. It also features an additional classifier that functions as a regularizer.
- **Xception**: "Extreme inception" is what Xception stands for. To enhance multi-scale feature extraction, the Inception model serves as inspiration. The improvements from ResNets and Inception are combined. The concept of depth wise separable convolution served as the foundation for the design of the Xception model. The Xception model consists of three main sections: the main flow, the middle flow (which consists of eight repetitions of the same block), and the exit flow. With the exception of the first and last modules, it has 36 convolutional layers that are organized into 14 modules with linear residual connections. Following all convolution and separable convolution layers is batch normalization.
- **DenseNet-121**: The basic CNN design has been modified to create the DenseNet-121 architecture. One 7 x 7 convolution, 58 3 x 3 convolutions, 61 1 x 1 convolutions, four average pooling layers, and one fully connected layer make up the DenseNet-121 architecture. The key elements of its architecture include bottleneck layers, growth rate, connectivity, and dense blocks. The CNN's next convolutional layer receives the feature map from the preceding layer. In this way, CNN provides L direct connections for L levels. On the other hand, the core idea of DenseNet-121 is the concatenation of feature maps from previous layers so that there are L(L+1)/d direct connections for each L layer. The DenseNets utilize the potential

representational power through feature reuse and solve the vanishing gradient problem instead of using highly deep or wide architecture.

- **NASNetMobile**: NASNetMobile is short for Neural Architecture Search (NAS) Network. NASNetMobile is a scalable CNN architecture based on reinforcement learning, made up of cells that serve as the fundamental structural components. Cells are used for operations like pooling and separable convolution. Depending on the capacity of the network, these procedures are repeated. There are 5.3 million trainable parameters and 12 cells in the NASNet mobile architecture.

4. Experimental Results and Performance Analysis

The experiments were caried out on Tensorflow [45] and Google Colaboratory [88] notebooks using Python and the Keras [89] library to evaluate the proposed model and transfer learning-based customized models. The model was trained over 50 epochs. The batch size was set at 32, and the loss function was optimized using the ADAM optimizer with a learning rate of 0.0001 and decaying across the epochs. The base model was built within the framework of the suggested system, and subsequently included this model in a mobile environment.

4.1 *Performance Evaluation Metric*

Accuracy, precision, recall, and F1-score were the four common evaluation metrics utilized to compare the performance of the proposed model and the customized model based on transfer learning on the with-mask and without mask datasets. In order to describe the evaluation measures, the terms "True Positive (TP), "False Positive (FP)," "True Negative (TN)," and "False Negative (FN)" were first defined. Suppose that the two classes (with-mask and without-mask) in the dataset are referred to as the "positive" and the "negative" classes, respectively, for a binary classification issue. Then TP refers to those examples being correctly identified as positive examples belonging to a positive class. Examples that should have been categorized as negative but were instead placed in the positive category are referred to as FP. An example belonging to the negative class and correctly categorized by the model is referred to be TN. FN stands for an example that belongs to the positive class but is mistakenly categorized as belonging to the negative class. The evaluation metrics can be defined as follows:

$$Accuracy_{class(i)} = \frac{TP_{class(i)} + TN_{class(i)}}{TP_{class(i)} + TN_{class(i)} + FP_{class(i)} + FN_{class(i)}} \tag{1}$$

$$Precision_{class(i)} = \frac{TP_{class(i)}}{TP_{class(i)} + FP_{class(i)}} \tag{2}$$

$$Sensitivity_{class(i)} = \frac{TP_{class(i)}}{TP_{class(i)} + FN_{class(i)}} \tag{3}$$

$$F1 - Score_{class(i)} = \frac{2 \times Precision_{class(i)} \times Sensitivity_{class(i)}}{Precision_{class(i)} + Sensitivity_{class(i)}} \tag{4}$$

4.2 *Experimental Results*

Using evaluation matrices, the experimental findings have been thoroughly discussed in this section. The experimental findings have been collected from still images. Additionally, a mobile device's camera is used to record live videos to gather the experimental data. Two subsections make up the explanation of the results and in-depth discussion. In the first subsection, the experimental results of the suggested model on a test dataset were shown, and in the second, the experimental findings of the face detection module were discussed in conjunction with the Caffe-DNN module to identify facemasks in real-time video images.

4.2.1 Facemask Identification

The model's training and validation results for the facemask identification task are shown in Figure 8 and the confusion matrix is shown in Figure 9. The Figure 8 shows that during the training and validation phase, the accuracy rises steadily to nearly 99%. The model's continued excellent training and validation accuracy

demonstrate that neither overfitting nor underfitting might be affecting the model's capacity for generalization. Furthermore, in a constrained number of training epochs, the model can rapidly learn and converge. Additionally, during the training phase, the misclassification rates shown by the loss function are incredibly low, averaging approximately 0.015, and increase to approximately 0.016 during the validation phase. These outcomes can also be realized by the confusion matrix shown in Figure 9. From the confusion matrix, we can see that the proposed model FaceLite predicts only 0.22% images as without-mask images (false negative) and only 2.29% images were misclassified to with-mask images (false positive).

Figure 8. Training loss and validation accuracy curve of FaceLite Model

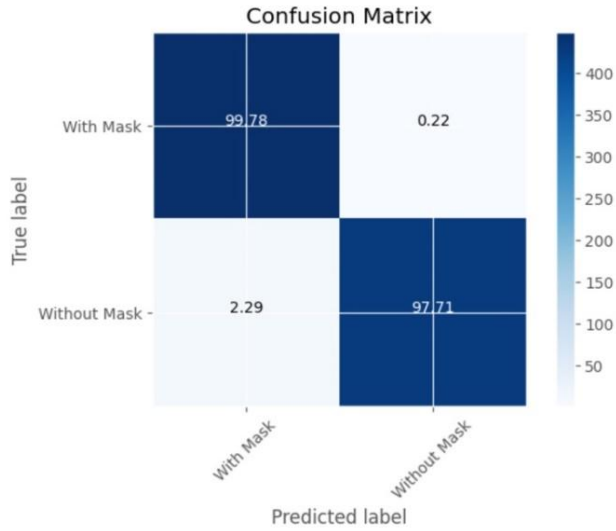
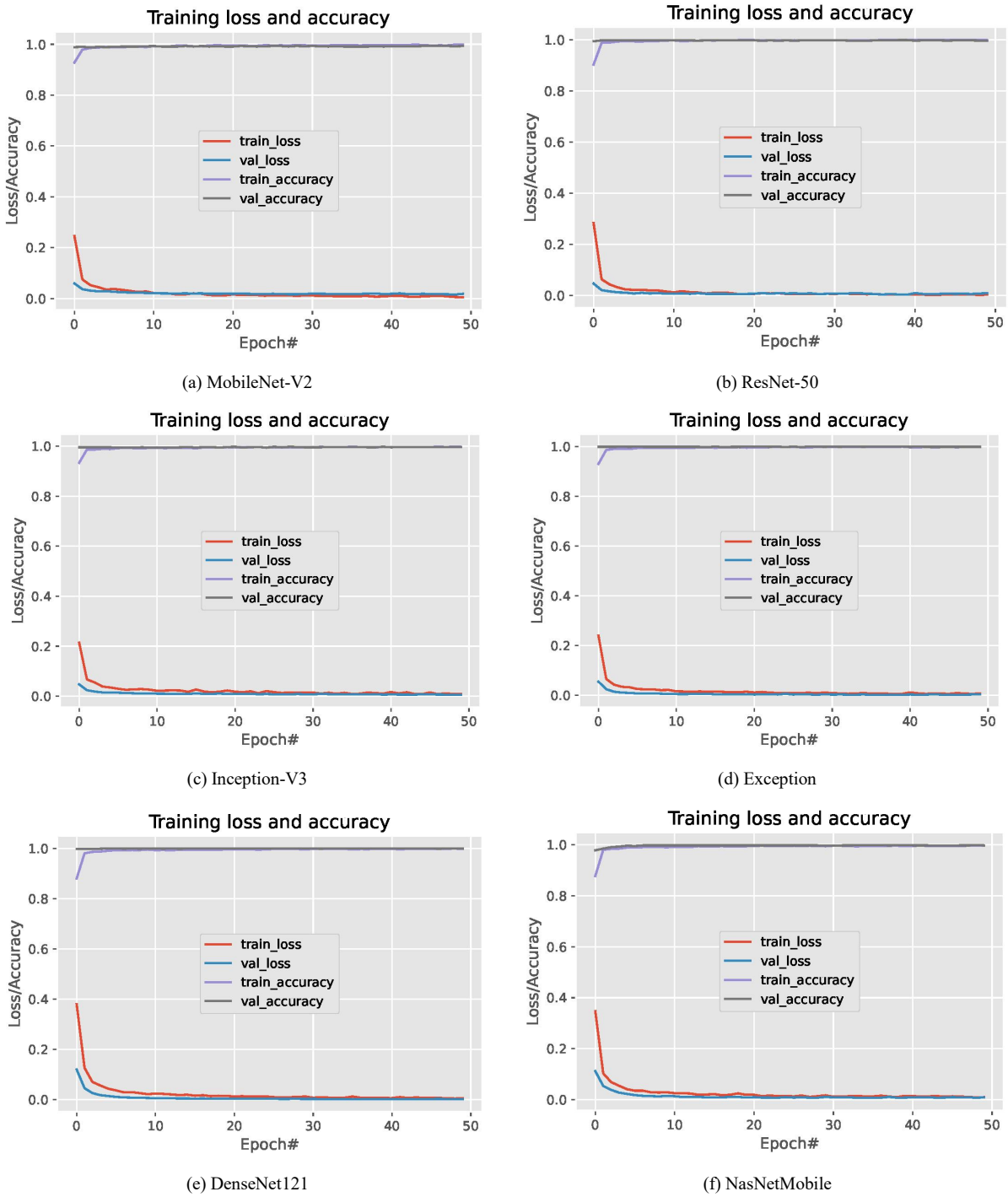


Figure 9. Confusion matrix of FaceLite model

The training and validation results for the transfer learning models are shown in Figure 10. Transfer learning models' training and validation accuracy are nearly perfect. The sophisticated architecture of the transfer learning model is the key to achieving such high accuracy. It is also obvious from the confusion matrix, as shown in Figure 11. All the models' false-negative and false-positive rates are very low. Among all the transfer learning-based models, MobileNet-V2's total training parameters are the lowest and thus have slightly higher false-positive and false negative rates. Thus, it is obvious from these results that, to achieve high accuracy, a trade-off must be made to increase the number of parameters and thus the model's disk size. In that respect, it can be claimed that the FaceLite model achieves significantly higher accuracy while keeping the number of trainable parameters the lowest among all the other models. This is a highly desirable characteristic a lightweight model should possess to be applicable to edge computing devices. To describe and choose the best discriminating descriptor of facial important features, the proposed model, FaceLite, generated fewer than 1 million parameters with a 99% total accuracy. Furthermore, the proposed model's ability to distinguish both masked and unmasked faces is demonstrated by the outstanding detection accuracy as measured by recall,

precision, and F1-score. The test dataset results demonstrate the proposed model's exceptional accuracy in predicting masked faces, with F1-score, precision, and recall all reaching values of 0.99, 0.99, and 0.99, respectively. Additionally, it achieves high accuracy in predicting unmasked faces in the test dataset, with precision, recall, and F1-score of 0.99, 0.99, and 0.99, respectively. To achieve this accuracy, the proposed model's disk size is only 11.45 MB which is the lowest among all other models' disk sizes as can be seen from table 3. This paper's aim was to build a lightweight model applicable to edge devices while retaining high accuracy.



(a) MobileNet-V2

(b) ResNet-50

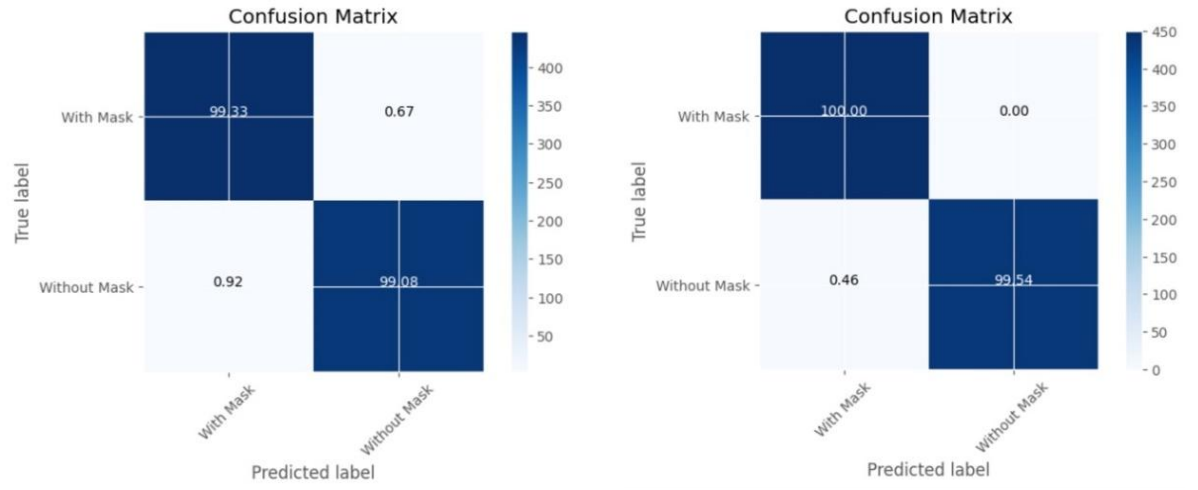
(c) Inception-V3

(d) Exception

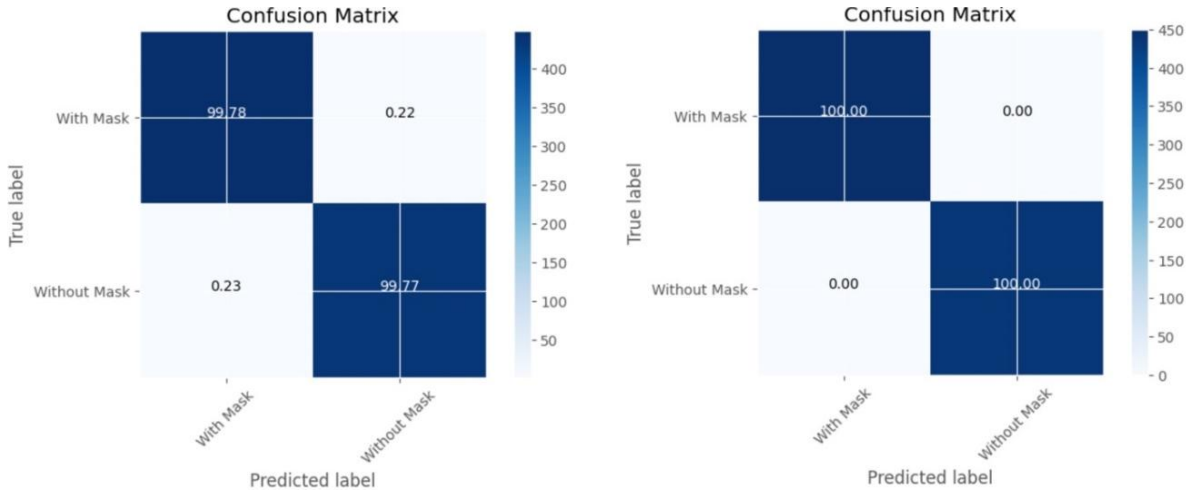
(e) DenseNet121

(f) NasNetMobile

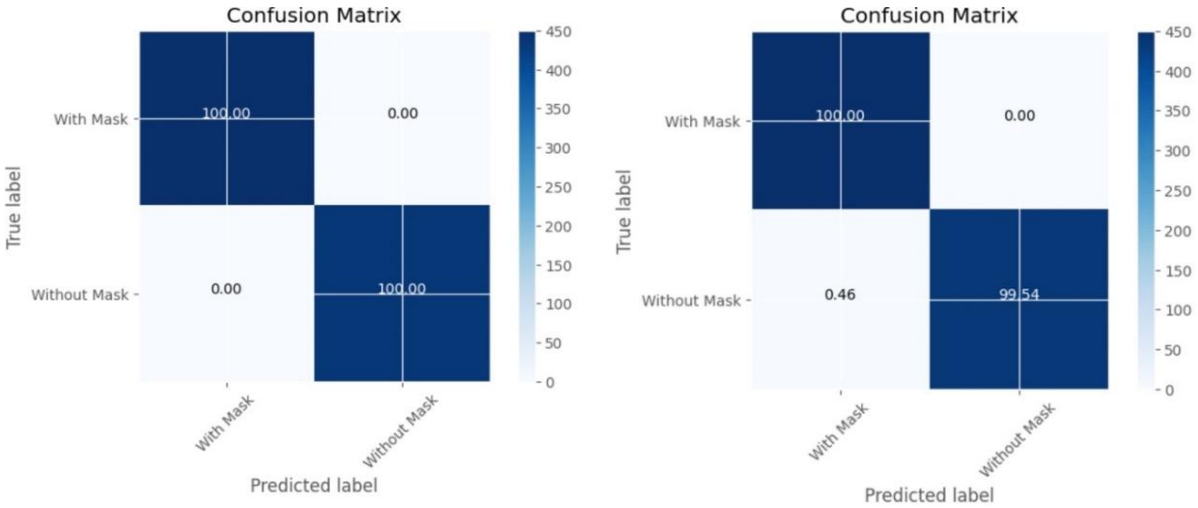
Figure 10. Training loss and validation accuracy curve of various transfer learning models.



(a) MobileNet-V2 (b) ResNet-50



(c) Inception-V3 (d) Exception



(e) DenseNet121 (f) NasNetMobile

Figure 11. Confusion matrix of various transfer learning models.

As a result, the model can detect masks accurately even when they have a variety of colors, styles, shapes, and coverage regions. Therefore, based on testing results, it can be said that the suggested model has a 99% accuracy rate in identifying masks. A overview of the performance outcomes derived from the confusion matrix is shown in Table 3 and attained by various transfer learning-based models in addition to the proposed model.

Table 3. Performance characteristics of various facemask detection models

Model	Class Category	Precision	Recall	F1-Score	Accuracy	Total Parameters	Model Size (MB)
FaceLite	With Mask	0.99	0.99	0.99	0.99	9,42,770	11.45
	Without Mask	0.99	0.99	0.99			
MobileNet-V2	With Mask	0.99	0.99	0.99	0.99	24,22,210	11.50
	Without Mask	0.99	0.99	0.99			
ResNet-50	With Mask	1.0	1.0	1.0	1.0	2,38,50,242	98.00
	Without Mask	1.0	1.0	1.0			
Inception-V3	With Mask	1.0	0.99	1.0	1.0	2,20,65,314	91.30
	Without Mask	0.99	1.0	1.0			
Xception	With Mask	1.0	1.0	1.0	1.0	2,11,24,010	87.00
	Without Mask	1.0	1.0	1.0			
DenseNet-121	With Mask	1.0	1.0	1.0	1.0	71,68,962	30.90
	Without Mask	1.0	1.0	1.0			
NasNetMobile	With Mask	1.0	1.0	1.0	1.0	44,05,270	20.90
	Without Mask	1.0	1.0	1.0			

4.2.2 Deployment in Mobile Environment

To facilitate a quick real-time detection of mask usage, the proposed framework FaceLite was integrated into an Android-based mobile application. By executing the proposed model locally, TensorFlow Lite could be used to give mobile devices deep learning capabilities. By significantly extending the deep model response period, it promoted hardware speedup, minimal memory utilization, and results in portable devices with low-latency inference efficiency. Facial images were taken using the Android mobile device's built-in camera when the model was installed.

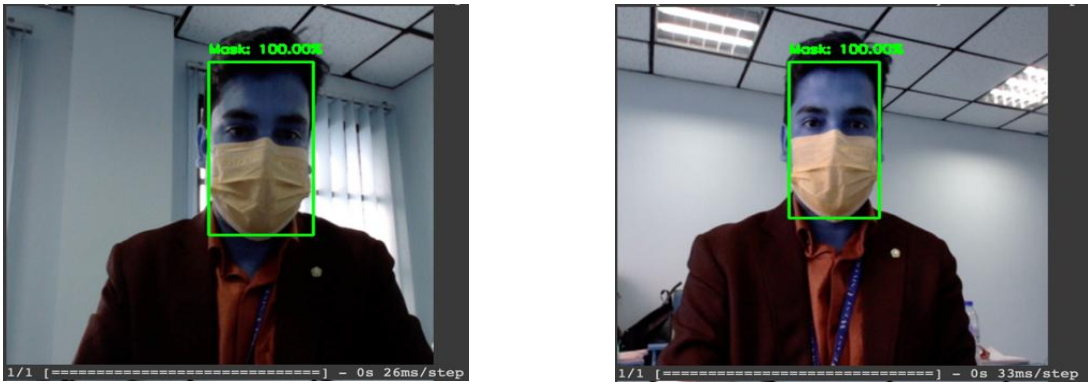
The system processed the input in three stages: accuracy with bounding boxes on faces, mask detection, and face detection. The chosen face detection method (Caffe-DNN) has the following features: it can track faces in video frames; it can process video frames in real-time; it can recognize and discover facial features; it can identify facial features; and it can recognize facial expressions. This work employed real-time face detection using 133 points to represent all facial features. The preprocessing phase was used to resize the image as 224 x 224 pixel values after face detection. The masked facial recognition system's boundaries were also established by the mobile solution. The system's digital camera captured an image, which was checked to see if any facial data was present. If a face was found, its location was surrounded by a frame that displayed details about the mask's function and the identity. In accordance with this system, the Android system was modified in mobile solutions using TensorFlow's "Object Detection Conical" example. Each face was automatically cropped and preprocessed before being given into the model, which distinguished between "masked" and "not masked."

Two more bitmaps were defined for processing. For devices with a sensor that was oriented in landscape, the first step was to rotate the input frame into portrait mode. Each recognized face was depicted using bitmaps, which were also utilized to trim the discovered face's position and resize the image to 224 x 224 pixels for the face detection model's input. The following choices were selected in our situation: green for "Masked" and red for "Not Masked." To build an Android application that uses the classes in the face dataset to categorize faces, a text file with the class names must exist.

Figure 12 shows screenshots of a sample facial image prediction using the proposed FaceLite model. From the figure, it can be seen that the FaceLite model's inference time to detect a facemask is only 24 ms. On the other hand, accuracy is 99.97%. Figure 13 and Figure 14 demonstrate real-time facemask detection (with mask and without a mask) using different transfer learning models. It can be seen that the accuracy of various transfer learning models is 100% in detecting whether a person is wearing a facemask or not. However, it can be seen from Table 4 that the inference times to detect a facemask are 26 ms, 33 ms, 37 ms, 27 ms, 36 ms, and 44 ms for MobileNet-V2, ResNet-50, Inception-V3, Xception, DenseNet-121, and NasNetMobile, respectively. Table 5 compares the performance of the proposed model FaceLite with other transfer learning models. It is clear that the proposed model's inference time on real-time video data is quicker than all other models' inference time which is a desirable feature for the implementation of any model on an edge device.

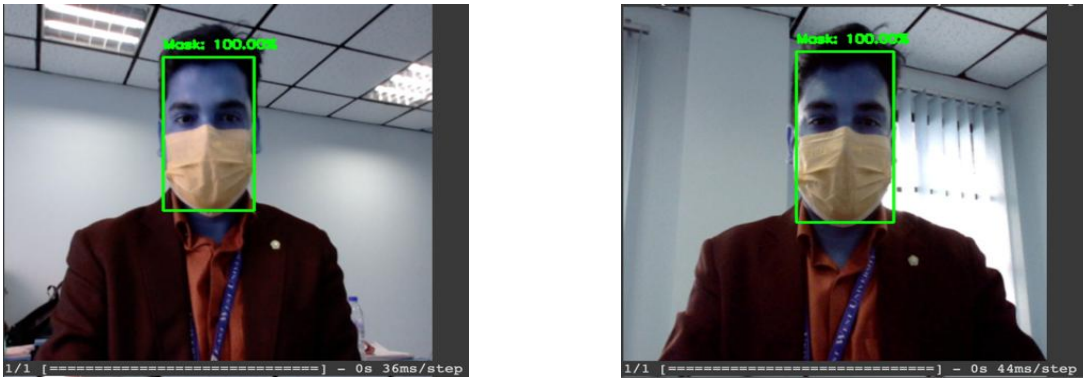
(a) (b)

Figure 12. Real-time facemask detection using FaceLite model



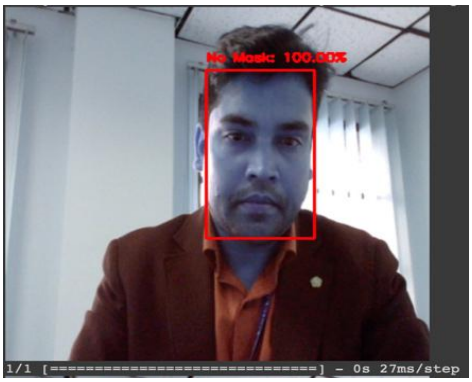
(a) MobileNet-V2 (b) ResNet-50

(c) Inception-V3 (d) Exception

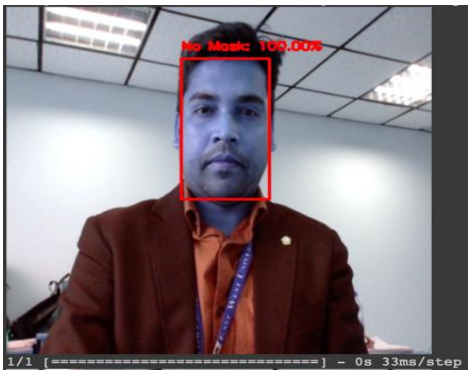


(e) DenseNet121 (f) NasNetMobile

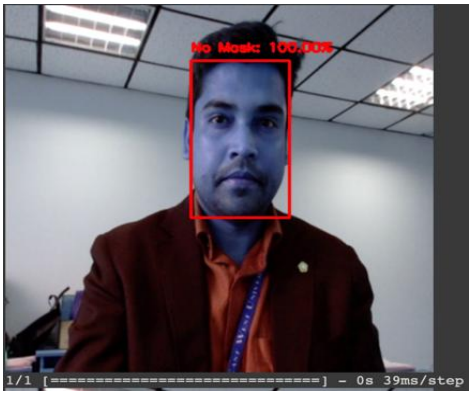
Figure 13. Real-time facemask detection (with mask) using various transfer learning model



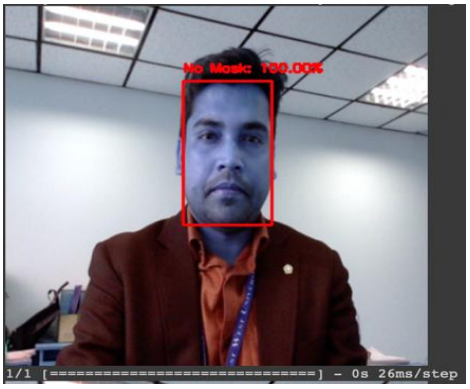
(a) MobileNet-V2



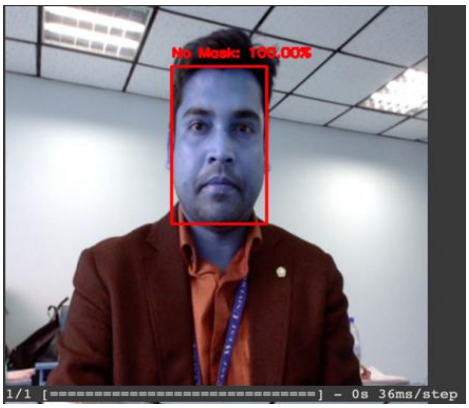
(b) ResNet-50



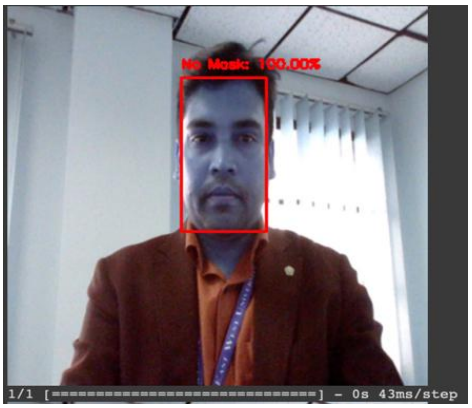
(c) Inception-V3



(d) Exception



(e) DenseNet121



(f) NasNetMobile

Figure 14. Real-time facemask detection (without mask) using various transfer learning model

Table 4. Performance comparison of FaceLite model with other transfer learning models on real-time video data

Model	Inference Accuracy	Inference Time (ms)
FaceLite	99.97	24
MobileNet-V2	100	26
ResNet-50	100	33
Inception-V3	100	37
Xception	100	27
DenseNet-121	100	36
NasNetMobile	100	44

Table 5. Performance comparison with other state-of-the-art facemask detection models

Model	Dataset	Data Description	Total Images	Reported Performance
FaceLite	FaceMask Detection: 12K Images Dataset [80]	Images without a mask are preprocessed from the CelebFace collection, whereas images with a mask are taken from Google searches.	12000	99% (Precision), 99% (Recall), 99% (F1-Score)
Loey [70]	FMDD [71] and MMD [72]	Combination of facemask and medical mask dataset.	1415	81% (Precision)
Yadav [36]	Customized Dataset	Author used customized dataset.	3165	91.7% (Precision)
Amit [90]	RMFRD [91], FMDD [71], and FFHQ [92]	To avoid bias and scarcity, combined three datasets.	7855	98.28% (Precision), 100% (Recall), 99.13% (F1 Score)
Qin [33]	MMD [72]	Medical Mask Dataset	3835	98.70% (Accuracy)
Jiang [69]	FMD [66]	Facemask dataset combined by widerface [93] and MAFA [68]	7959	68.3% (Mean Average Precision)
Shimming [68]	MAFA [68]	Author introduced the dataset	35806	74.6% (Average Precision)
Wei [94]	WiderFace [93] and MaskedFaces [94]	Trained with WiderFace and finetuned with MaskedFaces.	200	86.6% (Accuracy), 87.8% (Recall)

4.2.3 Limitation of the Proposed Model

Many of the mask-containing photos in the "facemask detection ~12K images" training dataset are synthetic. Instead of using photoshopped images of people wearing masks, we should collect real images in order to further refine the face mask detection algorithm. Even though the fake dataset performed admirably in this instance, the genuine thing is always preferable. Secondly, photos of faces that could "confuse" the classifier into believing a person is wearing a mask when they aren't should also be gathered; such examples include bandanas covering the mouth or shirts wrapped around faces. These are all instances of objects that the proposed face mask detector might mistake for face masks.

In the proposed approach of determining whether or not a person is wearing a mask, there are two steps involved:

Step 1: Carry out face recognition.

Step 2: On every face, apply the face mask detector.

This tactic has an issue because a face mask covers part of the face by definition. The face mask detector won't work if the face is sufficiently hidden to prevent the face from being recognized.

One workaround for that issue is to use a two-class object detector that is trained with a with-mask class and a without-mask class. Two ways that the model can be improved are by combining an object detector with a designated class.

First, the object detector will be able to automatically recognize people wearing masks when the face detector would not have been able to since too much of the face is obscured by them. Furthermore, by applying the object detector first, we can obtain bounding boxes for individuals wearing masks and those that don't in a single network forward pass do away with the requirement for face detection and the ensuing face mask detector model. Such a solution is more "elegant" and end-to-end, in addition to being more computationally efficient.

5. Performance Comparison with Other State-of-the-Art Algorithms

Table 5 shows how the suggested FaceLite model performs in contrast to other cutting-edge algorithms. It is clear that many works have been completed by fusing different datasets together or even by producing original images. The table includes descriptions of these datasets as well. An overview of the datasets used in current facemask detection techniques is provided in Table 5, along with their results for comparison to the proposed FaceLite model.

6. Conclusion

This paper presented a two-stage face mask identification framework optimized for real-time inference on Android-powered mobile devices. A DL-based framework was built with two primary parts. The proposed CNN

model and six transfer learning-based custom models were utilized in the second module to perform face mask identification based on the classification, whereas the first module employed Caffe-DNN to process each frame for face detection. The actual trials were done, and the outcomes were thoroughly discussed. First, the accuracy and loss curves for the training and validation datasets for the various models were discussed. After that, the classification report using the confusion matrix was presented and analyzed the predictions based on metrics for accuracy, recall, precision, and F1-score to determine how well each model performed on the test dataset. Through efficient inference with the fewest trainable parameters and disk model size compared to all other transfer learning models, experimental results showed that the proposed CNN model was effective at face masks classification with an accuracy comparable to six other transfer learning models. Future plans include for expanding the framework for human tracking problems and analyzing how the framework improvements affect the accuracy of tracking across various edge devices.

Conflict of Interest

There is no conflict of interest for this study.

References

[1] Nemhauser, J.B. Copyright Page. In *CDC Yellow Book 2024: Health Information for International Travel.* Oxford Academic: Oxford, England, https://doi.org/10.1093/oso/9780197570944.002.0004.

[2] Velavan, T.P.; Meyer, C.G. The COVID-19 epidemic. *Trop. Med. Int. Health* **2020**, *25*, 278–280, https://doi.org/10.1111/tmi.13383.

[3] Wikipedia. COVID-19 pandemic. Available online: https://en.wikipedia.org/wiki/COVID-19_pandemic (accessed on 5 January 2024).

[4] Tracht, S.M.; Del Valle, S.Y.; Hyman, J.M. Mathematical Modeling of the Effectiveness of Facem asks in Reducing the Spread of Novel Influenza A (H1N1). *PLOS ONE* **2010**, *5*, e9018, https://do i.org/10.1371/journal.pone.0009018.

[5] Jefferson, T.; Del Mar, C.B.; Dooley, L.; Ferroni, E.; A Al-Ansary, L.; A Bawazeer, G.; van Dri el, M.L.; Nair, N.S.; A Jones, M.; Thorning, S.; et al. Physical interventions to interrupt or reduc e the spread of respiratory viruses. *Emergencias* **2011**, *2011*, CD006207, https://doi.org/10.1002/146 51858.cd006207.pub4.

[6] Leung, N.H.L.; Chu, D.K.W.; Shiu, E.Y.C.; Chan, K.-H.; McDevitt, J.J.; Hau, B.J.P.; Yen, H.-L.; Li, Y.; Ip, D.K.M.; Peiris, J.S.M.; et al. Respiratory virus shedding in exhaled breath and efficacy of face masks. *Nat. Med.* **2020**, *26*, 676–680, https://doi.org/10.1038/s41591-020-0843-2.

[7] Fang, Y.; Nie, Y.; Penny, M. Transmission dynamics of the COVID-19 outbreak and effectiveness of government interventions: A data-driven analysis. *J. Med Virol.* **2020**, *92*, 645–659, https://doi. org/10.1002/jmv.25750.

[8] World Health Organization. Advice for the public: Coronavirus disease (COVID-19). Available online: https://www.who.int/emergencies/diseases/novel-coronavirus-2019/advice-for-public (accessed on 9 April 2020).

[9] Henson, B. How mask antiviral coatings may limit COVID-19 transmission. Available online: https://www.optometrytimes.com/view/how-mask-antiviral-coatings-may-limit-covid-19-transmission (accessed on 5 January 2024).

[10] Manzoor, S.; Kim, E.-J.; Joo, S.-H.; Bae, S.-H.; In, G.-G.; Joo, K.-J.; Choi, J.-H.; Kuc, T.-Y. Edge Deployment Framework of GuardBot for Optimized Face Mask Recognition With Real-Time Inference Using Deep Learning. *IEEE Access* **2022**, *10*, 77898–77921, https://doi.org/10.1109/access.2022.3190538.

[11] Fan, X.; Jiang, M.; Yan, H. A Deep Learning Based Light-Weight Face Mask Detector With Residual Context Attention and Gaussian Heatmap to Fight Against COVID-19. *IEEE Access* **2021**, *9*, 96964–96974, https://doi.org/10.1109/access.2021.3095191.

[12] Paul, A.K.; Das, D.; Kamal, M. Bangla Speech Recognition System Using LPC and ANN. In Proceedings of 2009 Seventh International Conference on Advances in Pattern Recognition, Kolkata, India, 4–6 February 2009, https://doi.org/10.1109/ICAPR.2009.80.

[13] Zhang, J.; Yu, K.; Wen, Z.; Qi, X.; Paul, A.K. 3D Reconstruction for Motion Blurred Images Us ing Deep Learning-based Intelligent Systems. *Comput. Mater. Contin.* **2021**, *66*, 2087–2104, https:/ /doi.org/10.32604/cmc.2020.014220.

[14] Chhowa, T.T.; Rahman, A.; Paul, A.K.; Ahmmed, R. A Narrative Analysis on Deep Learning in IoT based Medical Big Data Analysis with Future Perspectives. In Proceedings of 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), Cox'sBazar, Bangladesh, 7–9 February 2019, https://doi.org/10.1109/ECACE.2019.8679200.

[15] Ma, T.; Benon, K.; Arnold, B.; Yu, K.; Yang, Y.; Hua, Q.; Paul, A.K. (2020). Bottleneck feature extraction-based deep neural network model for facial emotion recognition. In Proceedings of 10th EAI International Conference on Mobile Networks and Management, MONAMI 2020, Cyberspace, 10–12 November 2020, https://doi.org/10.1007/978-3-030-64002-6_3.

[16] Arifuzzaman, M.; Hasan, R.; Toma, T.J.; Hassan, S.B.; Paul, A.K. An Advanced Decision Tree-B ased Deep Neural Network in Nonlinear Data Classification. *Technologies* **2023**, *11*, 24, https://doi. org/10.3390/technologies11010024.

[17] Paul, A.K.; Mou, J.K.; Turna, T. NEURAL NETWORK BASED REAL TIME PNEUMONIA DETECTION USING TRANSFER LEARNING AND IMAGE AUGMENTATION. *Khulna Univ. Stud.* **2022**, 70–82, https://doi.org/10.53808/kus.2022.icstem4ir.0093-se.

[18] Paul, A.K.; Bhuiyan, Y.S. EchoTrace: A 2D Echocardiography Deep Learning Approach for Left Ventricular Ejection Fraction Prediction. *J. Electron. Electr. Eng.* **2024**, 1–20, https://doi.org/10.372 56/jeee.3120243824.

[19] Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252, https://doi.org/10.1007/s11263-015-0816-y.

[20] Satyanarayanan, M. The Emergence of Edge Computing. *Computer* **2017**, *50*, 30–39, https://doi.org /10.1109/mc.2017.9.

[21] Yang, C.-Y.; Lin, Y.-N.; Shen, V.R.L.; Shen, F.H.C.; Wang, C.-C. A Novel IoT-Enabled System for Real-Time Face Mask Recognition Based on Petri Nets. *IEEE Internet Things J.* **2023**, *11*, 6992–7001, https://doi.org/10.1109/jiot.2023.3313583.

[22] Lin, C.-Y.; Chen, F.-J.; Ng, H.-F.; Lin, W.-Y. Invisible Adversarial Attacks on Deep Learning-Based Face Recognition Models. *IEEE Access* **2023**, *11*, 51567–51577, https://doi.org/10.1109/access.2023.3279488.

[23] He, L.; Liu, G.; Zhou, M. Petri-Net-Based Model Checking for Privacy-Critical Multiagent Systems. *IEEE Trans. Comput. Soc. Syst.* **2022**, *10*, 563–576, https://doi.org/10.1109/tcss.2022.3164052.

[24] Leblond-Menard, C.; Achiche, S. Non-Intrusive Real Time Eye Tracking Using Facial Alignment for Assistive Technologies. *IEEE Trans. Neural Syst. Rehabilitation Eng.* **2023**, *31*, 954–961, https://doi.org/10.1109/tnsre.2023.3236886.

[25] Ali, M.A.; Haque, M.; Alam, S.B.; Rahman, R.; Amin, M.; Kobashi, S. Medical Personal Protecti ve Equipment detection using YOLOv7. In Proceedings of 2023 International Conference on Mach ine Learning and Cybernetics (ICMLC), Adelaide, Australia, 9–11 July 2023, https://doi.org/10.110 9/ICMLC58545.2023.10327939.

[26] Al-Shamdeen, M.J.; Ramo, F.M. Improving Performance of Yolov5n v6. 0 for Face Mask Detecti on. In Proceedings of 2024 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Osaka, Japan, 19–22 February 2024, https://doi.org/10.1109/ICAIIC60209. 2024.10463515.

[27] Weil, A. Can edge computing exist without the edge? part 1: The edge. Available online: https://www.akamai.com/blog/edge/can-edge-computing-exist-without-the-edge-part-1-the-edge (accessed on 5 January 2024).

[28] Worldwide Spending on Edge Computing Will Reach $250 Billion in 2024, According to a New IDC Spending Guide. Available online: https://www.businesswire.com/news/home/20200923005190/e n/Worldwide-Spending-on-Edge-Computing-Will-Reach-250-Billion-in-2024-According-to-a-New-IDC-Spending-Guide (accessed on 5 January 2024).

[29] Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* preprint arXiv:1409.1556. https://doi.org/10.48550/arXiv.1409.1556.

[30] He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778, https://doi.org/10.1109/CVPR.2016.90.

[31] Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A.; Liu, W.; et al. Going deeper with convolutions. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1–9. https://doi.org/10.1109/CVPR.2015.7298594.

[32] Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition, Salt Lake City, UT, USA, 18–3 June 2018, pp. 4510–4520. https://doi.org/10.1109/CVPR.2018.00474.
[33] Qin, B.; Li, D. Identifying Facemask-Wearing Condition Using Image Super-Resolution with Classification Network to Prevent COVID-19. *Sensors* **2020**, *20*, 5236, https://doi.org/10.3390/s20185236.
[34] Kocacinar, B.; Tas, B.; Akbulut, F.P.; Catal, C.; Mishra, D. A Real-Time CNN-Based Lightweight Mobile Masked Face Recognition System. *IEEE Access* **2022**, *10*, 63496–63507, https://doi.org/10.1109/access.2022.3182055.
[35] Bao, Z.; Yang, S.; Huang, Z.; Zhou, M.; Chen, Y. A Lightweight Block With Information Flow Enhancement for Convolutional Neural Networks. *IEEE Trans. Circuits Syst. Video Technol.* **2023**, *33*, 3570–3584, https://doi.org/10.1109/tcsvt.2023.3237615.
[36] Yadav, S. Deep Learning based Safe Social Distancing and Face Mask Detection in Public Areas for COVID-19 Safety Guidelines Adherence. *Int. J. Res. Appl. Sci. Eng. Technol.* **2020**, *8*, 1368–1375, https://doi.org/10.22214/ijraset.2020.30560.
[37] Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016. Available online: http://www.deeplearningbook.org.
[38] Huang, J.; Rathod, V.; Sun, C.; Zhu, M.; Korattikara, A.; Fathi, A.; Fischer, I.; Wojna, Z.; Song, Y.; Guadarrama, S.; et al. Speed/accuracy trade-offs for modern convolutional object detectors. In Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017. pp. 7310–7311.
[39] Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525, https://doi.org/10.1109/cvpr.2017.690.
[40] Lu, Z.; Rallapalli, S.; Chan, K.; Pu, S.; La Porta, T. Augur: Modeling the Resource Requirements of ConvNets on Mobile Devices. *IEEE Trans. Mob. Comput.* **2019**, *20*, 352–365, https://doi.org/10.1109/tmc.2019.2946538.
[41] Ran, X.; Chen, H.; Zhu, X.; Liu, Z.; Chen, J. DeepDecision: A Mobile Deep Learning Framework for Edge Video Analytics. In Proceedings of IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, Honolulu, HI, USA, 16–19 April 2018, https://doi.org/10.1109/INFOCOM.2018.8485905.
[42] Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W., Weyand, T.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* preprint arXiv:1704.04861, https://doi.org/10.48550/arXiv.1704.04861.
[43] Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Sandler, M.; Howard, A.; Le, Q.V. MnasNet: Platform-Aware Neural Architecture Search for Mobile. *arXiv* **2019**, arXiv:1807.11626, https://doi.org/10.48550/arXiv.1807.11626.
[44] Taylor, B.; Marco, V. S.; Wolff, W.; Elkhatib, Y.; Wang, Z. Adaptive deep learning model selection on embedded systems. *ACM Sigplan Notices* **2018**, *53*, 31–43, https://doi.org/10.1145/3299710.3211336.
[45] Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Zheng, X. {TensorFlow}: a system for {Large-Scale} machine learning. In 12th USENIX symposium on operating systems design and implementation (OSDI 16), Savannah, GA, USA, 2–4 November 2016, pp. 265–283.
[46] TensorFlow. Tensorflow lite. Available online: https://www.tensorflow.org/lite (accessed on 10 January 2024).
[47] Edge TPU. Available online: https://cloud.google.com/edge-tpu (accessed on 10 January 2024).
[48] Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Darrell, T. Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the 22nd ACM international conference on Multimedia, Orlando, FL, USA, 3–7 November 2014, https://doi.org/10.1145/2647868.2654889.
[49] PyTorch. Available: https://pytorch.org (accessed on 11 January 2024).
[50] PyTorch Mobile. Available: https://pytorch.org/mobile/home/ (accessed on 11 January 2024).
[51] CUDA Zone. Available: https://developer.nvidia.com/cuda-zone (accessed on 11 January 2024).
[52] CUDNN. Available online: https://developer.nvidia.com/cudnn (accessed on 11 January 2024).
[53] Tan. J. How to choose hardware for edge ml! Available online: https://www.seeedstudio.com/blog/2021/04/02/how-to-choose-hardware-for-edge-ml/ (accessed on 11 January 2024).
[54] Dalal, N.; Triggs, B. Histograms of Oriented Gradients for Human Detection. In Proceedings of the Computer Vision and Pattern Recognition, San Diego, CA, USA, 20–26 June 2005; pp. 886–893, https://doi.org/10.1109/CVPR.2005.177.

[55] Viola, P.; Jones, M. Rapid Object Detection using a Boosted Cascade of Simple Features. In Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition, Kauai, HI, USA, 8–14 December 2001, https://doi.org/10.1109/CVPR.2001.990517.
[56] Lowe, D.G. Object recognition from local scale-invariant features. In Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 20–27 September 1999; pp. 1150–1157, https://doi.org/10.1109/ICCV.1999.790410.
[57] Felzenszwalb, P.; McAllester, D.; Ramanan, D. A discriminatively trained, multiscale, deformable part model. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, USA, 24–26 June 2008; pp. 1–8, https://doi.org/10.1109/CVPR.2008.4587597.
[58] Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Det ection and Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision a nd Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587, https://doi.org/10.11 09/CVPR.2014.81.
[59] Girshick, R. Fast r-cnn. In Proceedings of the IEEE international conference on computer vision, 7–13 December 2015, Santiago, Chile, pp. 1440-1448.
[60] Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. In Proceedings of the Advances in Neural Information Processing Systems 28 (NIPS 2015), Montreal, QC, Canada, 7–12 December 2015; pp. 91–99.
[61] Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788, https://doi.org/10.1109/CVPR.2016.91.
[62] Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* preprint arXiv:1804.02767. https://doi.org/10.48550/arXiv.1804.02767.
[63] Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016, https://doi.org/10.1007/978-3-319-46448-0_2.
[64] Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollar, P. Focal loss for dense object detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 318–327. https://doi.org/10.1109/TPAMI.2018.2858826.
[65] Roy, B.; Nandy, S.; Ghosh, D.; Dutta, D.; Biswas, P.; Das, T. MOXA: A Deep Learning Based Unmanned Approach For Real-Time Monitoring of People Wearing Medical Masks. *Trans. Indian Natl. Acad. Eng.* **2020**, *5*, 509–518, https://doi.org/10.1007/s41403-020-00157-z.
[66] Chiang, D. Detecting faces and determine whether people are wearing mask. Available online: https://github.com/AIZOOTech/FaceMaskDetection (accessed on 12 January 2024).
[67] Albalas, F.; Alzu'Bi, A.; Alguzo, A.; Al-Hadhrami, T.; Othman, A. Learning Discriminant Spatial Features With Deep Graph-Based Convolutions for Occluded Face Detection. *IEEE Access* **2022**, *10*, 35162–35171, https://doi.org/10.1109/access.2022.3163565.
[68] Ge, S.; Li, J.; Ye, Q.; Luo, Z. Detecting Masked Faces in the Wild with LLE-CNNs. In Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 426–434.
[69] Fan, X.; Jiang, M. RetinaFaceMask: A Single Stage Face Mask Detector for Assisting Control of the COVID-19 Pandemic. In Proceedings of 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC). Melbourne, Australia, 17–20 October 2021, https://doi.org/10.1109/SMC52 423.2021.9659271.
[70] Loey, M.; Manogaran, G.; Taha, M.H.N.; Khalifa, N.E.M. Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection. *Sustain. Cities Soc.* **2020**, *65*, 102600–102600, https://doi.org/10.1016/j.scs.2020.102600.
[71] FMDD: face mask detection dataset. Available: https://www.kaggle.com/datasets/andrewmvd/face-mask-detection (accessed on 3 February 2024).
[72] Medical mask dataset. Available online: https://www.kaggle.com/datasets/shreyashwaghe/medical-mask-dataset (accessed on 14 January 2024).
[73] Ristea, N.C.; Ionescu, R.T. Are you wearing a mask? Improving mask detection from speech usin g augmentation by cycle-consistent GANs. *arXiv* preprint arXiv:2006.10147. https://doi.org/10.48550 /arXiv.2006.10147.
[74] Nieto-Rodriguez, A.; Mucientes, M.; Brea, V.M. System for medical mask detection in the operating room through facial attributes. In Pattern Recognition and Image Analysis: 7th Iberian Conference, IbPRIA 2015, Santiago de Compostela, Spain, 17–19 June 2015, https://doi.org/10.1007/978-3-319-19390-8_16.

[75] Friedman, J.; Hastie, T.; Tibshirani, R. Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors). *Ann. Stat.* **2000**, *28*, 337–407, https://doi.org/10.1214/aos/1016218223.
[76] Lienhart, R.; Kuranov, A.; Pisarevsky, V. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In Pattern Recognition: 25th DAGM Symposium, Magdeburg, Germany, 10–12 September 2003, https://doi.org/10.1007/978-3-540-45243-0_39.
[77] Snyder, S.E.; Husari, G. Thor: A Deep Learning Approach for Face Mask Detection to Prevent the COVID-19 Pandemic. In Proceedings of SoutheastCon 2021, Atlanta, GA, USA, 10–13 March 2021, https://doi.org/10.1109/SoutheastCon45413.2021.9401874.
[78] Zhang, K.; Zhang, Z.; Li, Z.; Qiao, Y. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Process. Lett.* **2016**, *23*, 1499–1503, https://doi.org/10.1109/lsp.2016.2603342.
[79] Kim, J.H.; Poulose, A.; Han, D.S. The Extensive Usage of the Facial Image Threshing Machine for Facial Emotion Recognition Performance. *Sensors* **2021**, *21*, 2026, https://doi.org/10.3390/s21062026.
[80] Face mask detection 12k images dataset. Available online: https://www.kaggle.com/datasets/ashishjangra27/face-mask-12k-images-dataset (accessed on 15 January 2024).
[81] Large- scale celebfaces attributes (celeba) dataset. Available online: http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html (accessed on 15 January 2024).
[82] Keras documentation: Keras applications. Available online: https://keras.io/api/applications/ (accessed on 15 January 2024).
[83] Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826. https://doi.org/10.1109/cvpr.2016.308.
[84] Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2261–2269, https://doi.org/10.1109/CVPR.2017.243.
[85] Chollet, F. Xception: Deep learning with depthwise separable convolutions. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
[86] Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning Transferable Architectures for Scalable Image Recognition. In Proceedings of 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018, https://doi.org/10.1109/CVPR.2018.00907.
[87] ReLU6: A Modified Version of Rectified Linear Unit. Available: https://serp.ai/relu6/ (accessed on 22 January 2024).
[88] Bisong, E. Google Colaboratory. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*; Apress: Berkeley, CA, USA, 2019; pp. 59–64, https://doi.org/10.1007/978-1-4842-4470-8_7.
[89] Keras Documentation. Available online: https://keras.io (accessed on 26 December 2018).
[90] Chavda, A.; Dsouza, J.; Badgujar, S.; Damani, A. Multi-Stage CNN Architecture for Face Mask Detection. In Proceedings of 2021 6th International Conference for Convergence in Technology (I2CT). Maharashtra, India, 2–4 April 2021, https://doi.org/10.1109/I2CT51068.2021.9418207.
[91] Wang, Z.; Huang, B.; Wang, G.; Yi, P.; Jiang, K. Masked Face Recognition Dataset and Application. *IEEE Trans. Biom. Behav. Identit- Sci.* **2023**, *5*, 298–304, https://doi.org/10.1109/tbiom.2023.3242085.
[92] FFHQ: Flickr-faces-hq dataset (ffhq). Available: https://github.com/NVlabs/ffhq-dataset (accessed on 3 February 2024).
[93] Yang, S.; Luo, P.; Loy, C.C.; Tang, X. WIDER FACE: A Face Detection Benchmark. In Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 27–30 June 2016, Las Vegas, NV, USA; pp. 5525–5533.
[94] Bu, W.; Xiao, J.; Zhou, C.; Yang, M.; Peng, C. A cascade framework for masked face detection. In Proceedings of 2017 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM), Ningbo, China, 19-21 November 2017, https://doi.org/10.1109/ICCIS.2017.8274819.