COVID-19 FACE COVER TRACKER USING AMAZON WEB SERVICES

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Rahmanul Hoque

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

April 2021

Fargo, North Dakota

North Dakota State University
Graduate School

Title

COVID-19 FACE COVER TRACKER USING AMAZON WEB SERVICES

By

Rahmanul Hoque

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Simone Ludwig
Chair

Dr. Anne Denton

Dr. María de los Ángeles Alfonseca-Cubero

Approved:

7/13/2021
Date

Dr. Simone Ludwig
Department Chair

ABSTRACT

Numerous studies have shown that wearing face covers and following other social distancing guidelines reduce community transmission of air borne diseases like COVID-19 during a pandemic. While public health administrators can provide guidelines on wearing masks and social distancing, it cannot be guaranteed that everyone will follow those guidelines. It is very important to have a mechanism available for the public health administration to collect data on how well the population is following the guidelines provided. This information will allow them to anticipate when there will be peak hospitalization and prepare accordingly. This paper aims to look at the feasibility of using machine learning and image processing techniques to track the number of people wearing a face cover in a crowd in real time. The Amazon DeepLens camera, Amazon Rekognition and other Amazon Web Services have been used to make inferences and collect data.

ACKNOWLEDGEMENTS

First and foremost, I would like to take this opportunity to thank my adviser, Dr. Simone Ludwig, who gave me the opportunity to work on the DeepLens project. She provided me with valuable advice and timely feedback on every step of the project and helped me complete the paper on time. I would also like thank my committee members, Dr. María de los Ángeles Alfonseca-Cubero and Dr. Anne Denton, for their valuable time and interest in my paper.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

While there has been some controversy with the use of masks in public spaces, the US Centers for Disease Control and Prevention (CDC) has been recommending the public to wear face cover. Different model simulations based on data gathered from states like New York and Washington showed that community transmission of diseases like COVID-19 can be significantly reduced by wearing face masks. According to Eikenberry et al., other studies have shown that masks protect individuals from contracting and transmitting various infections. The authors concluded based on their data that face masks should be adopted nationwide to prevent the spread of the disease during a pandemic [1].

However, this may not be a universal solution as a sizable portion of the population tend to not wear masks and some states would not implement a mask mandate in public places. This can be attributed to two key factors. The first one is that some people do not believe that masks would prevent transmission of COVID-19. The other factor is psychological reactance, where people show a resistance to follow through when being forced to do something, i.e., wearing a mask in this case [2]. Hence, it is necessary to find a mechanism to monitor whether people are wearing a mask and following other social distancing guidelines so that the next spike in infection can be predicted. This way, hospitals and local government officials can prepare accordingly. Using machine learning and cloud computing technology like Deep Learning and Amazon Web Services to monitor the percentage of population following mask guidelines could be a good way of predicting the regions that may see a spike in infection rates.

1.1. Machine Learning

Machine learning is a part of computer science, which is associated with algorithms that act like human intelligence by gathering information from its surroundings. It is a constantly

evolving field. Different machine learning techniques have been used with great results in several areas ranging from pattern recognition, computer vision, spacecraft engineering, finance, entertainment, computational biology to biomedical and medical applications, etc. Machine learning algorithms have developed the ability to learn from a particular context that they are provided with and apply it into more general unseen tasks [3]. Decrease in the cost of computation and large volume of online data becoming available has been driving the progress of machine learning in the recent past [5].

1.2. Neural Network

A neural network is created by mimicking the concept of a human brain where billions of interconnected neurons process information parallelly. A typical neural network consists of an input layer of neurons, one or more hidden layers, and finally a layer of output neurons. Figure 1 shows the typical architecture of a simple neural network. The lines connecting neurons represent a numeric value called weight. The output, h_i, of neuron i in the hidden layer is calculated as follows:

$$h_i = \sigma(\sum_{j=1}^{N} V_{ij} x_j + T_i^{hid})$$

Here, σ() is the activation function, and N the represents the number of input neurons. V_{ij} represents the weights, x_j inputs to the input neurons, and T_i^{hid} the threshold term of the hidden neurons. The activation function contributes to the nonlinearity in the network. However, its primary purpose is to bound the value of the neuron in order to prevent the neural network from getting distorted by divergent neurons [4].

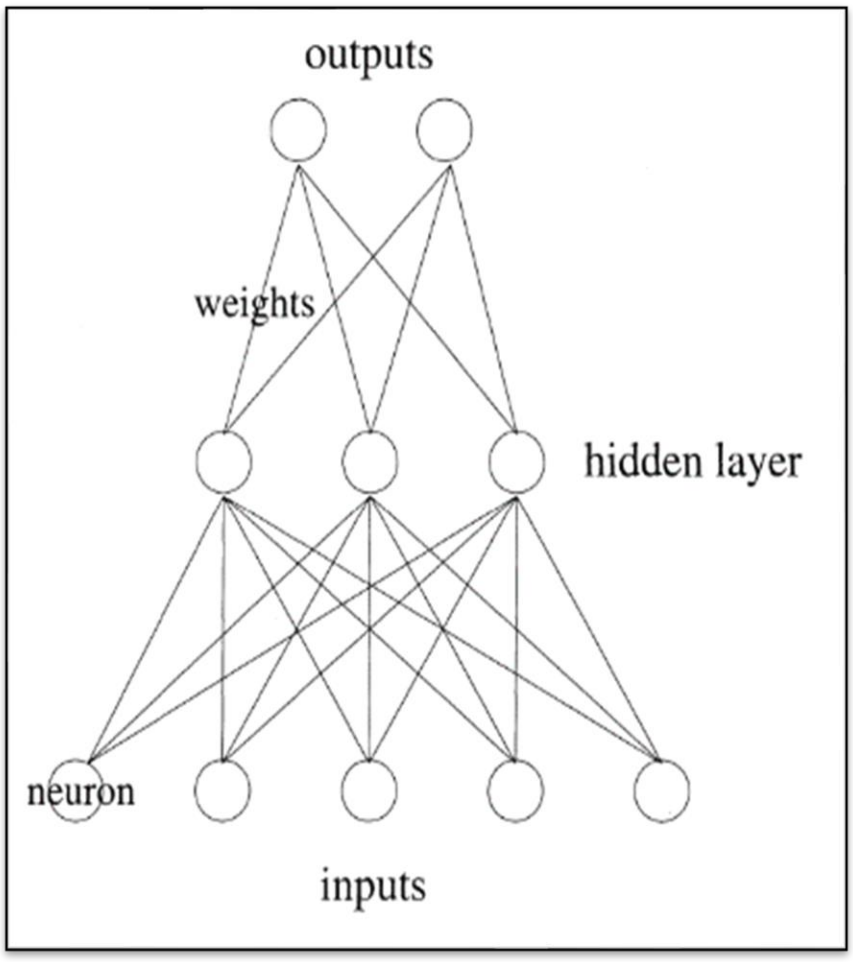

Figure 1: Structure of a Neural Network[1]

Neural networks with the above architecture have proven to be able to approximate various computable functions accurately. The numbers that the input neurons receive are independent variables. The numbers that are returned from the neurons in the output layer are, however, dependent on the function that has been approximated by the network. The input and output values can either be binary or symbols like green, red, etc. based on how the data was set up. This capability allows neural networks to be very effective [4].

[1] Source: https://doi.org/10.1007/978-1-4615-0377-4_5

1.3. Deep Learning

Deep learning is a type of machine learning that uses a deep graph with multiple processing layers to learn data representations with multiple levels of abstraction. It is based on a neural network that has more than three layers. It has contributed greatly towards the improvement in speech recognition, visual object recognition, object detection, and other fields such as genomics. It utilizes a technique called backpropagation to detect intricate patterns in large data sets with high dimensions by indicating how a machine adjusts its internal parameters. This is then used to process the representation in each layer from the representation in the previous layer [6].

1.4. Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is one of the most widely used deep neural networks. This is due to its ability of processing large amount of data and impressive performance in solving machine learning problems, especially ones that involve image data, computer vision and natural language processing (NLP). Linear operations between matrices are called convolution in mathematics, where CNN gets its name from. It comprises of multiple layers including convolutional layer, non-linearity layer and pooling layer. Convolutional layer reflects the key content in an image, while the non-linearity layer acquires the non-linear transformation and finally the pooling layers works to reduce the dimensions of the feature map [7]. Figure 2 shows the process diagram of CNN.

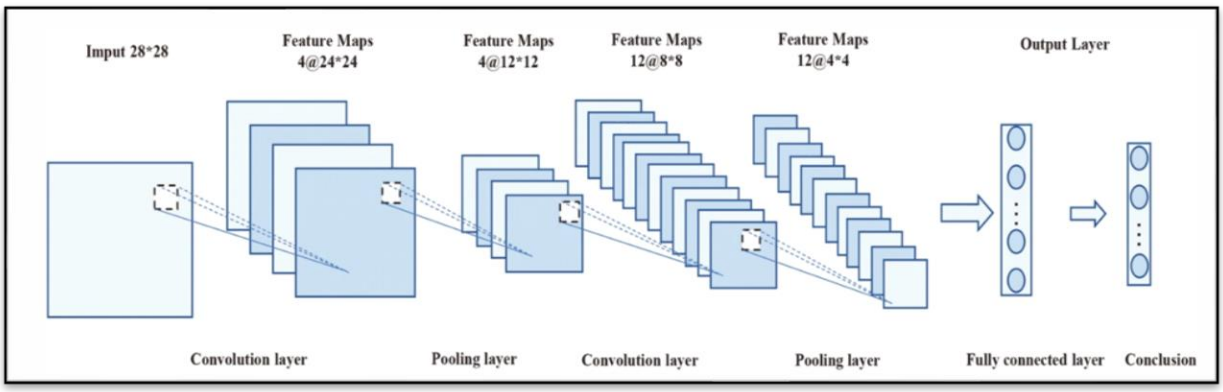

Figure 2: Convolutional Neural Network Architecture[2]

1.5. ResNet-50

Resnet, which uses the residual learning framework, is often used when there are significantly more layers in the neural network needed. It addresses the issue with the difficulty of training deeper neural network by viewing the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. It is proven to be more optimizable and provide accuracy from significantly increased depth [7]. ResNet-50 is ideal for image classification and feature extraction. Residual building block (RBB) is one of the key elements in ResNet-50. It is based on the concept of skipping blocks of convolutional layers by utilizing shortcut connections. These shortcuts allow the optimization of trainable parameters in error backpropagation so that vanishing gradients problem can be avoided. Hence, deeper CNN structures can be created to improve the overall performance [8].

1.6. AWS DeepLens

AWS DeepLens is a video camera with deep learning capabilities which, integrated with numerous machine-learning services, can make local inference against deployed models trained in AWSCloud. It can be used to develop computer vision applications using the latest artificial

[2] Source: https://link.springer.com/article/10.1007/s11554-019-00915-5

intelligence (AI) tools and technology based on a deep learning model [4]. Table 1 shows the specifications for the AWS DeepLens Camera.

Table 1: AWS DeepLens Camera Specifications

Camera	A 4-megapixel camera with MJPEG (Motion JPEG)
Memory	8 GB of on-board memory
Storage	16 GB of storage capacity
Storage	A 32-GB SD (Secure Digital) card
Network	Wi-Fi support for both 2.4 GHz and 5 GHz standard dual-band networking
Display	A micro-HDMI display port
Audio	Audio out and USB ports
Power consumption	20 W
Power input	5V and 4Amps

The AWS DeepLens device runs on Ubuntu OS-16.04 LTS and supports the current versions of software for the following frameworks:

- The AWS DeepLens device support Python 2.7 and Python 3.7.
- The supported deep learning AI architectures are MXNet 1.6.0 and TensorFlow 1.4

Figure 3 shows the workflow of an AWS DeepLens project.

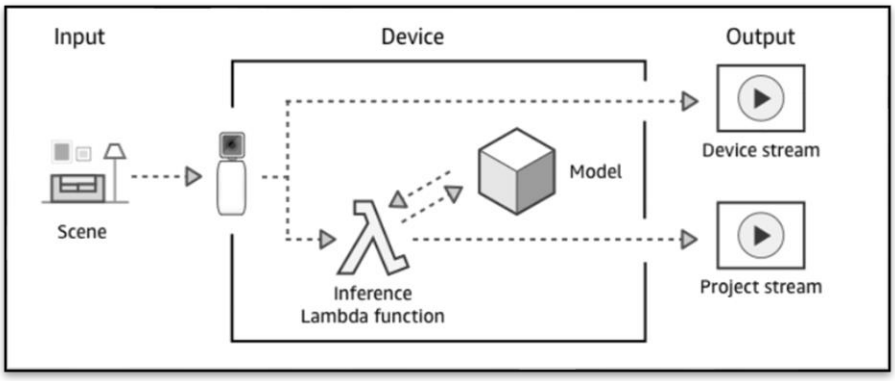

Figure 3: DeepLens Project Overview[3]

[3] Source: https://docs.aws.amazon.com/DeepLens/latest/dg/DeepLens-dg.pdf

The camera captures a video stream and produces two output steams, the device stream, and the project stream. The device stream is the video stream without any processing done while the project steam is a result of the model processing video frames. The unprocessed video frames are transferred by the Inference Lambda function into the model deployed to the camera. The frames are then processed and transferred back to the Inference Lambda function which in turn displays them in the project stream [9].

2. RELATED WORK

Researchers have tried utilizing different machine learning approaches in detecting the use of face masks and social distancing. Loey et al. used Resnet50 for feature extraction and used decision trees, Support Vector Machine (SVM), and ensemble algorithm for classification. They have achieved high level of testing accuracy using the SVM classifier with 9.64%, 99.49%, and 100% testing accuracies on three of the datasets that they have used, respectively. However, this approach only tests the algorithm on simulated still images and it is unclear how it will work in real life setting when connected to a recording device [10].

One other approach is the use of transfer learning model. It utilizes a pre-trained deep learning model, InceptionV3, to detect individuals not wearing a mask in public places. InceptionV3 is a convolutional neural network (CNN) developed by Google and comprises of 48 layers. Chowdary et al. have modified the InceptionV3 CNN by removing the last layer and adding 5 more layers which include pooling layer, and a decisive dense layer with softmax function to classify an individual with or without a mask [11]. The researchers in the paper utilized a technique called image augmentation to increase their dataset by artificially modifying their images. They have applied the following operations on an existing image: shearing, contrasting, flipping horizontally, rotating, zooming, and blurring. They have trained the new transfer learning model for 80 epochs. Their model achieved accuracies of 99.92%, 99.9% during training, and 100%, 100% during testing. While the results are very promising, there is no data on how this algorithm would work on making inferences in real time just like the previous paper discussed [11].

While the DeepLens camera has been rarely used for facemask detection, there are several papers that have leveraged both Amazon Webs Services and the DeepLens Camera to solve other

real-world problems. [19] is one such example, where the authors use Amazon Web Services and AWS DeepLens camera to detect diseases in plants.

There are several Machine Learning models available for classifying diseases in a plant leaf. But at times they fall short of providing true value due to the lack of integration with sophisticated hardware. Khan et al. addresses this issue by using the AWS DeepLens camera. They utilize the scalability of AWS and transfer learning by training their model in AWS SageMaker and then deploying it to their AWS DeepLens camera. They have achieved 98.78% accuracy with their model while detecting 25 separate diseases in real-time. The camera was able to detect the diseases in Apple, Grape, Peach, Potato, Strawberry and Tomato plants. Using the DeepLens camera, it took their deployed model only a fraction of a second to diagnose diseases in an image with an average time of 0.349s [19].

Another paper titled "The Hygiene Monitoring System" leverages the AWS DeepLens Camera quite well. They used deep learning to detect whether individuals were washing their hands properly. They gathered video footages and divided the hand motions into separate frames and uploaded them to AWS. They retrained a pretrained neural network using transfer learning. After training their model, they tested it using validation data from original subjects and new subjects. Then they deployed the model to the AWS DeepLens Camera and placed it in front of live video feed. They were able to gain 75% accuracy with instantaneous result [22].

In [23], Rafael et al. uses the DeepLens Camera and other Amazon Web Services to detect facial expression of individuals to provide a better mode of communication to blind people through audio signals. They compared the performance of the camera to on-premises deep learning. Their approach allowed them to achieve results with a 76.16% accuracy. They concluded that it is a

viable approach to use AWS and DeepLens camera although it may require high level expertise to yield better results[23].

3. APPROACH

For our initial person detection, we used DeepLens-face-detection, a pretrained model from Amazon. It is trained using MxNet and ResNet 50. ResNet-50 is a convolutional neural network that is 50 layers deep [12]. The DeepLens-face-detection model has been deployed to the AWS DeepLens where it acts as part of the Inference Lambda Function. Once the function detects and labels a person or multiple people in the captured frames, the image is uploaded to the Amazon S3 Bucket [9] as shown in Figure 4.

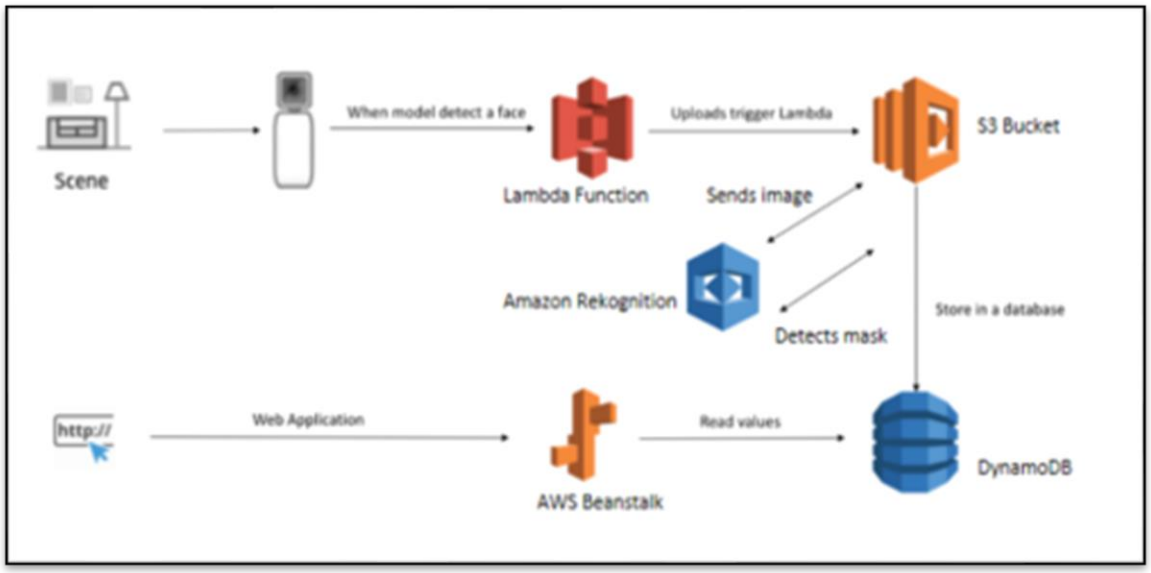

Figure 4: AWS Project Overview[4]

3.1. Amazon S3

Amazon S3, also known as Amazon Simple Storage Service, is an object storage service provided by Amazon as a part of its AWS platform. It allows customers to securely store data ranging from websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices to big data analytics [13]. Images captured by the DeepLens camera will be

[4] Source:https://aws.amazon.com/blogs/machine-learning/track-the-number-of-coffees-consumed-using-aws-deeplen

automatically uploaded and store in the S3 bucket before they are analyzed by Amazon Rekognition.

3.2. Amazon Rekognition

Amazon Rekognition is a service provided by Amazon that enables us to add powerful visual analysis to applications. It allows one to build applications with the capabilities of searching, verifying and organizing millions of images. Rekognition consists of two services, Rekognition Video and Rekognition Image. For our application, we have used Rekognition Image to detect and analyze images uploaded by our DeepLens Camera [14].

Rekognition Image provides different capabilities ranging from object detection and face recognition to text extraction and content analysis. It is based on the Deep Learning Technology developed by Amazon's computer vision scientists for analyzing images for Prime Photos. It uses deep neural network models to detect and label objects and scenes in images. Amazon Rekognition is part of the Amazon AI services that uses deep learning for image processing and computer vision along with other tasks such as speech recognition and natural language processing. Along with other deep learning architectures, Convolutional Neural Network (CNN) is being used to achieve desired results on the domains [14]. Line 23 in Figure 5 shows how the Rekognition API is called using the image file as a parameter.

3.3. Lambda Function

AWS Lambda is part of Amazon Web Services, which involves providing computational services to customers. It allows one to run code without having to manage servers. It is highly scalable; it can scale from a handful of requests per day to thousands of requests per second automatically.

```python
import boto3
import time
import uuid

s3 = boto3.client('s3')
dynamodb = boto3.client('dynamodb')
rekognition = boto3.client('rekognition')


def lambda_handler(event, context):

    utime = str(int(time.time())) #Current Unix Time

    bucket = event['Records'][0]['s3']['bucket']['name']
    key = event['Records'][0]['s3']['object']['key']
    image = {
        'S3Object': {
            'Bucket': bucket,
            'Name': key,
            }
        }

    response = rekognition.detect_protective_equipment(Image=image)

    print(response)
    persons = response['Persons']
    for person in persons:

        if person['BodyParts']:

            bodyPart = person['BodyParts'][0]

            if(bodyPart['Name'] =='FACE'):

                if bodyPart['EquipmentDetections']:

                    dynamodb.put_item(
                        TableName='Face_Covering',
                        Item={
                            'personID': {'S': str(uuid.uuid1())},
                            'confidence': {'S': str(bodyPart['EquipmentDetections'][0]['CoversBodyPart']['Confidence'])},
                            'date_time': {'S': str(utime)},
                            'face_cover': {'S': str(bodyPart['EquipmentDetections'][0]['CoversBodyPart']['Value'])}
                        })
                else:
                    dynamodb.put_item(
                        TableName='Face_Covering',
                        Item={
                            'personID': {'S': str(uuid.uuid1())},
                            'confidence': {'S': str('')},
                            'date_time': {'S': str(utime)},
                            'face_cover': {'S': str('False')}
                        })
```

Figure 5: Code Snippet of Lambda Function in Python

AWS Lambda supports code for almost any type of application or backend services possible. It manages and performs all of the necessary maintenance including server and operating systems, scaling, code monitoring, and logging. Lambda can be used to run code in response to events such as changes to data in Amazon Simple Storage Service (Amazon S3) bucket or an Amazon DynamoDB table. In our application, images uploaded to the S3 bucket triggers the

Lambda function. Figure 5 shows the code snippet for one of the lambda functions used for this application [23].

3.4. Amazon DynamoDB

AWS DynamoDB is a database service provided by Amazon as a part of Amazon Web Services. It is a NoSQL database service which supports a key-value and document database [15]. For this paper, a DynamoDB table named Face_Covering is created with four attributes: personID, confidence, date_time and face_cover. The personID gets populated by a universally unique identifier (UUID) generated in the lambda function as shown in line 40 of Figure 5. Confidence is the confidence level or the degree of certainty to which the Rekognition API decides a person is wearing a mask. This value is retrieved from the json object from the response of the API call. The date_time field is populated with the unixtime during which the API was called. The Boolean variable face_cover tells is an indicator of whether the face detected has a mask on or not. Figure 6 shows how the data in the Face_Covering table is arranged.

personID	confidence	date_time	face_cover
0a7fa6bc-38d9-11eb-9569-66524a29c703	97.61018371582031	1607379038	True
664cdcd0-38de-11eb-a4a3-6e54a255c33b	86.67153930664062	1607381339	True
a95ea2a0-38df-11eb-84e6-8af2cf6cb8bf	94.46965789794922	1607381881	True

Figure 6: Face_Covering Table

3.5. JSON Response

The reply after calling the Amazon Rekognition API is a json object, which consists of the information of whether a personal protective equipment is detected or not and the confidence level. Figure 7 shows how a typical response would be like if no face mask is detected:

Figure 7: Json Object without Face Cover

We parse the json reply and look for the EquipmentDetections array. If the array is empty, the person is not wearing a face cover and we update the Face_Cover table with a new entry with a unique id. We set the Face_Cover attribute to 'False' and update the confidence attribute. If the EquipmentDetections object is not empty, we look for the 'Face_Cover' attribute within EquipmentDetections and see if the value is set to True and has a Confidence number associated with it. If we can find those values, we update the Face_Cover table with a new entry with a unique id. We set the Face_Cover attribute as true and update the confidence attribute. Figure 8 shows a typical json reply when face cover is detected.

Figure 8: Json Object with Face Cover

4. EXPERIMENTS AND RESULTS

Initially, we test the Amazon Rekognition API by uploading a few images manually to AWS S3 Bucket. We find very promising results as the API detects all the people in the first image (Figure 9) with masks.

Figure 9: First Uploaded Image[5]

The second image (Figure 10), which consists of individuals with no mask, is then uploaded and Amazon Rekognition detects 11 individuals out of 17 who are visible. Not all of the individuals were detected as some have their faces either turned to the opposite direction or they are too far away.

[5] Source: https://www.ndsu.edu/healthprofessions/news/detail/59223/

Figure 10: Second Uploaded Image[6]

We then move on to testing API on a video file. A 22:16 minutes long video has been selected for the purpose of testing the application. It is a video of people walking down the streets of Seoul, South Korea. Before the test is being run, the video is played to count N_1, the total number of faces that appear, simply by observing. Then, it is played again from the beginning to count the number of people who appear to be wearing a face mask, N_2. The number of people that are not wearing a face cover, N_3, is then calculated by subtracting N_2 from N_1:

$$N_3 = N_1 - N_2$$

There are certain assumptions that were made while counting the sample size. People who are facing forward with the back of their head facing the camera are not counted as the DeepLens camera will not detect them unless it sees a face.

Also, some people may not have been counted as they could be missed due to the latency in human reaction time. The mean reaction time to detect visual stimuli is approximately 180–200

[6] Source: https://www.ndsu.edu/news/story_archive/stembuildingopens/

milliseconds [16]. For this reason, three different observations were made at three different times and only the average is used for the later comparisons. Table 2 shows the data that has been gathered by simply observing and counting.

Table 2: Population Data

	Sample Size (N_1)	Number of people wearing a face cover (N_2)	Number of people not wearing a face cover (N_3)
Observation 1	683	249	434
Observation 2	698	233	465
Observation 3	660	227	433
Average	≈ 680	≈ 236	≈ 444

We then test our application, which integrates the DeepLens camera and the Rekognition API. As discussed earlier, there are two different models at play. One is the pretrained person-detection model, which is deployed locally to the camera and the other one is Amazon Rekognition PPE detection model, which resides in the cloud. The camera is placed in front of a screen where the video will be played. If any face appears on the screen, the camera will detect it with the help of the person-detection model. Once the face is detected, the camera will take a snapshot and upload the image to the S3 bucket. Once uploaded to the S3 bucket, the lambda function will invoke the Rekognition API to analyze the image.

The video is then played again after turning the DeepLens camera on. Figure 11 and 12 show two of the several images captured by the DeepLens camera. The first figure shows a person with a mask and the second shows one without a mask. While the camera detected only one face initially, the Rekognition API is able to detect up to fifteen people in an image once it is uploaded to the S3 Bucket. Once uploaded, the Lambda function was called, and it analyzed the image and updated the database.

Figure 11: Person with a Face Mask

Figure 12: Person without a Face Mask

4.1. Results

For the first trial with the video playing in regular speed, a total of 78 people were detected. 76% of the 78 people were not wearing a face cover, which amounts to 59 people. The median degree of confidence is 96.75. 24% were detected with a face covering, which amounts to 19

people with a median degree of confidence of 98.35. Figures 13 and 14 show the data visualization of the first trial.

The subjects in the video appear to be moving too fast for the DeepLens camera at times. This is because the person recording the video is moving constantly and this increases the relative speed of the people coming from the opposite direction. In order to compensate for this, we reduced the playback speed by half and the results have improved slightly.

For the second trial with the video playing in 0.5 times the regular speed, a total of 100 people were detected. 73% of the people were detected to be not wearing a face cover with a median degree of confidence of 99.97. 27% were detected to be wearing a mask with a median degree of confidence of 97.16%. It is evident that the population size in each category has increased. While the degree of confidence for people not wearing a face cover has improved, there is a slight dip in the median degree of confidence in cases where face covers were detected. Figure 15 and Figure 16 show the data visualization of second trial.

It appears that the person detection model is slowing down the upload process of each image significantly. So, we also test a different approach where the camera would capture images at fixed intervals regardless of whether there is a person in the image or not. While the camera could be set up to do that, we decided to emulate it for the purpose of this paper by simply splitting the video file into 492 images at 1 frame per 2.7 seconds and upload it to the AWS S3 Bucket manually. The Face_Covering table in the database gets populated with new values. We get 833 unique entries, which means 833 individuals were detected. Of those, 70% or 583 were detected to be not wearing a mask while 30 % or 250 were detected wearing a face mask. Figure 17 shows the visual representation of the data. There is an improvement in the degree of confidence for the

final trial as shows in Figure 18. The median degree of confidence is 98.78 for the true values while 98.26 for the false values.

One noticeable difference with the final trial is that there are significantly more entries, 153 to be exact, compared to the number counted manually. This increase in number could be because there are duplicate entries in the database for the same people as they may appear in multiple frames. This is one of the drawbacks of just going with the image capture approach. The other major drawback is that there will be a very large number of images uploaded, which will incur significant cost. To put things into perspective, there are 86,400 seconds in a day and if the camera were to upload an image every three seconds, 28,800 images would be uploaded per day.

4.2. Diagrams

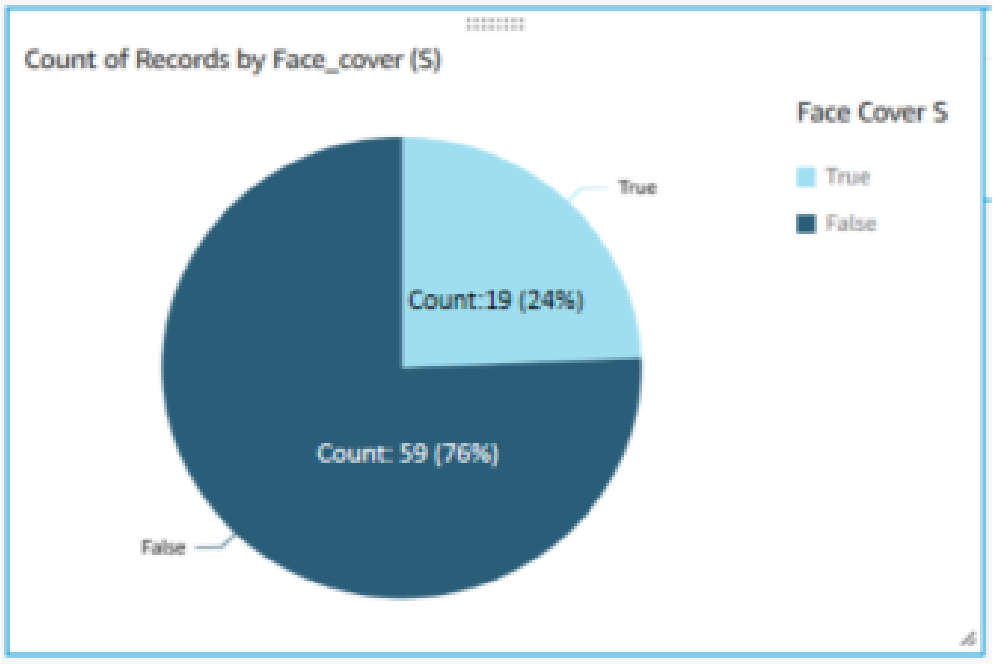

Figure 13: Results of First Trial

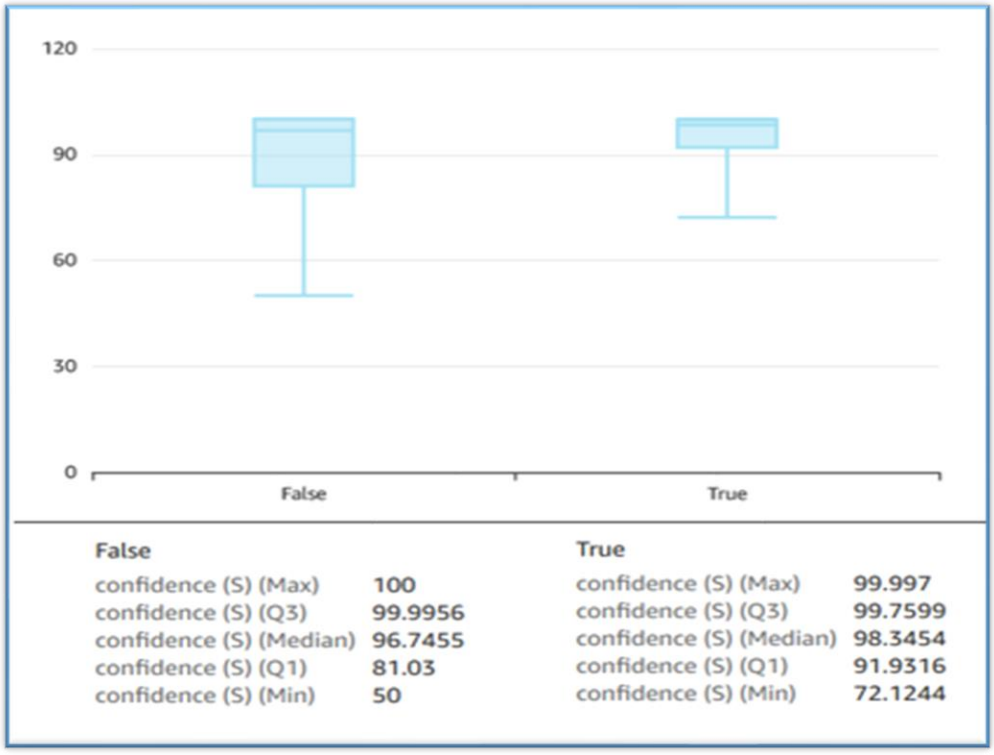

Figure 14: Degrees of Confidence for First Trial

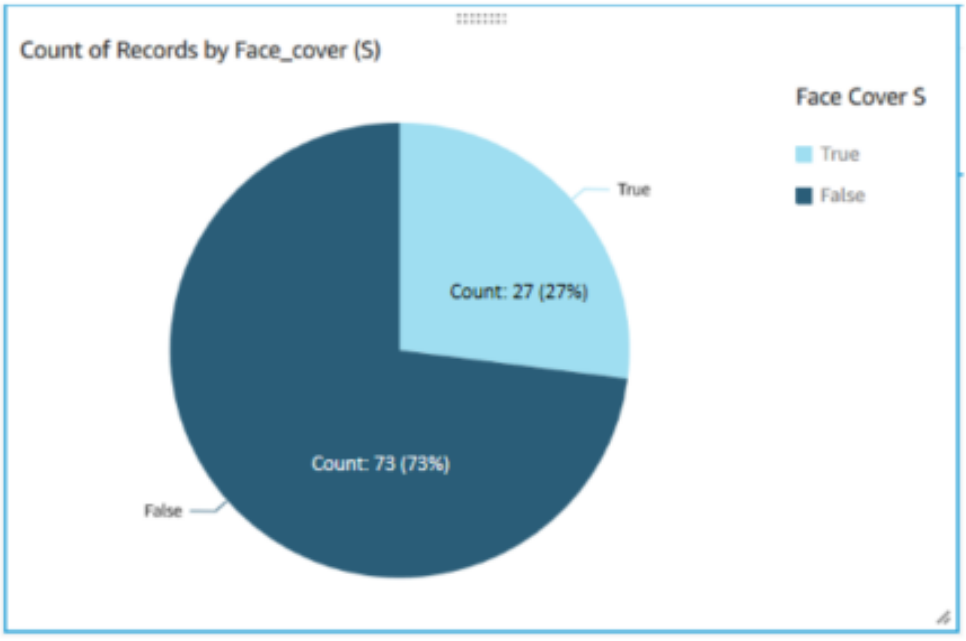

Figure 15: Results of Second Trial

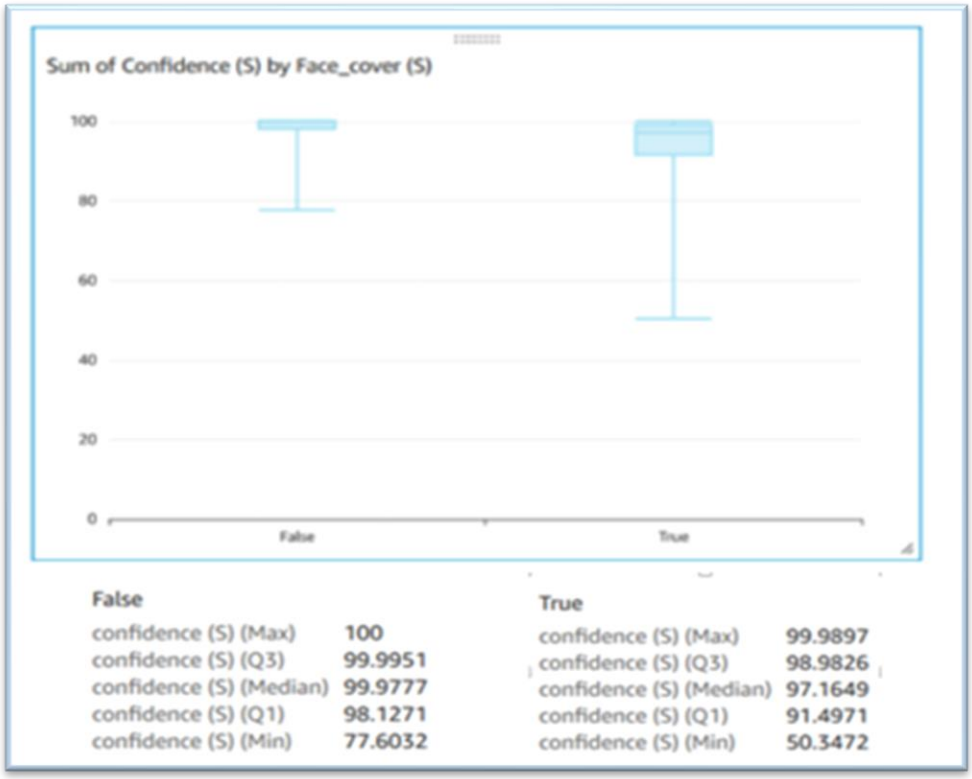

Figure 16: Degrees of Confidence for Second Trial

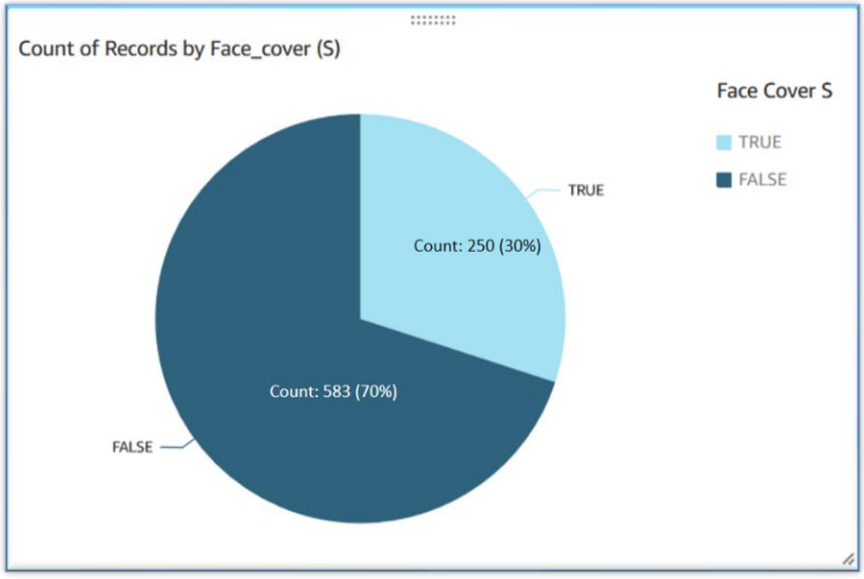

Figure 17: Results of Third Trial

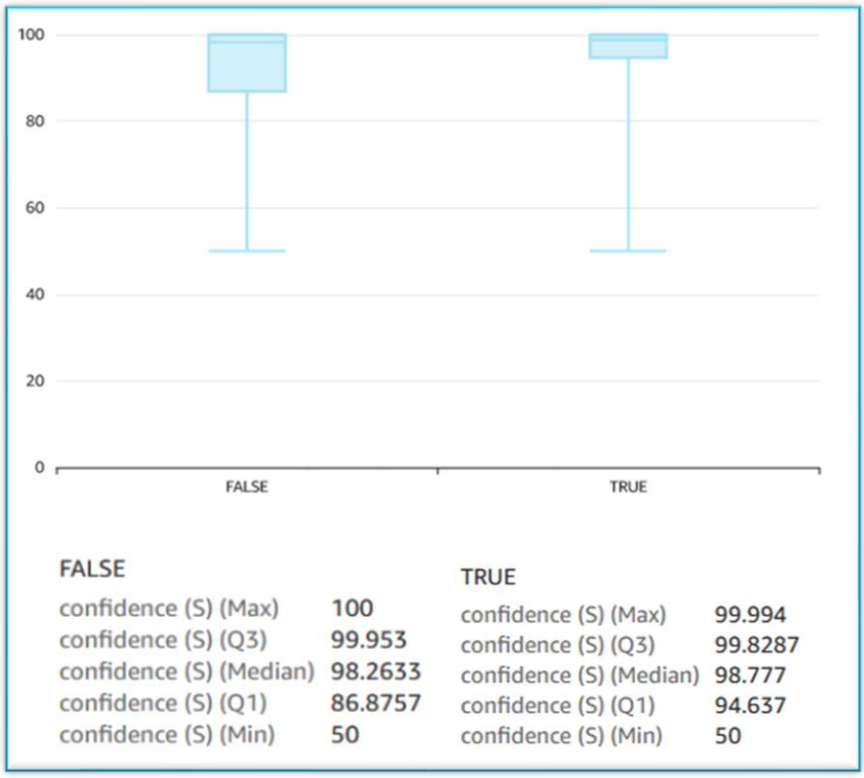

Figure 18: Degrees of Confidence for Third Trial

4.3. Hypothesis Testing

While the total number of detections varied between each experiment, the proportions of the data gathered appear to be consistent across all the trials. We can verify this using a hypothesis testing, more specifically, Z test for two proportions using the following formula:

$$z_0 = \frac{(\hat{p_1} - \hat{p_2}) - (p_1 - p_2)}{\sqrt{\hat{p}(1-\hat{p})}\sqrt{(\frac{1}{n_1} + \frac{1}{n_2})}}$$

Here, we compare the proportions of each experiment with the proportions of the value obtained from simply observing. We first calculate $\hat{p}_1$ by dividing the number of people with masks detected, x_1 with the total number of people detected, n_1. Since we have three different observations as listed in Table 3, we will consider the average of them.

$$\hat{p}_1 = \frac{x_1}{n_1}$$

Similarly, we will calculate $\hat{p}_2$ by dividing the number of people with masks detected in the experiment, x_2 with the total number of people detected in the same experiment, n_2.

$$\hat{p}_2 = \frac{x_2}{n_2}$$

The pooled proportion, $\hat{p}$ is calculated using the following formula:

$$\hat{p} = \frac{x_1+x_2}{n_1+n_2}$$

Our null hypothesis, H_0, is that the two-population proportion, p_1 and p_2, are equal. So, we can assume that $p_1 - p_2$ is 0 as $p_1 = p_2$ if $n_0$were to be true. Hence, we get the derived formula:

$$z_0 = \frac{(\hat{p}_1 - \hat{p}_2) - 0}{\sqrt{\hat{p}(1-\hat{p})}\sqrt{(\frac{1}{n_1}+\frac{1}{n_2})}}$$

We calculate the z_0 value using the formula above. The *P*-value for corresponding z_0 value is retrieved from a standard normal distribution table using the level of significance, α = 0.05. Similarly, z_0 and *P*-values are obtained Experiment 2 and Experiment 3, where we use the same $\hat{p}_1$ and n_1 values but different $\hat{p}_2$ and n_2 values. Table 3 shows the calculated z_o score and the corresponding P-values for each of the experiments.

Table 3: Z Score and *P*-value Relative to the Observed Numbers for Each Trial

	$\hat{p}_1$	$\hat{p}_2$	n_1	n_2	z_0	*P*-value
Trial 1	236	250	680	833	1.945	.05118
Trial 2	236	27	680	100	1.522	.12852
Trial 3	236	19	680	78	1.832	.06724

It is evident from the above table that the *P*-value for each experiment is greater than the level of significance, α = 0.05. Hence, it can be concluded that we do not have enough evidence to reject the null hypothesis, $H_0: p_1 = p_2$. This also verifies our claim that the data gathered in each trial is consistent with the initial observation.

5. CONCLUSION

This paper attempted at testing the feasibility of detecting the size of a crowd, number of people wearing a face cover, and the number of people not wearing one with the aid of the Amazon DeepLens Camera and various Amazon Web Services. While the application detected people with and without face covers with high confidence levels, the number of detections is much lower than the actual count. However, it can be argued that the camera will usually be recording from a stationary position in real life instead of having to make inferences from a moving video. This could increase the number of detections significantly. Also, it was proven that despite the fact that the number of detections was smaller than the actual count, the proportion of true values and false values remained same throughout each trial.

Future work could include testing the camera on a real crowd by placing this in a public place with the permission from appropriate authorities. Also, a frame-by-frame image capture approach can be tried instead of having to detect faces before capturing and uploading an image. While we already tested the feasibility of this in this paper, and found some drawbacks including over counting and uploading too many images, these issues can be solved by using a face-recognition model to discard any duplicate image of a person.

REFERENCES

[1] S. E. Eikenberry *et al.*, "To mask or not to mask: Modeling the potential for face mask use by the general public to curtail the COVID-19 pandemic," *Infectious Disease Modelling,* vol. 5, pp. 293-308, 2020.

[2] S. Taylor and G. J. G. Asmundson, "Negative attitudes about facemasks during the COVID-19 pandemic: The dual importance of perceived ineffectiveness and psychological reactance," Plos one, vol. 16, no. 2, p. e0246317, 2021.

[3] I. El Naqa and M. J. Murphy, "What is machine learning?," in *machine learning in radiation oncology*: Springer, 2015, pp. 3-11.

[4] S. C. Wang, "Artificial neural network," in *Interdisciplinary computing in java programming*: Springer, 2003, pp. 81-100..

[5] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science,* vol. 349, no. 6245, pp. 255-260, 2015.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature,* vol. 521, no. 7553, pp. 436-444, 2015.

[7] F. Ruan, X. Zhang, D. Zhu, Z. Xu, S. Wan, and L. Qi, "Deep learning for real-time image steganalysis: a survey," *Journal of Real-Time Image Processing,* vol. 17, no. 1, pp. 149-160, 2020.

[8] L. Wen, X. Li, and L. Gao, "A transfer convolutional neural network for fault diagnosis based on ResNet-50," *Neural Computing & Applications,* vol. 32, no. 10, 2020.

[9] J. Fernandes, "Introduction to aws deeplens," *Amazon*. [Online]. Available: https://docs.aws.amazon.com/deeplens/latest/dg/what-is-deeplens.html. [Accessed: 10-Nov-2020].

[10] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, "A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic," *Measurement,* vol. 167, p. 108288, 2021.

[11] G. J. Chowdary, N. S. Punn, S. K. Sonbhadra, and S. Agarwal, "Face mask detection using transfer learning of inceptionv3," 2020: Springer, pp. 81-90.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2016, pp. 770-778.

[13] "S3," *Amazon*, 2002. [Online]. Available: https://aws.amazon.com/s3/. [Accessed: 08-Nov-2020].

[14] "Amazon Rekognition," *Amazon*, 2002. [Online]. Available: https://aws.amazon.com/rekognition/. [Accessed: 08- Nov-2020].

[15] “Amazon DynamoDB,” *Amazon*, 2002. [Online]. Available: https://aws.amazon.com/dynamodb /. [Accessed: 08- Nov-2020].

[16] A. Jain, R. Bansal, A. Kumar, and K. D. Singh, “A comparative study of visual and auditory reaction times on the basis of gender and physical activity levels of medical first year students,” *International journal of applied & basic medical research*, 2015. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4456887/. [Accessed: 02-Nov-2020].

[17] “건대입구 - walking around KONKUK Univ. street at NIGHT, Seoul, Korea,” *YouTube*, 13-Aug-2020. [Online]. Available: https://www.youtube.com/watch?v=2R0DZZd3ePg. [Accessed: 14-Dec-2020].

[18] S. Engdahl, “Blogs,” *Amazon*, 2008. [Online]. Available: https://aws.amazon.com/blogs/machine-learning/track-the-number-of-coffees-consumed-using-aws-deeplens/. [Accessed: 20-Nov-2020].

[19] A. Khan, U. Nawaz, A. Ulhaq, and R. W. Robinson, "Real-time plant health assessment via implementing cloud-based scalable transfer learning on AWS DeepLens," *Plos one*, vol. 15, no. 12, p. e0243243, 2020.

[20] M. Jiang, X. Fan, and H. Yan, "Retinamask: A face mask detector," *arXiv preprint arXiv:2005.03950*, 2020.

[21] R. Sachdeva, "Face Mask Detection System," *Available at SSRN 3755508*, 2020.

[22] M. Galkin, K. Rehman, B. Schornstein, W. Sunada-Wong, and H. Wang, "A Hygiene Monitoring System," *Rutgers University’s School of Engineering,* 2019.

[23] G. Rafael and H. Kusuma, "The Utilization of Cloud Computing for Facial Expression Recognition using Amazon Web Services," 2020: IEEE, pp. 366-370.

[24] S. Mane and G. Shah, "Facial recognition, expression recognition, and gender identification," in *Data Management, Analytics and Innovation*: Springer, 2019, pp. 275-290.

[25] B. Murphy and R. Barr, “4. a hypothesis test regarding two Population Proportions,” *mat117.wisconsin.edu*. [Online]. Available: https://mat117.wisconsin.edu/4-a-hypothesis-test-regarding-two-population-proportions. [Accessed: 08-Mar-2021].

[26] “AWS Lambda,” *Amazon*, 2002. [Online]. Available: https://aws.amazon.com/lambda/. [Accessed: 20- Nov-2020].