computers

MDPI

Article

Real-Time Face Mask Detection Using Federated Learning

Tudor-Mihai David and Mihai Udrescu *

Computer and Information Technology Department, Politehnica University of Timisoara, 300223 Timisoara, Romania; tudor.david@student.upt.ro
* Correspondence: mihai.udrescu@cs.upt.ro

Abstract

Epidemics caused by respiratory infections have become a global and systemic threat since humankind has become highly connected via modern transportation systems. Any new pathogen with human-to-human transmission capabilities has the potential to cause public health disasters and severe disruptions of social and economic activities. During the COVID-19 pandemic, we learned that proper mask-wearing in closed, restricted areas was one of the measures that worked to mitigate the spread of respiratory infections while allowing for continuing economic activity. Previous research approached this issue by designing hardware–software systems that determine whether individuals in the surveilled restricted area are using a mask; however, most such solutions are centralized, thus requiring massive computational resources, which makes them hard to scale up. To address such issues, this paper proposes a novel decentralized, federated learning (FL) solution to mask-wearing detection that instantiates our lightweight version of the MobileNetV2 model. The FL solution also ensures individual privacy, given that images remain at the local, device level. Importantly, we obtained a mask-wearing training accuracy of 98% (i.e., similar to centralized machine learning solutions) after only eight rounds of communication with 25 clients. We rigorously proved the reliability and robustness of our approach after repeated K-fold cross-validation.

Keywords: federated learning; computer vision; real-time face mask detection

check for updates

Academic Editor: Aditya Kumar Sahu

Received: 22 July 2025
Revised: 25 August 2025
Accepted: 29 August 2025
Published: 31 August 2025

Citation: David, T.-M.; Udrescu, M. Real-Time Face Mask Detection Using Federated Learning. *Computers* **2025**, *14*, 360. https://doi.org/10.3390/computers14090360

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The COVID-19 pandemic spread rapidly across the planet, exposing the human species to a systemic risk [1]. Under uncertainty about the mechanism of viral spread and effective cures, the only possible mechanisms to prevent or mitigate the epidemic spread of infection are non-pharmaceutical: quarantines/lockdowns and mask use [2]. However, lockdowns damage economic and social life, so adaptive methods for individual isolation have been proposed [3,4]. Although the practical means of implementing adaptive quarantine methods in real-world societies remain uncertain, the effectiveness of mask-wearing—despite all the controversies [5]—is proven [6,7]. In addition, wearing protective masks at work, school, or public places does not hinder economic and social activities. Consequently, mask use should be enforced in all places where infection can spread [8].

Installing real-time tracking systems to detect those wearing masks is required to mitigate the consequences of epidemic diseases. Accordingly, we need an affordable hardware platform to apply the tracking solution at a large scale and provide the computational power for deep learning image recognition. One way to build a prototype for such a mask detection system is to use a Single Board Computer (SBC) such as Raspberry Pi [9], which has adequate resources, including a camera attached to supply the video stream.

There are significant anthropometric variations based on race, gender, culture, and age [10,11]. Furthermore, the type of mask also plays a role in training an efficient and accurate detection system [12]. Thus, obtaining comprehensive data for efficiently training a mask detection system is difficult [13]. The situation becomes even more difficult when considering a lightweight SBC implementation for the mask detection solution.

Traditional centralized mask-detection systems require that every raw video frame be sent to a cloud or data center server for training. Deployed on a large scale, this configuration faces three significant challenges. First, there is bandwidth usage: simultaneously transmitting continuous video streams from several cameras can stress connections. The second issue concerns privacy: concentrating raw biometric images in a single location subjects them to specific regulations (e.g., GDPR in the European Union). The third challenge is vulnerability to failures: any server or network interruption can completely halt detection at the local level. Federated learning [14,15] can address these challenges by keeping images on local devices and only sending encrypted model updates, thus reducing costs while significantly improving privacy and system robustness.

This paper addresses the problem of building a lightweight and scalable solution to the mask detection problem using image recognition and machine learning (ML). To this end, we propose a novel federated learning implementation that deals with the difficulty in collecting a comprehensive data set. The distributed training approach of federated learning is better suited for the heterogeneous distribution of anthropometric traits in a large population (i.e., at national, international, or continental levels [16]). Further, federated learning ensures better data privacy because the raw image data remains on the client side (i.e., not exposed by communicating it to a centralized server [17]). The main contributions of our work are as follows:

- We provide a novel FL framework for face mask detection using image recognition;
- We tailor a face and mask detection model that requires limited hardware resources, namely, Raspberry Pi single-board computers;
- We present simulation-based experiments that demonstrate the effectiveness and accuracy of our FL framework compared to the conventional, centralized ML approach.

The remainder of this paper is organized as follows: Section 2 presents existing work on mask detection systems, Section 3 describes the hardware platform used to train and run the face detection model, Section 4 describes the FL model, the dataset, and the software architecture, Section 5 shows the evaluations of our proposed mask detection system, and Section 6 states the final remarks.

2. Related Work

According to a comprehensive survey of computer-based face mask detection systems [18], such methods can generally be categorized into two main types: conventional approaches and those that use neural networks. Conventional approaches often employ boost algorithms and rely on manually crafted features but represent only a minor part. Methods that utilize neural networks are classified into three categories according to the number of stages involved in processing. The following subsections provide further details and explanations on both conventional and neural network methods.

2.1. Conventional Methods

The conventional approaches rely on the principle that when a mask is worn correctly, neither the nose nor the mouth will be visible. For example, a conventional approach to face detection, like the Viola and Jones detector trained with the AdaBoost algorithm, searches for distinct characteristics such as Haar-like features [19].

Another illustration of conventional classification presents an Internet of Things (IoT) indoor safety setup [20]. This system employs several AdaBoost classifiers from OpenCV to recognize faces, noses, and mouths. When a face area is identified without a nose and mouth, it is classified as wearing the mask correctly. Detection of a nose is labeled as an *improper mask* while the presence of a mouth is categorized as *no mask*. Multi-pose masked face detection using a nose and mouth classifier was introduced by [21]; their approach used Haar-like features based on a specular map, from which they formed a nose and mouth dataset. If neither the nose nor the mouth can be detected, it classifies the face as masked. Otherwise, it falls under the no mask category. Following a similar approach to [20], the system first identifies the face and then utilizes mouth detection to confirm the presence of a mask; this algorithm is optimized to operate efficiently on the PYNQ-Z2 SoC platform, achieving a response time of 0.13 s and a dataset-specific accuracy of 96.5%.

2.2. Neural Network-Based Methods

The neural network approaches fall into three groups, according to the number of stages involved: single-stage, two-stage, and multi-stage methods. Single-stage methods typically rely on transferring learning from object-detection algorithms. Two-stage approaches involve using a combination of neural networks, namely a neural network paired with a specially designed deterministic model or another neural network. Accordingly, the two-stage methods comprise two main components: face region predetection and face region classification. The former identifies potential facial regions, while the latter determines mask-wearing conditions. Multistage methods involve more intricate processing steps or the utilization of multiple models, leading to much higher computational expenses.

Single-stage strategies employing deep learning techniques with a transfer learning approach are effective in detecting faces wearing masks. Such methods capitalize on pre-existing object and face detectors to accomplish the mask-detection task. To illustrate this strategy, ref. [22] introduced the MobileNetMask, aimed at curbing the transmission of SARS-COV-2; their method identifies various types of facial masks. Mask classification is performed using ROI (Region of Interest) detection from the SSD and ResNet10 models. Due to its computational efficiency and compact structure tailored for mobile platforms, MobileNet-V2 emerges as a favorable option for embedded systems, demonstrating superior accuracy compared to alternative techniques. In [23], a lightweight deep learning model based on a Depthwise Separable Convolutional Neural Network (DWS-based MobileNet) is proposed; they aimed to develop a real-time, resource-efficient solution for identifying mask presence, achieving 93.14% accuracy.

Two-stage strategies for identifying masked faces involve an initial step of detecting faces followed by a verification stage for face classification. The first phase applies a range of face or object detectors to recognize potential faces, with object detectors also providing feature descriptors for potential candidates. The subsequent phase employs various classifiers or models to determine if a person is wearing a mask and, if so, whether it is worn correctly. Thus, by combining the functionalities of object detectors with classification models, this approach successfully achieves the objective of detecting masked faces. The methodology presented by [24] introduces such a two-phase technique for detecting obscured faces and implementing recommendations utilizing facial landmarks for recognition. Initially, this method employs MTCNN (Multi-Task Cascaded Convolutional Neural Networks) to determine if the individual is wearing a multicolored mask, followed by a check for a skin-colored mask; it can distinguish between five facial conditions (i.e., uncovered faces, faces with facial hair, single-colored mask wearers, multicolored mask wearers, and individuals with skin-colored masks), achieving an accuracy rate of 97.13%. Although effective in distinguishing skin-colored masks, this method requires higher computational

resources and faces challenges in adaptability due to its reliance on empirical thresholds. In another study, ref. [25] combined Haar-like cascade classifiers and two MobileNet models to detect face masks. The Haar-like cascade classifier recognizes facial regions, while the first MobileNet model distinguishes between faces with and without masks. The second MobileNet model evaluates the proper wearing of masks. Overall, this integrated system achieves an accuracy level of around 90%.

Multistage strategies involve multiple processing phases. These include, for instance, detecting humans or facial regions, extracting regions of interest (ROI), or feature vectors, normalizing these features, and performing classification or making predictions through sequences. Consequently, these multistage methods can be developed through various arrangements of these elements. In [26], a five-step approach is introduced, which involves collecting image data, examining human body positions, recognizing regions of interest (ROI), standardizing images, and categorizing faces as wearing masks. Their technique utilizes Openpose [27] to analyze body positions and standardize ROI. After the standardization procedure, the images are run through a facial mask detection network, resulting in success rates of 95.8% in daylight and 94.6% at night. Reference [28] proposes a methodology with four stages to determine the status of face mask wearing. These stages include face detection and cropping, image preprocessing, condition identification, and image resolution enhancement. A vital aspect of this method is the utilization of the super-resolution network SRNet, which enhances the quality of facial images, consequently improving the accuracy of classifying mask-wearing conditions. This approach achieves an impressive accuracy rate of 98.7%, surpassing traditional deep learning techniques by 1.5% thanks to the integration of SRNet. However, the use of multiple networks makes this methodology very demanding in terms of computational resources.

Table 1 comprehensively summarizes and compares conventional approaches and neural network-driven techniques (containing the three respective subgroups) by presenting individual forecasts for each method, including details on the datasets used and the average experimental results for training and deploying.

Table 1. A comparison of all the techniques we examined. It presents the category (conventional or neural network-driven), the reference to the corresponding paper, the number of classes representing the labels identified by the algorithm (2—{mask, no mask}; 3—{mask, incorrectly worn mask, no mask}), information on the dataset used for training, results showing the outcome of the experiments, and the experimental environment detailing the hardware setup used for running and training.

Category	Method Reference	Classes	Dataset	Results	Experimental Environment
Conventional	[20]	3	Not provided	Accuracy = 84–91%	Intel i7 7700-HQ quad-core CPU 2.80 GHz with 16 GB RAM
Conventional	[21]	2	1000 images	Accuracy = 86.9%	Not provided
Single-stage	[22]	2	3835 images	Accuracy = 93%	Google Colab (Tesla V100-SXM2-16 GB)
Single-stage	[23]	2	8000 images	Accuracy = 93.14%	Google Colab
Two-stage	[24]	2	5504 images	Accuracy = 97.13%	Not provided
Two-stage	[25]	3	1270 images	Accuracy = 90%	Not provided
Multi-stage	[26]	2	992 images	Accuracy = 95.8%	Not provided
Multi-stage	[28]	3	3835 images	Accuracy = 98.7%	Intel i7 and P600 GPU with 4 GB memory

In contrast to the centralized approaches summarized in Table 1, which rely on aggregating data in a single location, potentially exposing sensitive information to breaches, our federated learning solution operates in a decentralized manner, ensuring enhanced privacy and security by keeping raw data localized on edge devices without transmission to a central server. This mitigates risks associated with data centralization, such as unauthorized access or large-scale leaks, while still achieving a superior 98% accuracy through collaborative model updates across distributed nodes. Furthermore, our training was efficiently conducted on a resource-constrained Raspberry Pi 4 device, demonstrating scalability and energy efficiency on low-power hardware, unlike the more computationally intensive setups in the referenced methods that demand more potent hardware.

3. Experimental Configuration

We need a device capable of running all the algorithms required to achieve the performance that our mask detection task demands. Accordingly, at the client level, we identified the Raspberry Pi as the most convenient option for both quality and affordability, according to [29]. However, although the Raspberry Pi has the specifications of a standard computer, its computing power is limited compared to a traditional desktop or laptop.

For the project described in this paper, we selected the Raspberry Pi 4 Model B with 8GB of RAM and a Micro-SDXC card (to store the Raspberry Pi OS, project files, and a database of images necessary for training). The video capture component is a webcam that provides high-definition video quality and operates at a frame rate of 30 fps.

Given the processor's propensity to overheat, particularly during training sessions, we implemented an enhanced cooling solution. Consequently, our system comprises two large radiators (which also serve as a protective casing for the entire device) and two fans to maintain optimal temperature control. Figure 1 shows the entire hardware configuration.

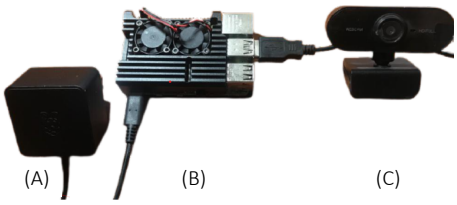


Figure 1. The hardware configuration for a client: (**A**) AC adapter, (**B**) Raspberry Pi 4 Model B, (**C**) webcam.

To address the constraints of implementing a lightweight yet accurate and adaptive mask detection model, we propose a federated learning framework, where the Raspberry Pi system shown in Figure 1 represents a single component (or client) within a larger network. This network includes M aggregators, each overseeing N clients; see Figure 2. The operational FL sequence is as follows: Initially, the server sends the initial model weights to each of the M aggregators linked to it. Subsequently, each aggregator relays these weights to its N clients. The training then commences individually among these clients, using their local data. After finishing the training, each client sends their updated weights to their respective aggregator. Each aggregator then builds a more general model with the weights received from its clients. In the final step, each aggregator transmits its aggregated weights back to the server. The server performs a comprehensive aggregation of all the weights received from the lower-level aggregators, thus delivering the final model weights. This marks the end of the first communication cycle, and the process is repeated until the desired results are obtained. This paper considers an FL architecture with one aggregator ($M = 1$) that oversees N clients.

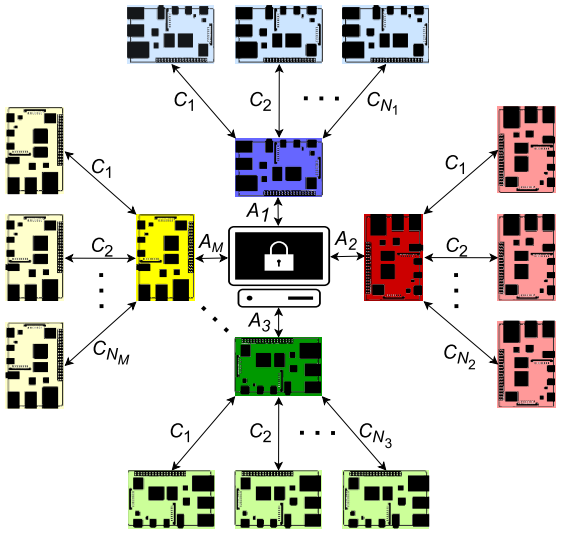


Figure 2. A federated learning architecture consisting of M aggregators, labeled $A_1, A_2, A_3, \ldots, A_M$, each managing N_i clients ($i = \overline{1, M}$) corresponding to each aggregator. Specifically, aggregator A_1 manages clients $C_1, C_2, \ldots, C_{N_1}$, extending up to aggregator A_M, which oversees clients C_{N_M}.

4. Materials and Methods

We propose a method combining computer vision, transfer learning via MobileNetV2, and federated learning for distributed training to create a real-time mask detector. This approach is designed with a strong focus on privacy, providing an efficient, scalable, and privacy-conscious solution for the real-time identification of face masks.

Computer vision plays a crucial role in our proposed system, enabling the real-time detection of people not wearing masks and the automatic analysis of visual data. This technology is particularly effective in monitoring large groups where direct supervision is challenging.

MobileNetV2, with transfer learning, enables effective processing by exploiting pre-trained ImageNet features to achieve high accuracy even with less data.

Federated learning is a key component of our proposed system, facilitating model training on multiple devices using local datasets without the need for data exchange. This distributed approach enhances data security, aligns with data protection regulations, and reduces privacy concerns. Moreover, federated learning enhances scalability and adaptability, enabling the model to glean insights from diverse data sources and improve its generalization across various user groups and data types.

The key concepts in federated learning are *client*, *aggregator/server*, *round*, *epoch*, and *privacy boundary*. The *client* is the edge device (e.g., Raspberry Pi board) responsible for conducting local training on its private dataset. The *aggregator/server* is the entity that gathers the weights from the clients and calculates their weighted average (FedAvg) to update the global model. The *round* refers to a complete cycle that involves updating global weights, client-side local training, and aggregation at the server. The local *epoch* is a process in which a client makes a complete pass over its dataset during a round. The *privacy boundary* ensures that the raw images are kept on the client device; only encrypted weight updates are sent.

4.1. Transfer Learning

We have chosen a neural network architecture that meets our needs in terms of efficiency and size so that it can be deployed efficiently on a Raspberry Pi, thus guaranteeing practicality. Accordingly, we adapt the pre-trained MobileNetV2 architecture as the basis for the initial model that the server passes to the clients [30].

MobileNetV2 encompasses only 3.4 million parameters (approximately 10 MB of storage); this makes it the most lightweight network to achieve an accuracy of at least 97% in our initial repeated cross-validation experiments. The use of depth-wise separable convolutions and inverted residual blocks helps to diminish the size of the weight-update tensors transmitted during federated learning rounds. Given its compact size and high accuracy, MobileNetV2 is particularly well-suited for deployment on resource-constrained devices, such as the Raspberry Pi 4, where memory and computational efficiency are crucial.

MobileNetV2 [31] was originally trained using transfer learning [32] on the ImageNet dataset [33], is known for its efficiency, and is designed for mobile and embedded vision applications. In our model, we only use the feature extraction components of MobileNetV2, eliminating the top classification layers. On top of this basis model, we introduce a custom head tailored to our task: an Average Pooling Layer, a Flatten Layer, a Dense Layer with ReLU Activation, and an Output Dense Layer.

The Average Pooling Layer reduces the spatial dimensions of the feature maps, the Flatten Layer transforms them into a one-dimensional vector, the Dense Layer with ReLU Activation provides the network with the capacity to learn complex patterns, and the Output Dense Layer has two neurons corresponding to the two classes of our classification task (i.e., mask and no mask), with the softmax activation function employed to interpret the model's outputs as class probabilities.

To emphasize the feature extraction capabilities of the pre-trained MobileNetV2, all of its layers are frozen during training, allowing us to harness the rich representations learned by MobileNetV2 on ImageNet while fine-tuning the model for our specific classification challenge. Figure 3 presents the entire neural network architecture we propose for mask detection.

Furthermore, the use of transfer learning with ImageNet pre-trained weights means that a large dataset for training the model is not required. In a real-world federated learning scenario, each client might have a small and heterogeneous dataset, yet the approach can still achieve robust performance. This is a significant advantage, as it demonstrates the effectiveness of the method even with limited data.

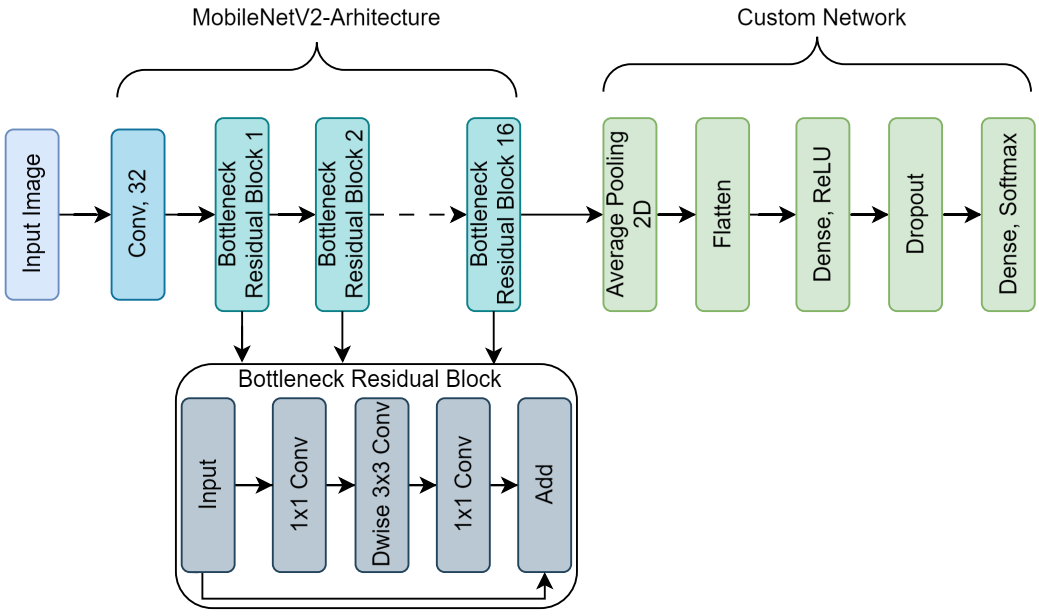


Figure 3. Our architecture for mask detection integrates MobileNetV2 with a custom network for image classification using pooling, flattening, and fully connected layers with dropout and softmax output.

4.2. Federated Learning

Federated learning involves numerous devices; hence, it is also termed *collaborative learning* [34,35]. Training is decentralized, using local data across multiple devices, eliminating the need to share user data between users [36].

The rationale behind using FL for the mask detection problem is twofold. First, FL ensures the individual's privacy, as the brute data (i.e., images) remain at the local level, with the server receiving only the model weights. Second, the main aim of our method is to minimize the occurrence of false positives. (The main problem is when an individual is labeled as correctly wearing the mask, even if it is not.) Experimental results suggest that a false positive is expected once per 500 FPS. FL repeatedly trains identical neural networks with various simulated clients, each possessing unique data, so—at least in theory—the probability of developing a sturdy model should be significantly enhanced within the FL paradigm.

To experiment with and test our mask detection system, we simulated the entire process, even if the software involved by running the prediction model at the client level can run on the Raspberry Pi described in Figure 1. The reason is the unavailability of sufficiently many Raspberry Pis. In this situation, the Raspberry Pi functions as both a server and a client (see Figure 4); however, the system was designed and implemented as two separate entities—server and clients. The simulation process starts with the server generating the clients. Since this is a client simulation, and we have only a few Raspberry Pi boards, we randomly split the entire dataset among the clients so that each client receives the same amount of Independent Identically Distributed (IID) data.

The dataset consists of 6000 images, evenly balanced in two classes (3000 with masks and 3000 without masks). To explain the distribution process, we define a general formula for allocating samples to clients, taking into account the training split used in preprocessing, where the following are defined:

- D is the total number of images in the dataset (i.e., 6000);
- S is the training split ratio (i.e., 0.8 for 80% training data, with the remaining 20% reserved for testing);
- C is the number of classes (i.e., 2: with mask and without mask);
- N is the number of clients (varies per experiment).

The total number of training images is calculated as $D \times S$, with the number of samples per client being $\frac{D \times S}{N}$, and the number of samples per class per client $\frac{D \times S}{N \times C}$.

For $D = 6000$, $S = 0.8$, and $C = 2$, we have:

- **Experiment 1** ($N = 100$): Each client receives $\frac{6000 \times 0.8}{100} = 48$ samples, with $\frac{48}{2} = 24$ samples per class (with mask and without mask);
- **Experiment 2** ($N = 50$): Each client receives $\frac{6000 \times 0.8}{50} = 96$ samples, with $\frac{96}{2} = 48$ samples per class;
- **Experiment 3** ($N = 25$): Each client receives $\frac{6000 \times 0.8}{25} = 192$ samples, with $\frac{192}{2} = 96$ samples per class.

This distribution ensures no overlap between clients' datasets, maintaining data privacy. The dataset assignment remains fixed across all communication rounds, with each client training on its unique subset throughout the experiment. This setup aligns with federated learning principles, preventing the server from accessing confidential data and ensuring robust model training across diverse client datasets.

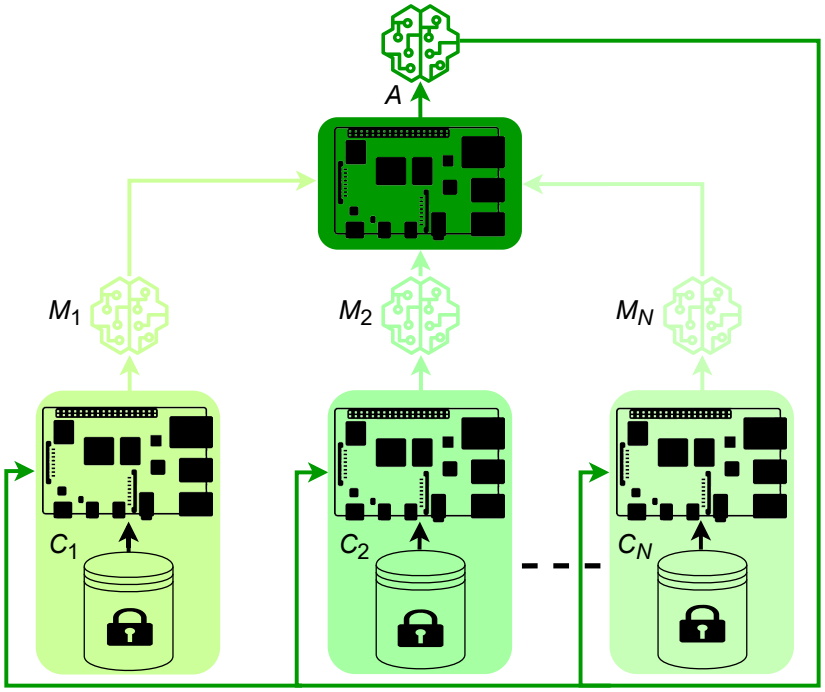


Figure 4. The federated learning framework with Raspberry Pi clients and a server (aggregator) coordinating training and weight aggregation.

The FL process we simulate (according to the architecture in Figure 4) begins when clients acquire the initial global model (based on MobileNetV2) from the server. Subsequently, the clients will use their local data to train their models over several epochs. (Throughout this local training phase, the mean gradient is employed for optimization.) Following several iterations of local updates, the resulting local models are dispatched to the server for comprehensive aggregation. This cycle continues until the aggregated model meets the desired performance standards.

The aggregation creates a new global model in each t communication round [37]. In this paper, we perform the aggregation according to Equation (1), where w_{t+1} represents the global weights, K is the total number of clients, n is the total number of data samples across all clients, n_k is the number of data samples held by client k, and w_{t+1}^k are the local weights from client k. Algorithms 1 and 2 present the pseudocode for the federated learning procedure that we emulate in this paper.

$$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k \tag{1}$$

Algorithm 1 ServerUpdate

1: Initialize w_0
2: **for** each round $t = 1, 2, \ldots$ **do**
3: $\quad S_t \leftarrow$ random set of n simulated clients
4: $\quad$ **for** each client $k \in S_t$ in parallel **do**
5: $\quad\quad w_{t+1}^k \leftarrow$ UpdateClientWeights (k, w_t)
6: $\quad$ **end for**
7: $\quad$ **// update global weights**
8: $\quad w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
9: **end for**

Algorithm 2 UpdateClientWeights (k, w)

1: **// run on client** k
2: **for** each local epochs $i = 1$ to e **do**
3: $\quad w \leftarrow w - \nabla(w)$
4: **end for**
5: **return** w

4.3. Computer Vision

The application running at the client level to achieve mask detection requires a few procedures in the domain of computer vision [38]. Each data stream that arrives from the video frame is processed independently. If a person's face is detected, mask recognition and label prediction follow. Accordingly, the output is presented in real-time as a rectangle that borders the person's face, and its color corresponds to the final mask-wearing decision (red—no mask; blue—mask incorrectly worn; green—mask correctly worn).

As prescribed by [39], we utilized diverse libraries to analyze the photos collected from the video stream, such as OpenCV, TensorFlow, and Numpy; we used them to extract relevant information from incoming photos, make predictions, scale the photos, and create the appropriate output. To make predictions on people's faces we used the OpenCV model [40]. An essential observation is that we integrated data collection. When a person is recognized with a mask, the corresponding photographs will be recorded on the device. Indeed, each device/client can collect data to perform federated learning. The diagram in Figure 5 illustrates the application running at the client level.

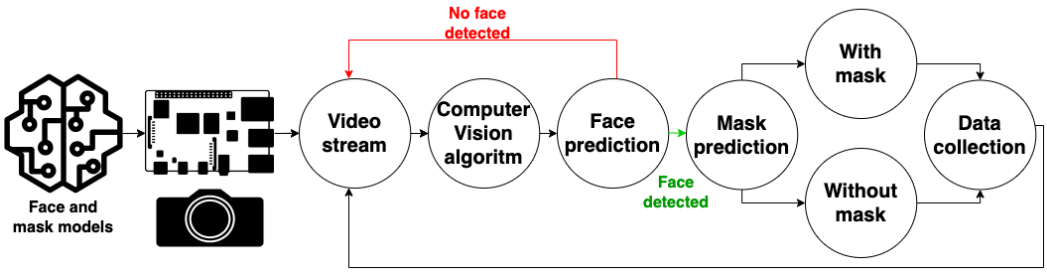


Figure 5. The diagram describes the client-level computer vision workflow for face and mask recognition, from model import to prediction and frame storage.

4.4. Dataset

We took the facemask-wearing dataset from Kaggle [41], which contains 12K images divided into test, train, and validation folders. The dataset includes images with different resolutions and orientations individuals wearing the mask properly, improperly, or not wearing the mask.

For this paper's experiments, we randomly select 6000 images divided equally into two groups, with and without masks, each containing 3000 images. The dataset includes individuals from different ethnic backgrounds, skin tones, genders, and facial structures; this diversity is crucial to develop models that perform well in different demographics and minimize bias. The photos show the faces of men and women from various regions around the world, ensuring a comprehensive representation. The with-mask group includes face masks commonly used during the COVID-19 pandemic, such as surgical masks, standard disposable blue/green masks, N95/FFP2 respirators, high-filtration masks (with and without exhalation valves), cloth masks, reusable masks (made from different fabrics, colors, and patterns), KN95 masks (similar to N95 but with unique design variations), and fashion masks (masks with custom designs, logos, and patterns). In this way, we ensure that models trained on the dataset can accurately recognize faces in various scenarios, enhancing

their reliability and fairness. To significantly mitigate demographic bias, we established an equal distribution of 50% female and 50% male participants, ensuring that each of the four main visible-ethnicity categories provided approximately 25% of the images.

4.5. Data Preprocessing

Image preprocessing is an essential step in preparing data for neural network models, especially with a pre-trained neural network architecture such as MobileNetV2. This step ensures that the input data meet the necessary format and scale requirements, thus improving model performance and convergence. In Figure 6, we provide a breakdown of each step, focusing on how images are prepared as a MobileNetV2 input.

The images are loaded and resized; each image is loaded from the specific directory path. The target size is set to 224 × 224 pixels, which resizes all images to a uniform input shape—this size is standard for many prebuilt models.

We converted each image from a PIL (i.e., Pillow, a Python 3.9 image processing library) image object to a NumPy array; this transformation allows for further image manipulations and operations. After completing these steps, we begin the preprocessing that prepares the image array specifically for MobileNetV2, which rescales the pixel values from their original range of [0, 255] to a new range that matches the neural network requirements, namely the range $[-1, 1]$, to normalize the input data.

In the next step, we extract the labels for each image, which are obtained from the directory name path. We transform the class labels into binary vectors: label binarization. After this step, we transformed the binary vectors into one-hot codes, which are required for categorical classification models—this allows MobileNetV2 to assign probabilities to each class.

The final step splits the dataset, where 80% of the data is used for training and 20% for testing; to perform all these operations, we use the Tensorflow image preprocessing library. For label binarization, one-hot encoding, and data splitting, we use the Sklearn library.

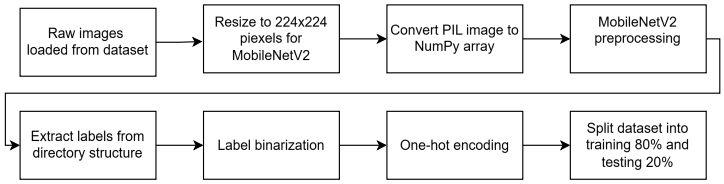


Figure 6. Data processing: a chronological-order display of the data processing steps we applied to the original dataset.

4.6. Assessment Metrics

We evaluated the performance of our model with the accuracy metric; as shown in Table 1, column 'Results,' previous work mostly used this metric as being the most relevant. Indeed, for our problem, namely correctly labeling a person as properly wearing or not wearing a face mask, what matters is the ratio of correctly labeled persons to the total number of assessed individuals, hence accuracy.

5. Experimental Results

In the experiments, we evaluated the performance of our federated learning system across different combinations of communication rounds and clients. Our goal is to analyze model accuracy, loss trends, and overall classification performance while ensuring stability through repeated cross-validation, confusion matrices, and statistical measures such as mean with standard deviation, 95% confidence intervals, and p-values. All data from the results are available at https://github.com/tudordavidz/fedmask (accessed on 7 January 2025).

5.1. Experiment 1: Two Communication Rounds, 100 Clients

This experiment was designed to test the effectiveness of federated learning with a large number of participating clients (100) over a minimal number of communication rounds (2). The main objective was to evaluate whether the model can achieve high performance with a limited number of aggregation rounds. The performance trend can be seen in Figure 7a, where the model showed rapid convergence and reached near-optimal accuracy in only two rounds. The loss decreased steadily, indicating an effective learning process. We present the cross-validation results in Figure 7b, where the repeated k-fold validation boxplot demonstrates high stability with minimal variance in accuracy across validation folds. The mean confusion matrix in Figure 7c shows near-perfect classification performance with low misclassification rates. The diagonal dominance indicates that the model effectively discriminates between classes. This experiment demonstrates that a large number of clients can enable fast convergence with only two rounds of communication. The results indicate that additional rounds are not necessary if a sufficient number of clients are involved.

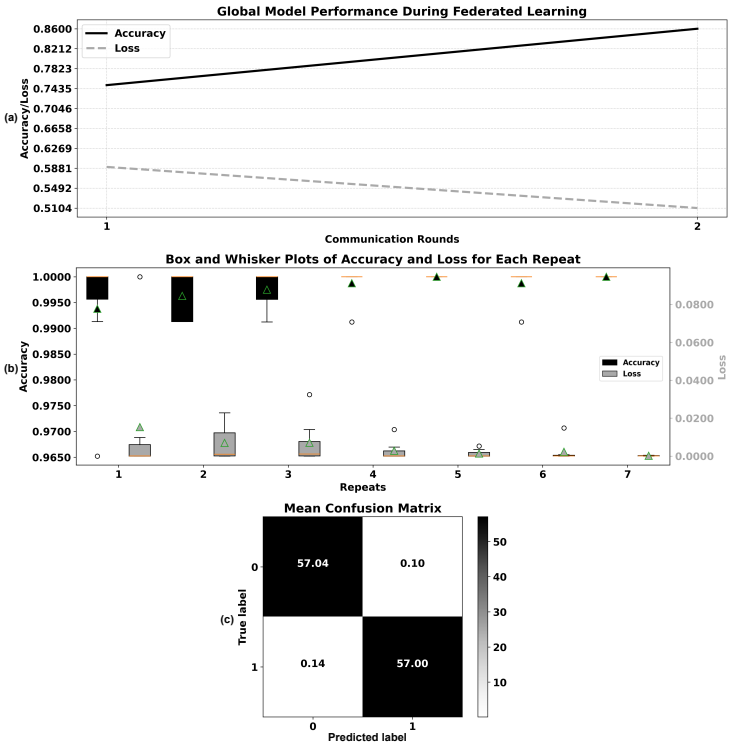


Figure 7. (**a**) Global accuracy and loss trends after two communication rounds with 100 clients; (**b**) box plots of mean and median accuracy and loss across 49 evaluations (7 repeats × 7 folds); and (**c**) mean confusion matrix for mask classification with labels: (0) mask; (1) no mask.

5.2. Experiment 2: Four Communication Rounds, 50 Clients

Here, the number of communication rounds was doubled (4), and the number of clients was reduced to 50. The aim was to find out whether increasing the number of rounds could compensate for the effects of a lower number of clients and improve learning stability. On the performance trend, Figure 8a shows that the accuracy continued to improve over the four rounds, while loss steadily decreased. In this case, the longer training process allowed for greater refinement of the model updates. The cross-validation results in Figure 8b reveal that the accuracy variance increased slightly compared to Experiment 1, suggesting that the reduction in the number of clients led to some instability between the validation runs. The confusion matrix in Figure 8c shows that the model

continued to perform well in classification, but the classification rates were slightly lower than in Experiment 1, suggesting that a smaller number of clients may contribute to greater variability in model updates. Increasing the number of communication rounds helped maintain high accuracy despite a lower number of clients, but it could not fully compensate for the increased variability. This experiment suggests that both the number of clients and the communication rounds play a role in the stability of performance.

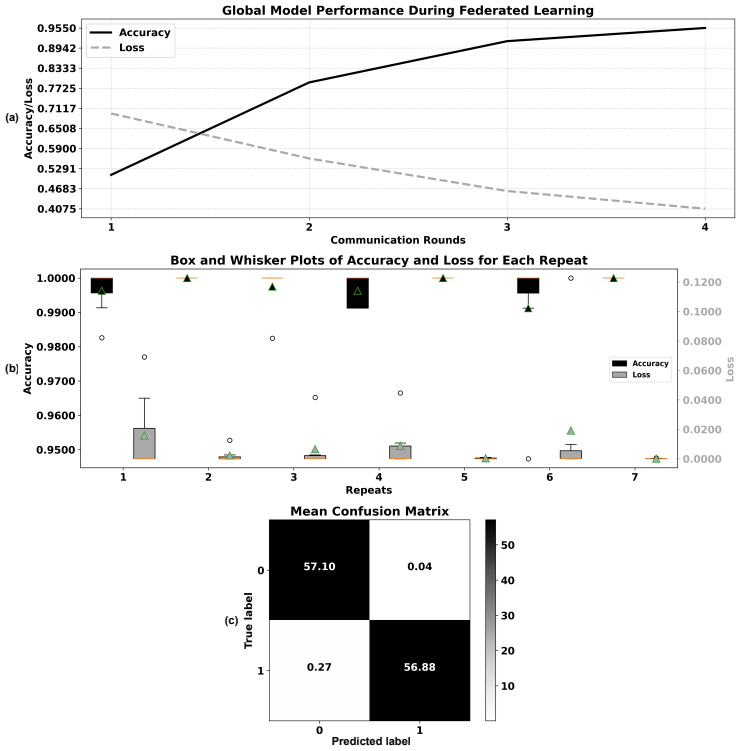


Figure 8. (**a**) Global accuracy and loss trends after four communication rounds with 50 clients; (**b**) box plots of mean and median accuracy and loss across 49 evaluations (7 repeats × 7 folds); and (**c**) mean confusion matrix for mask classification with labels: (0) mask; (1) no mask.

5.3. Experiment 3: Eight Communication Rounds, 25 Clients

In the third experiment, the communication rounds increased to 8, while the number of clients was reduced to 25. The aim was to find out whether a higher number of rounds can compensate for the lower number of clients to ensure a comparable level of performance. The performance trend in Figure 9a shows that the accuracy improved steadily in the eight rounds and eventually stabilized at a high level. The loss continued to decrease, indicating an effective model refinement over multiple iterations. The cross-validation results in Figure 9b reveal that the variance was slightly higher than in previous experiments, which is expected due to the lower number of clients contributing to global updates. The confusion matrix in Figure 9c proves that the model maintained low misclassification rates, comparable to Experiments 1 and 2; this emphasizes that increasing the number of rounds may help to compensate for fewer clients to some degree. Although the highest accuracy was achieved in this experiment, there were slightly more fluctuations due to the lower number of clients. The results suggest that more communication rounds can improve accuracy but do not fully compensate for a lower number of clients.

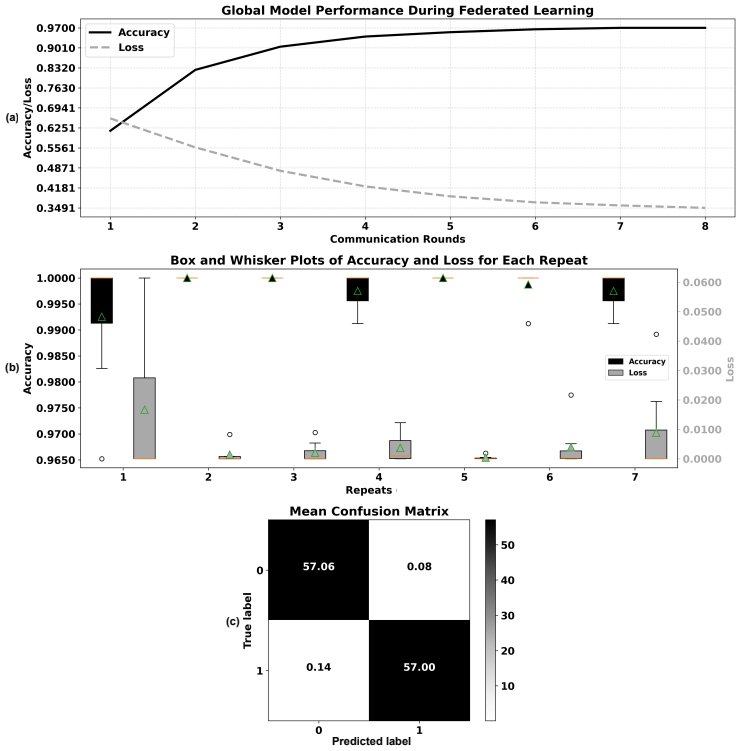


Figure 9. (**a**) Global accuracy and loss trends after eight communication rounds with 25 clients; (**b**) box plots of mean and median accuracy and loss across 49 evaluations (7 repeats × 7 folds); and (**c**) mean confusion matrix for mask classification with labels: (0) mask; (1) no mask.

5.4. The Impact of the Number of Clients and Communication Rounds

A local epoch was used in Experiments 1, 2, and 3, but the number of communication rounds and clients per round varied (rounds/clients: 2/100, 4/50, and 8/25, respectively). We present the key observations in terms of accuracy and losses.

5.4.1. Accuracy

All three configurations achieved a high final accuracy (in the order of 97–100% when testing with repeated k-fold validation), but Experiment 3 (8 rounds, 25 clients each) achieved the highest mean accuracy. As the number of rounds increased (and clients per round decreased), the accuracy of the model improved slightly. This suggests that more frequent global updates (communication rounds) may slightly increase performance, even when the total number of client updates is constant.

5.4.2. Loss

The final loss also decreased slightly as the number of rounds increased. Experiment 3 had the lowest mean loss, indicating better convergence, while Experiment 1 (only 2 rounds) had a slightly higher loss. More rounds (with smaller customer batches) led to a more gradual and stable training process, resulting in a lower final loss compared to fewer, larger rounds.

5.4.3. General Trend

Increasing the number of communication rounds (while proportionally reducing the number of clients per round) benefited the model performance. The improvements were incremental—accuracy gains of a few tenths of a percent and corresponding loss reductions—but consistent. Such results suggest that frequent aggregation helps the federated model reach a slightly higher performance plateau by incorporating updates

more frequently. (However, we note that the total work—and total number of clients seen—was similar in these experiments.)

5.5. Impact of Local Training Epochs (One vs. Two Epochs per Round)

We compare each of the previous three experiments (which use one epoch) with their two-epoch counterparts to evaluate the effect of more local training per round. In these one-epoch with two-epoch pairs, the number of rounds and clients per round remain the same; only the local epochs differ (1 and 2). The results for the next group of experiments based on two epochs are in the Supplementary Materials.

5.5.1. Experiment 1 with Two Epochs

We present the results of Experiment 1 (two rounds, 100 clients), which uses two epochs instead of one. The two-epoch experiment achieved slightly higher accuracy than Experiment 1 with one epoch. With two epochs of local training, clients made more progress on their local data before averaging, which improved the accuracy of the global model and resulted in a lower final loss. However, the improvement was substantial: the additional local epoch led to a notable increase in performance, boosting the results of the training accuracy from 86% at one epoch to 95.5% at two epochs, despite the limited number of rounds.

5.5.2. Experiment 2 with Two Epochs

We present the results of Experiment 2 (four rounds, 50 clients) with two epochs. Similarly, Experiment 2 with one epoch was surpassed by its two-epoch counterpart in performance, as the model achieved higher accuracy at the end of the four rounds, reaching a training accuracy of 97.5%, along with a reduced loss. Additional local training per round in this four-round scenario accelerated learning so that the model was better trained after the same number of communication rounds. The advantage of two epochs was clear but still relatively small in absolute terms (since the one-epoch version was already performing well).

5.5.3. Experiment 3 with Two Epochs

We present the results of Experiment 3 (eight rounds, 25 clients) with two epochs instead of 1. This setting has the best overall performance: an accuracy of 98% during training. It maintained a slight advantage in accuracy over the one-epoch Experiment 3, indicating that even with many rounds, additional local training per round can further refine the model. The loss was the lowest among all experiments, confirming a cumulative positive effect of more local epochs in combination with frequent rounds. In particular, as the number of rounds increases, the additional gain from an extra local epoch decreases somewhat—Experiment 3 already had high performance, so the gain is marginal, but still present.

5.6. Statistics of Experimental Results

Federated learning enables models to be trained collaboratively across distributed clients while maintaining data privacy protection. However, evaluating the robustness and generalization of such models requires rigorous statistical validation, as presented in Table 2. These results are derived from repeated k-fold validation applied to Experiments 1, 2, and 3, with statistical measures including mean with standard deviation, 95% confidence intervals, and p-values.

All three experiments show high accuracy (99.8%) with low standard deviation, indicating model stability and consistency across distributed clients. The narrow 95% confidence intervals confirm statistical reliability, while low loss values suggest effective error

minimization. Extremely small p-values indicate that accuracy and loss improvements are statistically significant, not some random occurrence or artifact. This strengthens confidence in the validity and robustness of the model across different data splits. The consistency of our experiments shows that federated learning achieves high accuracy in a decentralized setting and provides performance comparable to centralized models with strong generalization and minimal variance.

Table 2. The statistics of accuracy and loss metrics across Experiments 1, 2, and 3, including mean ± standard deviation, 95% confidence intervals, and p-values from repeated k-fold validation in federated learning.

Experiment	Metric	Mean ± Std Dev	95% Confidence Interval	p-Value
1	Accuracy	0.9979 ± 0.0058	[0.9962, 0.9995]	0.0000
	Loss	0.0051 ± 0.0147	[0.0009, 0.0093]	0.0192
2	Accuracy	0.9973 ± 0.0084	[0.9949, 0.9997]	0.0000
	Loss	0.0075 ± 0.0218	[0.0013, 0.0138]	0.0195
3	Accuracy	0.9980 ± 0.0060	[0.9963, 0.9998]	0.0000
	Loss	0.0053 ± 0.0133	[0.0014, 0.0091]	0.0081

5.7. Real-Time Predictions

This section presents how our client application, optimized for the Raspberry Pi platform, performs mask-wearing recognition. Figure 10 shows how the system displays predictions in real time. Figure 5 shows that the first prediction identifies the face area; when the face is detected, the last prediction is also examined to determine the three possible cases: no protective mask (framed in red), with protective mask (framed in green), and improperly wearing mask (framed in blue). This last case is an extension of the protective mask detected case, but it only activates when the prediction probability is between 60–90%. If the accuracy exceeds this range, we assume a properly worn mask. In addition, when one of the two cases (with and without a mask) is detected, the system collects the data automatically. This data is then used in the federated learning process to train a new model.



Figure 10. Real-time mask detection with face coordinates, illustrating correctly worn (green), incorrectly worn (blue), and no mask (red) scenarios.

6. Discussions and Conclusions

This paper introduces a novel federated learning framework to address the challenge of creating an efficient and scalable face mask detection system. The core motivation is to avoid potential public health disasters and severe disruption associated with epidemic respiratory infections; in this context, we use FL to circumvent the difficulties associated with centralized data collection and model training— particularly in the context of heterogeneous data distribution across different populations—and improve data privacy while still producing a highly accurate model. To the best of our knowledge, this is the first mask detection system to integrate MobileNetV2 with a federated learning structure, operating

end-to-end on Raspberry Pi-like devices, retaining raw images locally, and addressing the privacy and bandwidth challenges faced by prior centralized systems.

Our federated learning mask recognition system achieved a remarkable training accuracy of 98% after eight rounds of communication with 25 clients and two epochs on a Raspberry Pi 4, outperforming almost all the methods listed in Table 1. The only exception is the last method in the table, which achieved a slightly higher accuracy of 98.7%. However, it is important to note that their model was trained in a centralized environment, while our model works in a decentralized environment. In contrast to the other approaches, which utilized high-performance hardware such as Intel i7 CPUs or Google Colab for training, our solution showcases outstanding resource efficiency on the Raspberry Pi 4. Moreover, our FL model prioritizes privacy by retaining data on client devices, making it perfectly suited for low-resource and privacy-critical applications. By training decentrally, our model learns from a much wider variety of data, resulting in a significantly more robust model. Nevertheless, issues such as early-round loss variations indicate a need for further optimization, differing from centralized methods, which rely on more powerful hardware, but do not address privacy concerns.

6.1. Performance

The experiments demonstrate the accuracy of our FL mask detection framework. We demonstrated that the accuracy of the FL model progressively increased with the number of communication rounds, thus showing that more frequent interactions between clients and the server contribute positively to the model performance. Indeed, the model achieved a strong accuracy of 98% after eight communication rounds with 25 clients and two epochs; this significantly improved the accuracy observed in earlier rounds with higher client counts. These results confirm the assumption of the FL paradigm, so even if fewer clients in each round may initially slow down the model's learning, increased communication brings more precise weight updates, thus increasing accuracy over time.

Even with a relatively small number of communication rounds, our mask detection FL model demonstrated substantial accuracy, which, although slightly lower than the centralized model, is still highly effective for practical applications. This means that FL is a viable alternative to centralized training, particularly when dealing with data privacy issues, heterogeneous environments, and hardware limitations.

However, our results also emphasize specific challenges inherent in FL, such as variability in performance based on the number of clients and communication rounds. Fluctuations in loss values (especially in the earlier rounds) imply that there may be inefficiencies or imbalances in how the model aggregates information from different clients. These variations suggest the need for further investigation into optimization techniques that could stabilize the training process and ensure more consistent performance across different configurations.

By implementing the MobileNetV2 architecture on resource-constrained devices such as the Raspberry Pi, we demonstrate that it is possible to achieve high accuracy in mask detection while preserving data privacy and accommodating the diverse anthropometric traits of a broad population.

Moreover, by implementing the MobileNetV2 architecture on resource-constrained devices such as the Raspberry Pi, we demonstrate that high accuracy in mask detection can be achieved while preserving data privacy and accommodating the diverse anthropometric traits of a broad population.

6.2. Epidemic Respiratory Infections Scenario

Traditional centralized models require large-scale data collection for training at the server level, thus exposing individual data to privacy issues. Furthermore, communication

with a central server entails additional logistic problems. In contrast, FL uses local training at the device level, reducing individual data exposure and thus enhancing individual privacy. As such, when used in public health emergencies —such as epidemic respiratory infections—the FL architecture allows institutions to monitor mask compliance safely, a crucial factor in public health response where trust and privacy are paramount.

The FL model is compatible with smaller-scale and low-power hardware devices, such as the Raspberry Pi, making it a cost-effective and accessible choice for organizations in all sectors, including public health agencies, schools, businesses, retail stores, and more. Because it has reduced hardware requirements, the FL mask detection model can be deployed in various public spaces, including work environments and educational institutions, with minimal costs. In addition, the distributed approach to private mask detection enables areas with limited infrastructure and resources to participate, enhancing this solution's reach and inclusivity.

Furthermore, the decentralized nature of the FL model supports scalability in diverse regions and heterogeneous situations. Its capacity to be expanded by adding more devices makes it well suited to geographic scalability and inclusion, as it adapts to real-time local data variations; this enables people from various regions and backgrounds to contribute to and benefit from the system more quickly, making it a democratic solution that can be implemented globally without expensive centralized resources. Indeed, our FL solution fosters fast, equitable access and resilience in managing health crises.

6.3. Limitations

Using federated learning on the Raspberry Pi has limitations, mainly related to low hardware resources. Consequently, the most crucial downside is the training time, which increases significantly due to the limited processor and memory of the device. In a federated learning scenario, where training is performed on multiple devices in parallel, each Raspberry Pi will contribute gradually, slowing down the whole process of aggregating and updating the centralized model. Therefore, an increase in latency is expected when we aim for high accuracy. However, the federated approach can have significant advantages in the long run: using multiple devices that perform parallel training on distributed and varied datasets will achieve a more robust and generalized model.

6.4. Future Work

To reduce loss variability and enhance model accuracy, particularly in the early stages of communication, future research should focus on improving the process for aggregating client updates. To confirm the usefulness and adaptability of this Federated Learning framework, it should also be applied to various other image recognition tasks. Further, we plan to advance our system in future studies by implementing advanced aggregation techniques to ensure more consistent training and better convergence. Our objective also includes integrating secure aggregation and differential privacy techniques to improve data protection and overall system resilience. Moreover, we can extend this framework to cover additional computer vision tasks, such as emotion or activity recognition, which will underscore its adaptability.

Supplementary Materials: The following supporting information can be downloaded at https://www.mdpi.com/article/10.3390/computers14090360/s1: Figure S1: Experiment 1 (two rounds, 100 clients) with two epochs; Figure S2: Experiment 2 (four rounds, 50 clients) with two epochs; Figure S3: Experiment 3 (eight rounds, 25 clients) with two epochs; Table S1: This table contains a summary of accuracy and loss metrics across Experiments 1, 2, and 3, including mean $\pm$ standard deviation, 95% confidence intervals, and p-values from repeated k-fold validation in federated learning.

Author Contributions: Conceptualization, T.-M.D. and M.U.; methodology, T.-M.D. and M.U.; software, T.-M.D.; validation, T.-M.D. and M.U.; investigation, T.-M.D. and M.U.; resources, T.-M.D. and M.U.; writing—original draft preparation, T.-M.D.; writing—review and editing M.U.; visualization, T.-M.D. and M.U.; supervision, M.U.; project administration, T.-M.D. and M.U. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: All images provided in this manuscript are original and were created by the authors.

Data Availability Statement: The dataset utilized in this research study is publicly available on https://www.kaggle.com/datasets/ashishjangra27/face-mask-12k-images-dataset (accessed on 16 February 2023). The source code for this project can be found https://zenodo.org/records/15051505 (accessed on 19 March 2025).

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviation is used in this manuscript:

FL Federated Learning

References

1. Rizwan, M.S.; Ahmad, G.; Ashraf, D. Systemic risk: The impact of COVID-19. *Financ. Res. Lett.* **2020**, *36*, 101682. [CrossRef]
2. Topirceanu, A.; Udrescu, M.; Marculescu, R. Centralized and decentralized isolation strategies and their impact on the COVID-19 pandemic dynamics. *arXiv* **2020**, arXiv:2004.04222. [CrossRef]
3. Hurtado, S.; Marculescu, R.; Drake, J. Quarantine in Motion: A Graph Learning Framework to Reduce Disease Transmission Without Lockdown. In Proceedings of the 2022 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Istanbul, Turkey, 10–13 November 2022; pp. 454–461.
4. Horstmeyer, L.; Kuehn, C.; Thurner, S. Balancing quarantine and self-distancing measures in adaptive epidemic networks. *Bull. Math. Biol.* **2022**, *84*, 79. [CrossRef] [PubMed]
5. Abboah-Offei, M.; Salifu, Y.; Adewale, B.; Bayuo, J.; Ofosu-Poku, R.; Opare-Lokko, E.B.A. A rapid review of the use of face mask in preventing the spread of COVID-19. *Int. J. Nurs. Stud. Adv.* **2021**, *3*, 100013. [CrossRef]
6. Damette, O.; Huynh, T.L.D. Face mask is an efficient tool to fight the COVID-19 pandemic and some factors increase the probability of its adoption. *Sci. Rep.* **2023**, *13*, 9218. [CrossRef]
7. Qiu, Z.; Espinoza, B.; Vasconcelos, V.V.; Chen, C.; Constantino, S.M.; Crabtree, S.A.; Yang, L.; Vullikanti, A.; Chen, J.; Weibull, J.; et al. Understanding the coevolution of mask wearing and epidemics: A network perspective. *Proc. Natl. Acad. Sci. USA* **2022**, *119*, e2123355119. [CrossRef] [PubMed]
8. Feng, S.; Shen, C.; Xia, N.; Song, W.; Fan, M.; Cowling, B.J. Rational use of face masks in the COVID-19 pandemic. *Lancet Respir. Med.* **2020**, *8*, 434–436. [CrossRef]
9. Upton, E.; Halfacree, G. *Raspberry Pi User Guide*; John Wiley & Sons: Hoboken, NJ, USA, 2016.
10. Dhaliwal, J.; Wagner, J.; Leong, S.L.; Lim, C.H. Facial Anthropometric Measurements and Photographs—An Interdisciplinary Study. *IEEE Access* **2020**, *8*, 181998–182013. [CrossRef]
11. Kukharev, G.; Kaziyeva, N. Digital facial anthropometry: Application and implementation. *Pattern Recognit. Image Anal.* **2020**, *30*, 496–511. [CrossRef]
12. Chen, Y.; Hu, M.; Hua, C.; Zhai, G.; Zhang, J.; Li, Q.; Yang, S.X. Face mask assistant: Detection of face mask service stage based on mobile phone. *IEEE Sens. J.* **2021**, *21*, 11084–11093. [CrossRef]
13. Zhang, X.Q.; Jiang, R.H.; Fan, C.X.; Tong, T.Y.; Wang, T.; Huang, P.C. Advances in deep learning methods for visual tracking: Literature review and fundamentals. *Int. J. Autom. Comput.* **2021**, *18*, 311–333. [CrossRef]
14. Zhang, C.; Xie, Y.; Bai, H.; Yu, B.; Li, W.; Gao, Y. A survey on federated learning. *Knowl.-Based Syst.* **2021**, *216*, 106775. [CrossRef]
15. Lameh, S.F.; Noble, W.; Amannejad, Y.; Afshar, A. Analysis of federated learning as a distributed solution for learning on edge devices. In Proceedings of the 2020 International Conference on Intelligent Data Science Technologies and Applications (IDSTA), Valencia, Spain, 19–22 October 2020; pp. 66–74.
16. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and Open Problems in Federated Learning. *Found. Trends® Mach. Learn.* **2021**, *14*, 1–210. [CrossRef]
17. Sen, J.; Waghela, H.; Rakshit, S. Privacy in Federated Learning. *arXiv* **2024**, arXiv:2408.08904.

18. Wang, B.; Zheng, J.; Chen, C.P. A survey on masked facial detection methods and datasets for fighting against COVID-19. *IEEE Trans. Artif. Intell.* **2021**, *3*, 323–343. [CrossRef]
19. 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. CVPR 2001, Kauai, HI, USA, 8–14 December 2001; Volume 1, p. I.
20. Petrović, N.; Kocić, Đ. IoT-based system for COVID-19 indoor safety monitoring. In Proceedings of the IcETRAN, Belgrade, Serbia, 28–29 September 2020.
21. Dewantara, B.S.B.; Rhamadhaningrum, D.T. Detecting multi-pose masked face using adaptive boosting and cascade classifier. In Proceedings of the 2020 International Electronics Symposium (IES), Surabaya, Indonesia, 29–30 September 2020; pp. 436–441.
22. Dey, S.K.; Howlader, A.; Deb, C. MobileNet mask: A multi-phase face mask detection model to prevent person-to-person transmission of SARS-CoV-2. In Proceedings of the International Conference on Trends in Computational and Cognitive Engineering: Proceedings of TCCE 2020, Dhaka, Bangladesh, 17–18 December 2020; Springer: Singapore, 2020; pp. 603–613.
23. Asghar, M.Z.; Albogamy, F.R.; Al-Rakhami, M.S.; Asghar, J.; Rahmat, M.K.; Alam, M.M.; Lajis, A.; Nasir, H.M. Facial mask detection using depthwise separable convolutional neural network model during COVID-19 pandemic. *Front. Public Health* **2022**, *10*, 855254. [CrossRef] [PubMed]
24. Zereen, A.N.; Corraya, S.; Dailey, M.N.; Ekpanyapong, M. Two-stage facial mask detection model for indoor environments. In Proceedings of the International Conference on Trends in Computational and Cognitive Engineering: Proceedings of TCCE 2020, Dhaka, Bangladesh, 17–18 December 2020; Springer: Singapore, 2021; pp. 591–601.
25. Rudraraju, S.R.; Suryadevara, N.K.; Negi, A. Face mask detection at the fog computing gateway. In Proceedings of the 2020 15th Conference on Computer Science and Information Systems (FedCSIS), Sofia, Bulgaria, 6–9 September 2020; pp. 521–524.
26. Lin, H.; Tse, R.; Tang, S.K.; Chen, Y.; Ke, W.; Pau, G. Near-realtime face mask wearing recognition based on deep learning. In Proceedings of the 2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 9–12 January 2021; pp. 1–7.
27. Cao, Z.; Simon, T.; Wei, S.E.; Sheikh, Y. Realtime multi-person 2d pose estimation using part affinity fields. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7291–7299.
28. Qin, B.; Li, D. Identifying facemask-wearing condition using image super-resolution with classification network to prevent COVID-19. *Sensors* **2020**, *20*, 5236. [CrossRef] [PubMed]
29. Mustakim, N.; Hossain, N.; Rahman, M.M.; Islam, N.; Sayem, Z.H.; Mamun, M.A.Z. Face Recognition System Based on Raspberry Pi Platform. In Proceedings of the 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), Dhaka, Bangladesh, 3–5 May 2019; pp. 1–4.
30. Phan, T.V.; Sultana, S.; Nguyen, T.G.; Bauschert, T. *Q*-TRANSFER: A Novel Framework for Efficient Deep Transfer Learning in Networking. In Proceedings of the 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Fukuoka, Japan, 19–21 February 2020; pp. 146–151.
31. Patel, R.; Chaware, A. Transfer Learning with Fine-Tuned MobileNetV2 for Diabetic Retinopathy. In Proceedings of the 2020 International Conference for Emerging Technology (INCET), Belgaum, India, 5–7 June 2020; pp. 1–4.
32. Dong, K.; Zhou, C.; Ruan, Y.; Li, Y. MobileNetV2 Model for Image Classification. In Proceedings of the 2020 2nd International Conference on Information Technology and Computer Application (ITCA), Guangzhou, China, 18–20 December 2020; pp. 476–480.
33. Ridnik, T.; Ben-Baruch, E.; Noy, A.; Zelnik-Manor, L. Imagenet-21k pretraining for the masses. *arXiv* **2021**, arXiv:2104.10972.
34. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A.y. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the Artificial Intelligence and Statistics, PMLR, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
35. Sébert, A.G.; Pinot, R.; Zuber, M.; Gouy-Pailler, C.; Sirdey, R. SPEED: Secure, PrivatE, and efficient deep learning. *Mach. Learn.* **2021**, *110*, 675–694. [CrossRef]
36. Fadaeddini, A.; Majidi, B.; Eshghi, M. Secure decentralized peer-to-peer training of deep neural networks based on distributed ledger technology. *J. Supercomput.* **2020**, *76*, 10354–10368. [CrossRef]
37. Ye, Y.; Li, S.; Liu, F.; Tang, Y.; Hu, W. EdgeFed: Optimized federated learning based on edge computing. *IEEE Access* **2020**, *8*, 209191–209198. [CrossRef]
38. Zhang, X.; Xu, S. Research on Image Processing Technology of Computer Vision Algorithm. In Proceedings of the 2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL), Chongqing, China, 10–12 July 2020; pp. 122–124.
39. Voulodimos, A.; Doulamis, N.; Doulamis, A.; Protopapadakis, E. Deep learning for computer vision: A brief review. *Comput. Intell. Neurosci.* **2018**, *2018*, 7068349. [CrossRef] [PubMed]

40. Khan, M.; Chakraborty, S.; Astya, R.; Khepra, S. Face detection and recognition using OpenCV. In Proceedings of the 2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 18–19 October 2019; pp. 116–119.
41. Jangra, A. Face Mask Detection 12K Images Dataset. 2020. Available online: https://www.kaggle.com/datasets/ashishjangra27/face-mask-12k-images-dataset?resource=download (accessed on 16 February 2023).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.