EdgeCart: An Edge Intelligence based Smart Shopping Cart with Mask Surveillance System using IoT

Md. Ishan Arefin Hossain, Dept. of CSE, Southeast University, ishan.arefin@seu.edu.bd
S. M. Shahidur Harun Rumy, Dept. of CSE, Southeast University, rumywithy@gmail.com

Abstract--**Recent insurgences of global pandemics have already unearthed some of the weaknesses of health surveillance in busy areas such as superstores, markets, malls. Especially, shopping in the crowded superstores has become a lengthy and health-threatening phenomenon nowadays. In this era of the global pandemic, ensuring that every customer at the super shops is wearing a facemask is a difficult ask. Added to that, providing a safe and compelling shopping experience for the customers is another challenge related to this because the customers have to wait in the long queues with others to pay for the purchased products after a lengthy scanning process. The proposed edge intelligence-based smart shopping cart with mask surveillance system using IoT connectivity will ensure that the customers will have a facemask on while using the cart along with a minimal queue for product scanning at the payment counter. The proposed surveillance system provides reduced latency and cost related to processing in the cloud with the help of edge intelligence. Moreover, the IoT based product detection and scanning system of the cart diminishes the need for long queues at the counter where the risk factors related to health are involved because of close human presence. The proposed unique IoT based smart cart with edge intelligence is the key to the pandemic surveillance in superstores, and it concurrently reduces the health-threatening and tiring long time expenditure at the superstore counters for product scanning.**

Index Terms-- **Edge Intelligence, IoT, Machine Learning, Mask Surveillance, Smart Shopping Cart.**

I. Introduction

Nowadays, there are supermarkets in every corner of the city. In every supermarket, all the products are labeled with barcodes. So, when a customer picks some products, he/she has to go to the checkout counter, and the counter manager has to unload all the products from the cart and scan them one by one to calculate the bill for the customer. When the market is crowded, the queue becomes longer. In this pandemic situation, it is advised to maintain distance for reducing health threats. For solving this problem, a new smart shopping cart is introduced in this proposal. In the proposed system, every product has an RFID tag. When a shopper picks a product for keeping that in the shopping cart, he/she has to scan the product with the RFID reader embedded in the cart, and then the product gets verified by the smart processing system installed in the cart. After the product verification, the system shows the total price of all the products using a display and adds the scanned product's details in the cloud database. At present, just as it is crucial for people to maintain distance, it is also important to use masks. For this, we have to keep an eye on the customers all the time. But it is pretty impossible or needs more human resources to make sure every customer is wearing a mask if we try to do it manually. That is why our system also proposed a feature that will automatically make sure every customer wears a face mask. If a customer does not wear a mask, our system will prompt them to do so.

There are six sections to this study. Section 2 summarizes prior research on our topic. The proposed work is presented in Section 3. Section 4 explains the technique we used to implement and evaluate our proposed work. The experimental results are presented in Section 5. Finally, in Section 6, the work is summarized in the form of a conclusion. The future scope has also been discussed in this section.

II. Related Work

Similar research carrying the idea of smart carts emits the significant potentiality of probable implementation of these carts in real-life scenarios. One research paper proposed a system using an RFID reader in front of the checkout counter, and that counter's IOT based system is connected with the cloud [3].

Moreover, another research proposal has mentioned about developing a centralized billing system using Zigbee. Zigbee is less power-consuming and suitable for indoor based IoT network [1].

Another paper illustrates a primary model of a smart shopping cart (SSC) that can be integrated into the smart mall system. That proposed SSC can give clients a productive UI with the goal that the shopping management can be adequately advanced. In that model, with the capacity of facial recognition on the UI, the SSC can perceive the client and further provide the related assistive shopping data dependent on the shopper's buying history [2].

In paper [4], Fan et al. has used the OpenCV based AdaBoost algorithm for face detection. But he has modified the AdaBoost algorithm to get a better result.

Hsu et al. [5] proposed a method based on novel lighting compensation technique and a nonlinear color transformation. It identifies skin regions in the image and generates face candidates based on their spatial arrangement.

Jiang et al. [6] proposed the RetinaFaceMask face mask detection model in collaboration with a cross-class object removal algorithm. The developed model includes a single stage detector that uses a feature pyramid network to achieve slightly better precision and recall than the baseline result.

III. Proposed System

In this experimental system, a regular shopping basket is used as an intelligent shopping cart. At the bottom of the cart, a load cell is installed. In front of the cart, there is an LCD to display information, and an RFID reader for scanning the products with RFID tags. There is a camera attached to the back of the cart. The camera can be adjusted according to the height of the customer. The system starts when a buyer adjusts the camera according to his/her height so that it can capture their face. Because our system will detect whether the customer is wearing a mask, and it will warn the customer to wear a mask if the customer does not. After scanning a product, the shopper keeps the product in the cart. Then the weight sensor embedded into the cart verifies the product with the information it gets from the RFID tag.

A. Components Description

Raspberry Pi: The microcomputer Raspberry Pi 3B was used as the processing unit of this system. The Raspberry Pi is used for interfacing with the rest of the sensors and actuators. The main task of the Raspberry Pi is to communicate with the cloud database and update the product list into the cloud. It also detects any suspicious shop-lifting activity using the weight sensor.

Camera Module: To capture images for classification, we have used a camera module. It has a resolution of 5MP. The Raspberry Pi's CSI Camera Port can be used to connect the camera module to the Raspberry Pi.

RFID Card: A radio frequency identification reader (RFID reader) is a device used to gather information from an RFID tag, which is used to track individual objects. Radio waves are used to transfer data from the tag to a reader. In the proposed system, the RFID Card stores name, price and weight of a product. This information is used to detect a product without a barcode reader.

Load Cell: A load cell is a type of transducer which performs the functionality of converting force into an electric output. It is basically a device that measures strain and then converts force into electric energy. In the proposed system, the load cell is used as the weight sensor to measure the weight of the products so that they can be added to the cloud database.

Push Button: Our system has two buttons. The green button is used to start the shopping cart. The red one is used when the shopper wants to remove a product from the cart. As the shopper pushes the button and scans the RFID tag, the RFID reader sends the data to delete that product from the item list.

Ultrasonic Sensor: Ultrasonic sensor is used to measure the depth of the cart so that it can detect whether the cart is full or not.

LCD Display: The LCD interfaced with the proposed system will show the total price of the products kept in the cart.

B. System Implementation

The customer has to press the green button to start the shopping cart. However, the customer must first adjust the camera to allow it to capture his/her face. After pressing the green button, the camera will start to capture photos. It takes one image every 5 seconds. After taking the picture, it checks if the customer is wearing a mask or not. This classification is done on the edge device. We have trained a classification model which can be executed on Raspberry Pi. If our device senses that the customer is not wearing a face mask, the buzzer will begin to beep, and the LCD will prompt the customer to do so.

RFID reader constantly searches for RFID signal. If it detects any RFID signal, it reads its details and then takes input from the load sensor. If the system finds that the load sensor load has changed according to the details, it uploads the relevant product details into the firebase cloud and then displays the total price in the LCD. The price of any product which is added to the cart according to this process will be added to the displayed amount of total cost. If the load sensor detects any changes without scanning the appropriate RFID tag, the buzzer will start buzzing, and the display will show that a product has been kept into the cart without scanning.

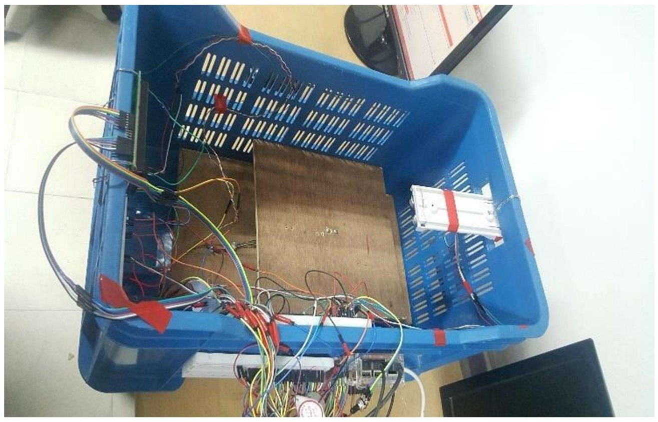


Fig. 1. Intelligent Shopping Cart

The red button is for the customers to return any product gain to the shop's shelves. If any user pushes the button and then takes a product out from the cart and scans it, the details of that product will be removed from the cloud server's database, and the LCD will show the deducted price. If the user's returned product's details do not match with the RFID readings, the buzzer will buzz, and the display will show that a wrong product has been lifted. By using this process, any user can remove an already registered product from the cart.

Suppose the shopping cart becomes full of products. In that case, the depth value from the ultrasonic sensor combined with the weight value will indicate the user for using another cart with the help of the display so that the user of the cart can know about the overload situation instantly.

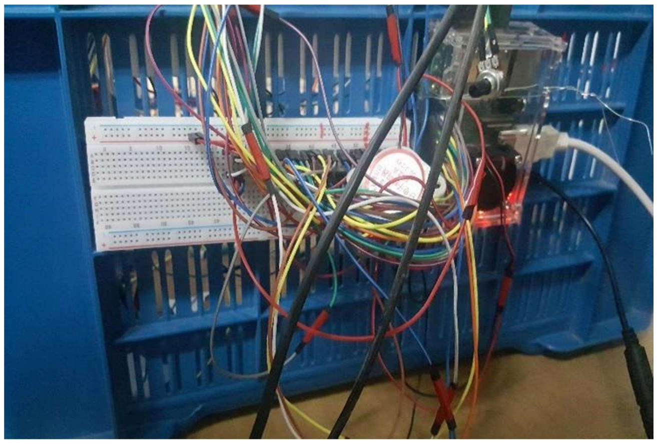


Fig. 2. Raspberry Pi & the Buzzer

C. Schematic View

The sensors and actuators of the proposed system are mainly interfaced with the Raspberry Pi. The data flow goes in and out from the Raspberry Pi itself. So, the center of the data flow is the Raspberry Pi, which can be easily understood from the schematic view. Fig. 3. Illustrates the schematic view of the proposed system.

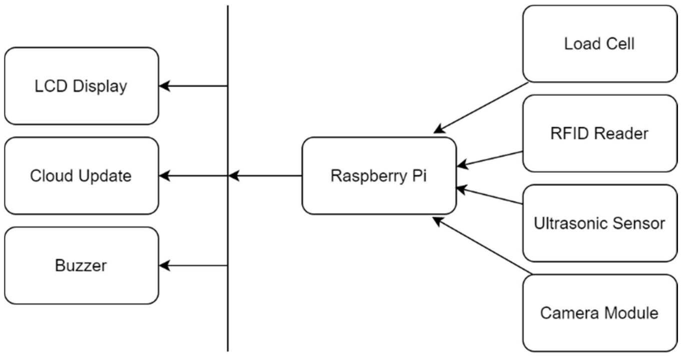


Fig. 3. Schematic View of EdgeCart.

In Google Firebase, we have implemented role-based authorization for each database table to ensure the privacy of the data we collect. We are not storing any images that will be captured through our edge device. Each image gets deleted after classification.

IV. System Process

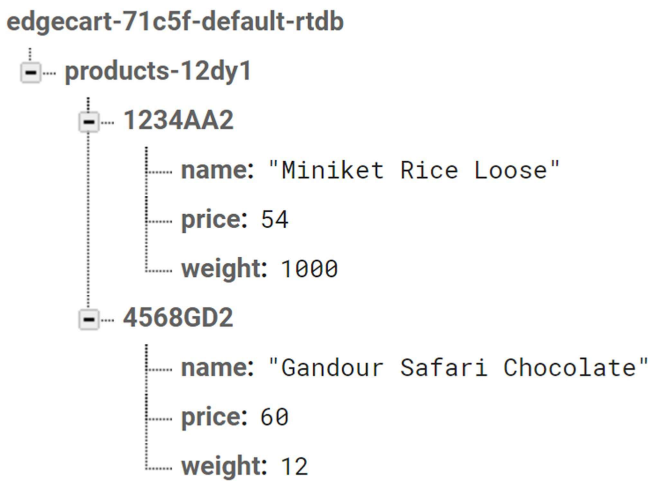
The proposed system constantly checks to see whether the consumer is wearing a mask and updating the cloud database according to the scanned products list.

A. Updating Cloud Database

The scanned products get instantly added to an online cloud database. The database is based on the NoSQL cloud database named Firebase, which is a service provided by Google Cloud Services.

This database has been implemented mainly using the REST API and the MQTT protocol of IoT so that each of the transactions requires less bandwidth and creates a minimum amount of bottleneck.

Fig. 4. Illustrates sample entry in the firebase cloud database. In this particular entry, it can be seen that it carries the information of the product name, product price, and product weight. In this proposed system, each product's RFID tag carries this information so that the system can process with all products total price and the cart's current legit weight.



```
edgecart-71c5f-default-rtdb
 ⊟ products-12dy1
    ⊟ 1234AA2
        name: "Miniket Rice Loose"
        price: 54
        weight: 1000
    ⊟ 4568GD2
        name: "Gandour Safari Chocolate"
        price: 60
        weight: 12
```

Fig. 4. Product Update in the Firebase Cloud Database.

B. Detecting Face Mask

Detection of face masks consists of three phases. They are:

a) *Create Face Mask Detection.*
b) *Face Detection.*
c) *Detect Mask on the Detected Face.*

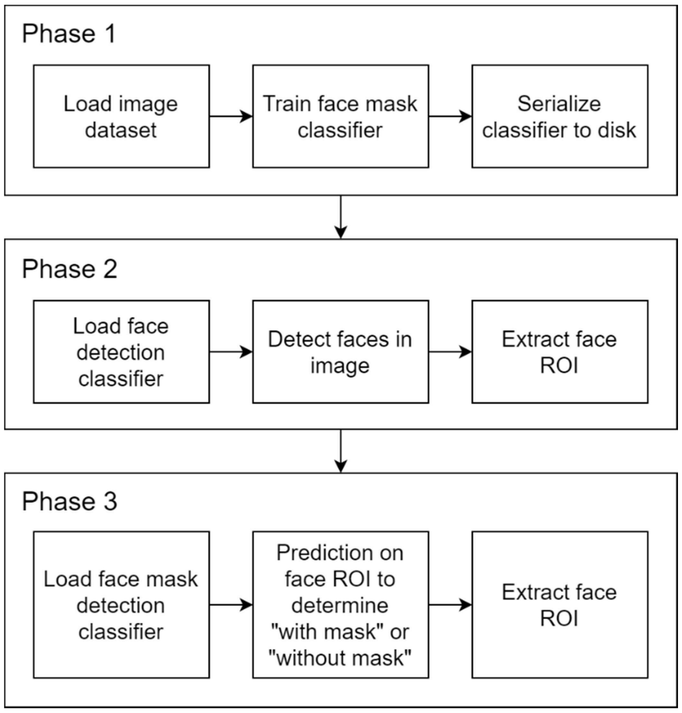

Phase 1
Load image dataset
Train face mask classifier
Serialize classifier to disk
Phase 2
Load face detection classifier
Detect faces in image
Extract face ROI
Phase 3
Load face mask detection classifier
Prediction on face ROI to determine "with mask" or "without mask"
Extract face ROI

Fig. 5. Face Mask Detection Phases.

To create a face mask detection model, we need a dataset to train our model. We have used the Real-World Masked Face Dataset (RMFD) [7] with some of our own collected images. This dataset has 2203 images with masks and 90468 images without the mask. However, we have chosen 1510 images with masks and 1510 images without the mask for training. For testing, we have chosen 380 images with masks and 380 images without the mask. Images are chosen randomly. Fig 6. shows the sample of our dataset.

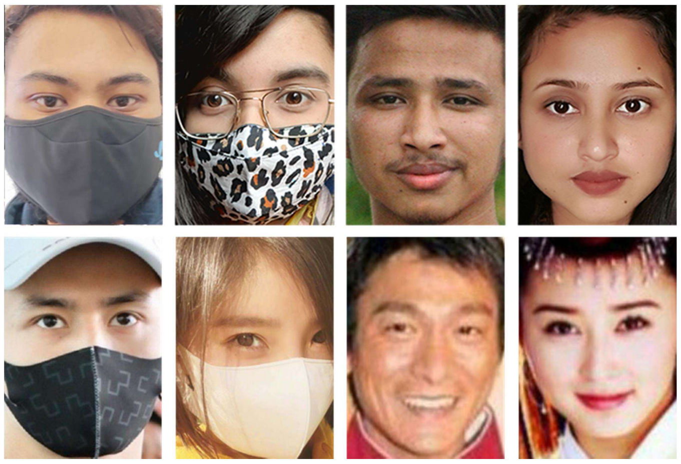


Fig. 6. Dataset Sample

We aim to develop a custom deep learning model to detect whether someone is wearing a mask or not. We have trained our classifier using Keras and TensorFlow. We have fine-tuned the MobileNetV2 [10] architecture to achieve this goal. This highly efficient architecture can be used on embedded devices with limited computing power, such as the Raspberry Pi.

First, we have set hyperparameter constants for our deep learning network. We have set the initial learning rate, the number of training epochs, and the batch size. After that, we have pre-processed our data. For that, we have resized all the images to 224x224 pixels. And then, we have converted the images to an array. Then we have normalized the pixel intensities of the images to the range [-1, 1] using the preprocess input function of Keras.

In order to enhance generalization, we have applied data augmentation to our images during training. Image data augmentation is a technique for artificially increasing a training dataset's size by modifying the dataset's images. The ImageDataGenerator class in the Keras deep learning neural network library allows us to fit models with image data augmentation. We have set random rotation, zoom, shear, shift, and flip parameters using the ImageDataGenerator class.

After that, we loaded MobileNetV2 with pre-trained ImageNet [11] weights and fine-tuned it. When we fine-tune a network, we remove the network's head. Following that, we build a new, fully-connected head and place it on top of the existing architecture. The network's base layers are then frozen. These base layers weights will not be changed during backpropagation, but the head layers weights will.

Then we have compiled our model using Adam [12] optimizer. After compiling, we have trained our model on the training dataset. Then we have quantized our model and converted it to a TensorFlow Lite model to minimize model size while also enhancing CPU and hardware accelerator latency. Finally, we have serialized our model to the disk.

One of the most fundamental elements of computer vision is face detection. We have tried pre-trained classifiers like Haar Cascades, Multi-Task Cascaded Convolutional Neural Network (MTCNN), Dlib Frontal Face Detector, and DNN Face Detector in OpenCV for detecting faces.

Haar Cascades: This method was proposed by Paul Viola and Micheal Jones in 2001 [8]. It is a machine-learning technique in which a cascade function is trained using many positive and negative images. It is capable of extracting a large number of features from images. Adaboost is then used to choose the best functions. The original 160000+ features are now reduced to 6000. However, integrating all of these features into a sliding window can take a long time. As a consequence, they implemented a Cascade of Classifiers, which groups the features. If a window fails at the first level, the cascade's remaining features are ignored. If it passes, the next function will be checked, and the process will be repeated.

Multi-task Cascaded Convolutional Neural Network (MTCNN): In 2016 Zhang, et al. [9] introduced this model. It can detect five key points along with the face. It uses a three-stage CNN cascade structure. To obtain candidate windows and their bounding box regression vectors, they first use a convolutional network. On-maximum suppression (NMS) is used to overlap heavily overlapped candidates. Following that, these candidates are sent to another CNN, which filters out many false positives and calibrates bounding boxes.

Dlib Frontal Face Detector: It is a C++ toolkit. Nevertheless, it has python bindings to run it in python. The dlib frontal face detector uses Histogram of Oriented Gradients (HOG) to extract features, which are then passed through a Support Vector Machine (SVM). The distribution of gradient directions is used as features in the HOG feature descriptor.The dlib library includes an implementation of Kazemi and Sullivan's paper [13] on facial landmark detection.

DNN Face Detector in OpenCV: In 2017, OpenCV introduced a highly improved "deep neural networks" (dnn) module. It supports Caffe, TensorFlow, and Torch deep learning framework. Aleksandr Rybnikov has put in much effort to make this module a reality. He has included a more accurate deep learning-based face detector with OpenCV. The Single Shot Detector (SSD) architecture with a ResNet base network is the basis for OpenCV's deep learning face detector. It also comes with a quantized Tensorflow version.

From our testing, we have found that Haar Cascades performed the worst. Multi-task Cascaded Convolutional Neural Network (MTCNN) and Dlib Frontal Face Detector had performed head-to-head. Moreover, the DNN Face Detector in OpenCV performed better than all of them. The best part of this method is, it is best for detecting a face from different angles. That is why we are using DNN Face Detector in OpenCV for face detecting in our system. As our approach is based on edge computing concept and using Raspberry Pi as our edge device, we have used the model's quantized version.

Now that our face mask detector is trained and we have found an efficient way of face detection from images, we can detect if a person is wearing a mask or not. We have transferred the face detector and mask detector model to Raspberry Pi. Then we have loaded the models in memory. Our Raspberry Pi captures an image every 5 seconds. After capturing an image, we have pre-processed the image for face detection. We have resized the image to 300x300 pixels and performed mean subtruction on the image. Then we have performed face detection on the image to localize the faces in the image. Before extracting the ROI (Region of Interest), we have ensured they meet the confidence threshold. Then we have pre-processed the ROI the same way we have done for the training dataset. The ROI was then used to perform mask detection to determine whether the face was wearing a mask.

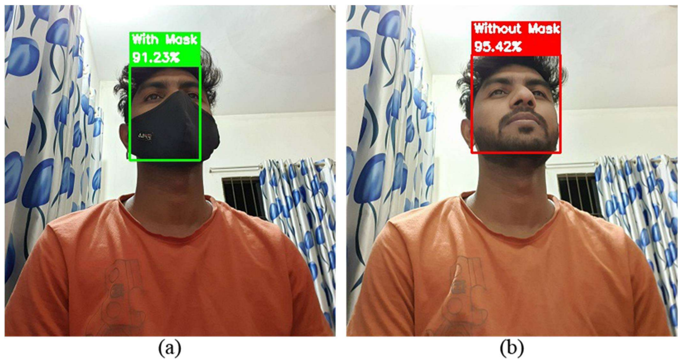


Fig. 7. (a) Predicted "with mask",(b) Predicted "without mask"

V. Experimental Outcome

We have performed prediction on our test dataset. The prediction result is pretty good. We have got 98.98% accuracy, and the loss is 4.15%.

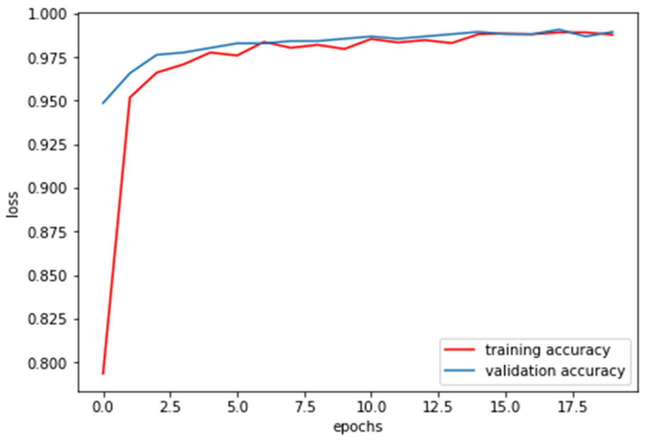


Fig. 8. Training & Validation Accuracy Curve

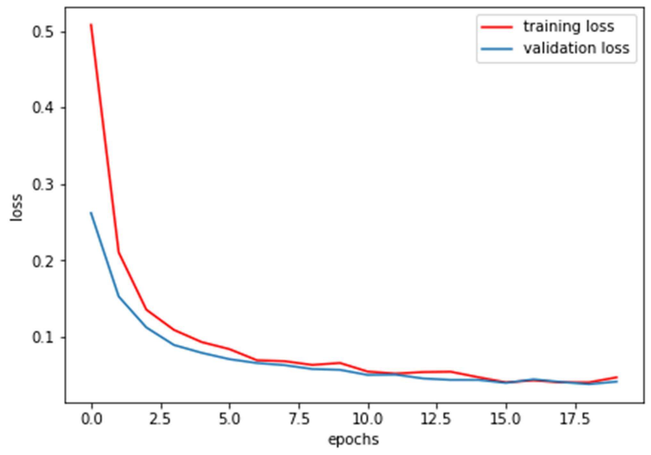


Fig. 9. Training & Validation Loss Curve

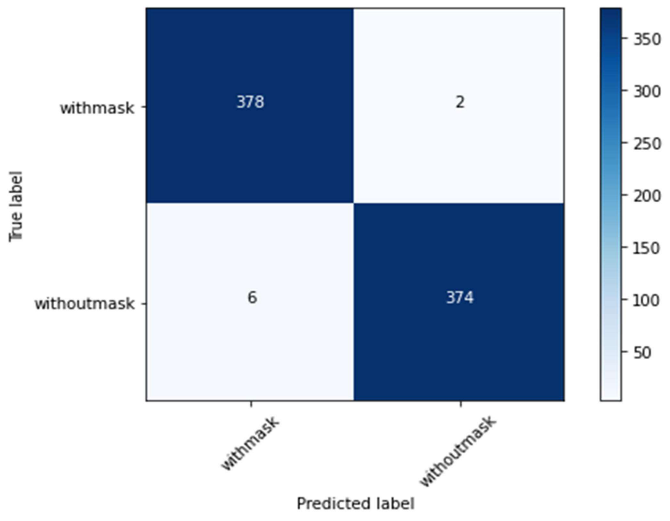


Fig. 10. Confusion Matrix

Our face mask classification model's training accuracy is visualized in Fig. 8. Our model achieves optimal output after ten epochs and retains constant learning, as can be shown. Fig. 9 shows our training loss curve. We can see that loss is consistent after 15 epochs. By examining both the curves, we can say that there is no sign of overfitting in our model. Fig. 10 depicts our classification model's confusion matrix. It shows that our model has wrongly classified eight images out of 760 images.

TABLE I. Classification Report

	precision	recall	f1-score	support
withmask	0.98	0.99	0.99	380
withoutmask	0.99	0.98	0.99	580
accuracy			0.99	760
macro avg	0.99	0.99	0.99	760
weighted avg	0.99	0.99	0.99	760

Table I shows the classification report of our classification model.

VI. Conclusion and Future Scope

A. Conclusion

The latest outbreaks of global pandemics have taught us that it is vital to wear masks all the time while roaming in public places, and it is necessary to spend as less time as possible in crowded places like supermarkets and shopping malls to reduce health risks. In these places, customers have to stand in long lines with other customers to scan their goods before the payment, which is time-consuming, and maintaining distance at the product scanning queue on peak hours is close to impossible. Ensuring everyone is wearing masks in these circumstances is a highly complicated task.

That is why we have proposed EdgeCart, a smart cart that allows people to scan their own goods. It will cut down the amount of time spent in the long lines for product scanning. Furthermore, our smart cart will warn customers if they are not wearing a face mask, creating instant awareness among the people surrounding that place. We have achieved excellent accuracy in detecting face masks, but it is difficult for the proposed intelligent surveillance system to detect masks from awkward angles. Our proposed face detection model generates less accuracy while extracting faces from awkward angles. Another constraint of the proposed system is that it is not dominantly a shop-lifting proof. Besides these nominal constraints, the EdgeCart has been able to successfully maintain the mask surveillance with a convenient and faster shopping experience for the shoppers.

B. Future Scope

We have plans to add a sneezing and coughing detection feature to the proposed EdgeCart in the future, which will be effective for the health surveillance system as a whole. Our EdgeCart will use a microphone array to listen for sneezes and coughs, and our trained deep neural network will classify whether it is sneezing/coughing or anything else. Thermal imaging techniques can also be used to monitor the subjects. The camera sensor can be used as well to effectively detect shop-lifting with the use of Computer Vision at the edge.

VII. References

[1] P. Chandrasekar, "Smart shopping cart with automatic billing system through RFID and ZigBee" in Information Communication and Embedded Systems (ICICES), 2014 International Conference, 2014.

[2] Hsin-Han Chiang, "Development of smart shopping carts with customer-oriented service" in System Science and Engineering (ICSSE), 2016 International Conference, 2016

[3] Ruinian Li, "IoT applications on Secure Smart Shopping System" in IEEE Internet of Things Journal 2017.

[4] Xianghua Fan, Fuyou Zhang, Haixia Wang and Xiao Lu, "The system of face detection based on OpenCV," 2012 24th Chinese Control and Decision Conference (CCDC), Taiyuan, China, 2012, pp. 648-651, doi: 10.1109/CCDC.2012.6242980.

[5] Rein-Lien Hsu, M. Abdel-Mottaleb and A. K. Jain, "Face detection in color images," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 5, pp. 696-706, May 2002, doi: 10.1109/34.1000242.

[6] M. Jiang, X. Fan and H. Yan, "RetinaMask: A Face Mask detector", 2020, [online] Available: http://arxiv.org/abs/2005.03950.

[7] X-zhangyang. "Real-World Masked Face Dataset, RMFD" GitHub, 13-02-2020, https://github.com/X-zhangyang/Real-World-Masked-Face-Dataset.

[8] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Kauai, HI, USA, 2001, pp. I-I, doi: 10.1109/CVPR.2001.990517.

[9] [1] K. Zhang, Z. Zhang, Z. Li and Y. Qiao, "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks," in IEEE Signal Processing Letters, vol. 23, no. 10, pp. 1499-1503, Oct. 2016, doi: 10.1109/LSP.2016.2603342.

[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510-4520, doi: 10.1109/CVPR.2018.00474.

[11] J. Deng, W. Dong, R. Socher, L. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

[12] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[13] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1867-1874, doi: 10.1109/CVPR.2014.241..