Final Project Report – Team 7

Face Masks Detection System

Shuo Zhang, Zhiqi Bei, Zhuxian Ding, Wenhan Liu
SYDE 660A
Systems Design Department
University of Waterloo, Canada
{s829zhang, zbei, z73ding, w288liu}@uwaterloo

I. Introduction

The covid pandemic was a global problem since 2019, it caused the deaths of millions of people and seriously damaged the world economy. As many countries have implemented the vaccines against the virus, the pandemic is coming to an end in most area. However, many countries are still doing their best to suppress the spread of the virus, they have strict policies against the virus, everyone need to wear masks in public area, and the improper wearing of face masks is a violation of the government's policy, also, the citizens' freedom were limited, there exists many restrictions. China, for example, is one of them. And the implementation of our product mainly focuses on such countries. In most public areas, the wearing of face masks could be enforced by staff/security staying at the entrances or certain areas. But compared with the implementation of mask detection system, it's obvious that the system would cost less than the employment of staff and can be more reliable as it doesn't need to rest. For our project, we will develop a robust and reliable face mask detection system so that it can detect if a person is wearing face mask properly and if anyone is not following the policy regarding the wearing of face masks, the data will then be sent to the corresponding authority for further actions. In this way, no staff will be needed to check if everyone's wearing their masks properly, and staffs will only be needed when people not wearing masks properly are detected by the system. In such cases, the system will send a notification to security close by for them to identify the person not wearing mask properly and take corresponding actions. And just wearing a mask is not enough, the system will check if people are wearing masks so that both their nose and mouth are covered by the mask. The goal of this project is to maximize the accuracy of the system so that it's making the right judgement on whether people are wearing masks properly and with the success of our project, it will not only help with the pandemic, but it can also help contain the spread of other infectious airborne virus before the effective vaccines are developed and put on the market.

In this project, A face masks detection system will be developed to help find the people not wearing masks properly in high traffic activities. Staff do not need to make an inspection tour, instead, they just need to check the face masks detecting system results. When there are enough people against the mask policy detected in a certain area, they can notify people in that place to follow the covid measures by broadcast if radios are deployed or send the security groups nearby to remind these people to wear the mask properly. The system will focus on the single person face image detection, which means extracting each person from a multi-people picture is not in current scope and we assume it has been completed by other algorithms. In this project, we will not only detect people that wear face masks or not, but also find out whether masks are worn in a correct way among people that have masks equipped (cover mouth and nose).

The goal of this project is to maximize the accuracy of the system as well as minimize the number of failures on identifying people that do not wear masks properly (type I error) and

misclassifying people that wear masks in the correct way as cases against the mask policy. Our project will aim at not only the covid pandemic, but also help contain other infectious diseases transmitted by airborne or droplet-nuclei before the effective vaccines developed and put on the market.

II. Project Scope & Objectives

Nowadays, the virus is still very active in many countries, and it's still taking away lives from many innocent people. In order to suppress the spread of the virus, many countries have different policies in place to protect people from getting the virus. Amongst all the methods that officials suggest, the proper wearing of face masks is the most effective and easiest way to suppress the spread of the virus. However, as the pandemic is coming to an end globally, many places have loosened the restrictions against the proper wearing of face masks. Also, as most people have already been vaccinated, the wearing of face masks is not being taken seriously by a large group of people even when they are at places where they should be wearing masks properly. And in this case, it's necessary to find and detect which people is not wearing masks properly and causing dangerous environments to those around them. For this problem, our project can be very useful, it could be used to detect which person's not wearing their face masks properly, and will then show the person on the monitor screen and send the data to corresponding authorities for further actions. For our project, we intend to develop a system so that it could not only detect if a person is wearing the face mask, but also if that person's wearing it so that both his/her nose and mouth are covered by the face mask. Our model will have a photo of a person as the input and will detect if the person's wearing the mask, and if the nose and mouth of the person are covered. This model can be used at any place where people are expected to wear face masks, and the hardware needed for it to work would be the security cameras. However, as there will always be a high traffic of population in public areas, there must be more than one human face existing within each frame of video the camera takes. For our project, due to the time constraints, we will only implement the model for detection, and we have assumed that there will be another system that extracts human faces from what the camera has taken, and then the extracted human faces will be sent to our model one by one for further detection. And the person not wearing mask the right way will be detected and highlighted so that the staff watching the monitor would know and further actions could be taken. The final goal of our project were to develop the model so that it can make the final decision fast and accurate. And for the final product that we developed, we wanted it to have high accuracy, high efficiency, and also easy to use. For the development process, each team member will develop a unique model of their own, there will be 4 models for members to choose from, they are Resnet + softmax/svm and self-developed CNN + softmax/svm. Within 2 weeks after the preliminary presentation, each member should come up with a model that reached accuracy of at least 60%. And each member will try their best to make the model perform even better, and have at least 85% of accuracy as

well as a short processing time. and after weeks of researching and tuning the developed model, the model with the highest accuracy as well as fastest processing time will be chosen as our final product. And all team members will then be working together on solving existing problems and tuning the chosen model in order to make it perform even better. The whole project objectives were developed based on the time frame of the whole duration of this course, and these are guaranteed to be achievable. These objectives made sure that we could complete the face mask detection model in time, and can also allow us to choose the best model as our final product. And finally, for the model that we developed, we will measure the performance and judge how well it does by its accuracy and processing time, and a confusion matrix will be used as well. The accuracy of the model is very important as it needs to detect which people are not wearing masks properly and show it to the staff that is responsible for this matter. If the accuracy of the model does not meet our goal, then there would be many false detections and this could actually increase the workload of the staff and cause opposite effect that we were expecting. Hence, for this model we developed, we expected it to have at least 85% of accuracy, and the higher the better, and we only considered the accuracy objective being met if it has 85% or higher accuracy in the end. Otherwise, the authorities might as well hire staffs to check if each person is wearing mask properly by themselves. Due to the high traffic of population in public areas, the model have to be able to receive the image and make decision in a timely fashion, so that the person not wearing mask properly will be detected while they are still in the range of the security camera, and can be shown to the security staff watching the monitor. We need the processing time to be short as it allows quick response of the staffs and the less time the person not wearing mask properly got exposed to a dangerous environment, the better. If the processing time is taking too long, then its' more likely for the person not wearing mask right to get infected by the virus or for the person to spread more virus, if he/she is already infected. And with a longer processing time, the purpose of using this model would be defeated as staffs could check which person's not wearing mask properly even faster. Hence, for our model, the time range would be from the time it received the image of human face to the time it gives out decision on whether the person is wearing the mask properly. And for the whole development process, we expected the time range to be within 3 seconds, and the shorter the better.

III. Designed Solution

A. Design Concept and Decision making

According to our previously listed requirements, our group discussed the structure of the mask detection system. Since we want to make our system easy to use, including people with no technical background, and all functions in the system are easy to modify and extend. So, we decide to divide the system into different parts according to different functions.

There are four main parts in our system as show in Fig. 1: surveillance cameras, face detection module, mask detection module, and notify module. First, we get real-time photos from surveillance cameras, these pictures are then sent to a face recognition module to extract the faces on the pictures, and each face will then be made into many separate images, these images will be sent to the mask detection module to detect if someone is not wearing a mask correctly. If someone is not wearing a mask properly, the notify module will send a message to the user (maybe a security guard) that not everyone in this area is wearing a mask correctly.

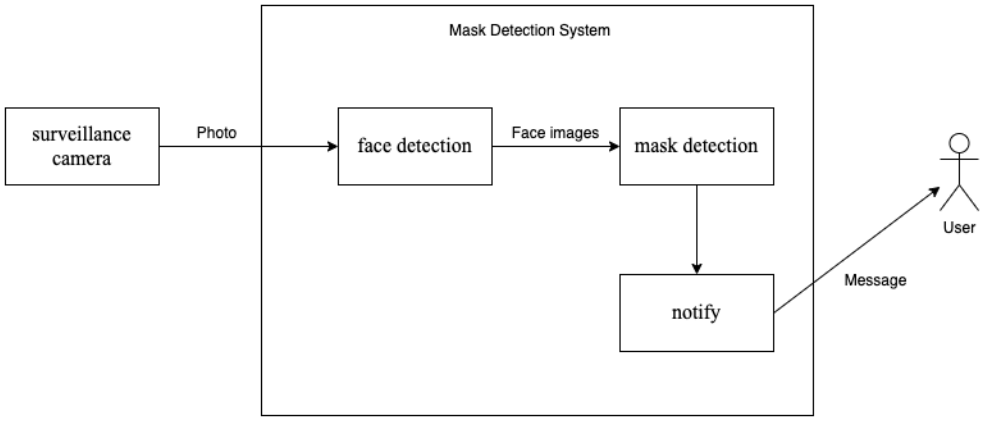


Fig. 1: The structure of whole system

However, in our plan, the whole project must not take longer than 5 weeks to complete by a team of four members. Therefore, due to time constraints, we will only complete parts of the system first. After discussion, we decided to prioritize the mask detecting module. Because for a mask system, the mask detection module is the most important part.

For the face detection module, we decided to use machine learning algorithms to achieve. We first need to process the images that receive from the face detection module, extract the features, and then categorize these pictures base on the features. The system will classify the images into four categories:

1. The mask is worn correctly, covers the nose and mouth.
2. The mask covers the mouth but does not cover the nose.
3. The mask is on but does not cover the nose or mouth.
4. There is no mask on the face.

There are many different methods have been developed by researchers that can be used to extract features and classify images, like Convolutional Neural Network (CNN), Deep residual network (ResNet), AlexNet, VGG, etc. These are some widely used neural network in the field of image recognition.

Neural network and CNN: Neural network is a mathematical model or computational model that imitates the structure and function of biological neural network (the central nervous system

of animals, especially the brain). Neural networks are composed of many artificial neurons, which are constructed in different ways, CNN is one of them, CNN are best at processing images and is widely used in the field of image recognition. CNN has 2 major features: 1) It can effectively reduce the dimensionality of pictures with large amounts of data into small amounts of data. 2) It can effectively retain image features and conform to the principles of image processing.

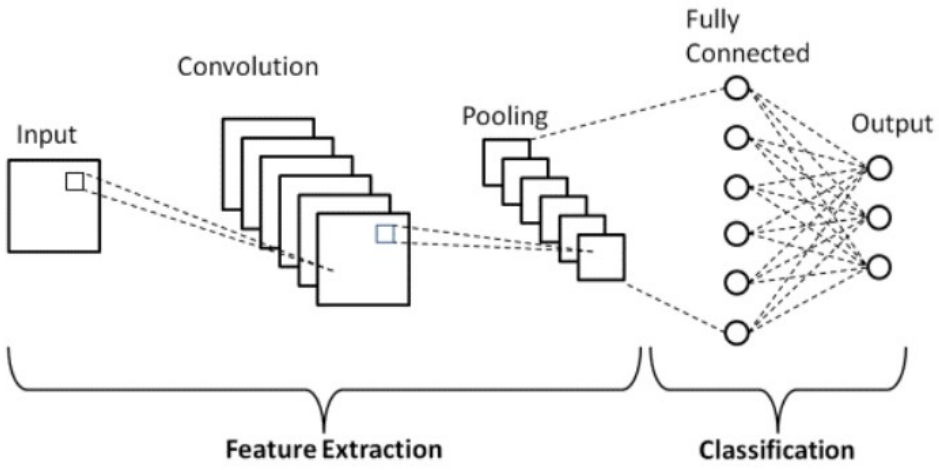


Fig. 2: Basic CNN structure

ResNet: ResNet has similar structure as CNN, the main difference between CNN and ResNet is that ResNet introduce an ”identity shortcut connection” [25], so that the accuracy will not decrease when the model increases their number of layers.

AlexNet: AlexNet was proposed in 2012 by Alex Krizhevsky et al, it apply the basic principles of CNN to deeper and wider networks and won the 2012 ImageNet LSVRC champion.

VGG: The VGG model was proposed by Oxford’s Visual Geometry Group and was the second place in the 2014 ILSVRC competition. An improvement of VGG compared to AlexNet is to use several consecutive 3x3 convolution kernels instead of larger convolution kernels in AlexNet (11x11, 7x7, 5x5)

All those neural networks are having similar architecture, which consist of 3 layers:

(1) Convolution Layer, used to extract features.

(2) Max Pooling Layer, used for down sampling without corrupting the recognition result.

(3) Fully Connected Layer, used to classification.

Since we also need to classify how the mask is worn (correctly worn, only covering the mouth, or hanging on the chin), so we plan to modify the classification layer and try different classification method to see if it’s possible to improve the accuracy.

There are also many existing methods that can be used to classification, like support vector machine (SVM), k-nearest neighbour (KNN), decision tree, etc.

SVM: SVM is a supervised learning model and related learning algorithm for analyzing data in classification and regression analysis. It a very common classification model which can solve high-dimensional (large feature spaces) problems and has no local minimum problem (compared to algorithms such as neural networks).

KNN: KNN is to search the entire training set of the K most similar instances (neighbors) and summarizing the output variables of those K instances and making predictions for new data points. The KNN is easy to implement and the time complexity of training the KNN is only O(n). However, for datasets with large sample size, the amount of calculation is relatively large.

Decision Tree: Decision tree is an algorithm that uses a tree-like data structure to solve classification problems and is a supervised learning algorithm based on if-then-else rules. It is also easy to implement, but when the dataset is large, it is prone to overfitting.

Our target for the algorithm is to maximize the accuracy and minimize the executing time. We want to try many different models to find the target model, however, there were only four of us in our group and we only had five weeks. It's difficult for us to try every method mentioned above, so we made a trade-off. We finally chose to use CNN and ResNet to extract the features. Since ResNet, AlexNet and VGG are enhanced models of CNN, therefore we decide to use CNN as our basic model. all the ResNet, AlexNet and VGG are good methods, and they are similar. So, it's hard to tell which of them is better. We made our decision based on when they were proposed. ResNet was proposed in 2015, VGG was proposed in 2014, and AlexNet was proposed in 2012.

For the classification layer, we keep the original method in the CNN and ResNet, which is softmax function, and we choose SVM as a comparison. Since we just want to replace the last layer of each network, KNN and Decision tree are not suitable as a layer of the network, they are more suitable for independent execution.

Finally, we combine each extract features method and classify method, that is, let softmax and SVM be the last layer of CNN and ResNet respectively, therefore, we will get four different model:

(1) CNN with softmax as its last layer (classification layer),

(2) CNN with SVM as its last layer,

(3) ResNet with softmax as its last layer, and

(4) ResNet with SVM as its last layer.

And each group member will work on one of the model above.

B. Final Design Idea

We decide to implement our models in the Google colab platform using python, since Google Colab can provide free GPU which can increase the execution speed.

CNN model

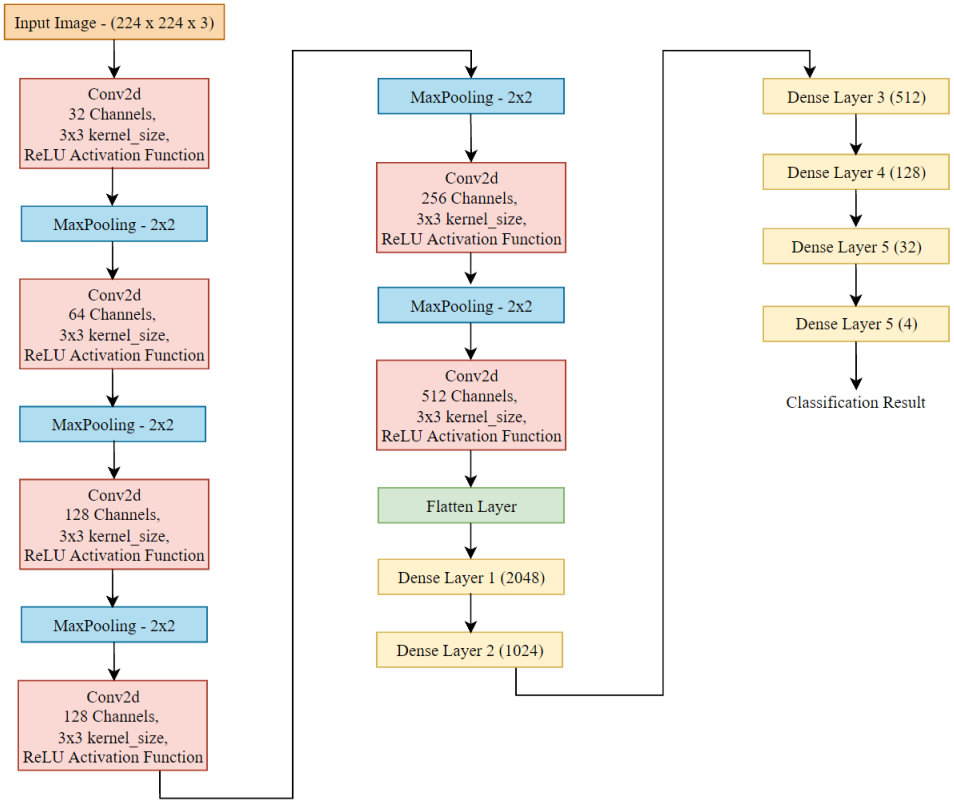


Fig. 3: The CNN model structure

we first tried and implemented VGG model. With the VGG model implemented, the test accuracy and processing time did rather well, and our initial expectations were already met by this model. However, we wanted to try something else, and hope to gain an even better results than what we had. Hence, we started to implement out self-designed CNN model, and it ended up having similar performance as the VGG model. For the CNN model that we implemented, processing time was much better than what we expected. For the 2 self-designed CNN models that we implemented, the only difference between them was at the final layer, one model had softmax function at the final layer and the other one had svm at the final layer. For the CNN model, the entire structure is shown below:

1) Input: 224 x 224 x 3
2) Conv: 32 channels, 3×3 kernel size, stride of 1
3) ReLU activation function
4) MaxPooling: 2×2 kernel size, stride of 2
5) Conv: 64 channels, 3×3 kernel size, stride of 1
6) ReLU activation function
7) MaxPooling: 2×2 kernel size, stride of 2

8) Conv: 128 channels, 3×3 kernel size, stride of 1
9) ReLU activation function
10) MaxPooling: 2×2 kernel size, stride of 2
11) Conv: 128 channels, 3×3 kernel size, stride of 1
12) ReLU activation function
13) MaxPooling: 2×2 kernel size, stride of 2
14) Conv: 256 channels, 3×3 kernel size, stride of 1
15) ReLU activation function
16) MaxPooling: 2×2 kernel size, stride of 2
17) Conv: 512 channels, 3×3 kernel size, stride of 1
18) ReLU activation function
19) MaxPooling: 2×2 kernel size, stride of 2
20) Flatten Layer
21) Dense Layer: 2048 neurons
22) Dense Layer: 1024 neurons
23) Dense Layer: 512 neurons
24) Dense Layer: 128 neurons
25) Dense Layer: 32 neurons
26) Dense Layer: 4 output classes

In order for the whole thing to work, several libraries were used. From tensorflow we have imported, Sequential, Dense, MaxPooling2D and Flatten as they were needed to construct our model. We also imported pandas and numpy for the reading and processing of the dataset. With the model that we implemented, we used colab for the whole implementation and have trained it for 20 epochs.

ResNet model

Similar as the VGG model, the only difference between our two ResNet model is the last layer, one model had softmax function at the final layer and the other one had SVM at the final layer. For the two ResNet model, the entire structure is shown below:

1) Input: 224 x 224 x 3
2) ResNet model
3) GlobalAveragePooling2D
4) Dense Layer: 2048 neurons
5) Dense Layer: 512 neurons
6) Dense Layer: 4 output classes

All the above layers can be imported from the tensorflow package provided by the python.

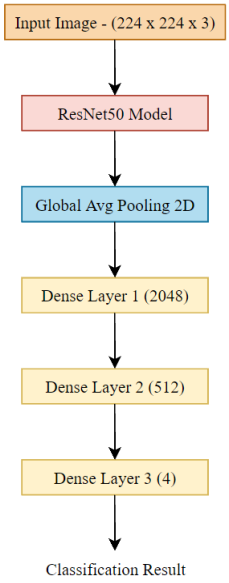


Fig. 4: The ResNet model structure

IV. Design Analysis & Testing

In this section, our team focuses predominantly on engineering analysis and modeling. Since our team members all specialize in AI & ML, the project aims to develop a machine learning model with high performance for the face mask detection system, and a user interface is not included. As a result, we emphasized engineering analysis and modeling in this section for the four algorithms we developed, with only a small part about testing.

A. System input and output

According to the project scope, the system input is a RGB image (in JPG or PNG format) of single person wearing or not wearing a mask which filtered out from an original picture taken in a fixed location by other algorithms. In order to increase efficiency and stability of our deep learning network for mask detection, all input images are resized into the size of 224×224×3 during data preprocessing.

After feeding the image data into the deep learning network, a result with the probability of each class the image classified into is generated. As mentioned above, there are four classes in total and the class with the highest probability is the final classification result of the image. Then the system outputs the specific mask wearing type for all images generated from the original picture and calculates the number and the percentage of people who are not wearing a mask properly. If there are people not wearing a mask in the correct way, such as type 2, type 3 and type 4, the system will display a red warning on the screen with the number of these people to inform the employees.

B. Basic CNN structure and our model architectures

Since face mask detection is an image classification problem, convolutional neural network (CNN) was chosen as the basic framework for the system due to its excellent performance on multi-class image classification tasks. As can be seen from the Fig. 2, CNN typically consists of three layers: convolution layer, pooling layer and fully connected layer.

Convolutional layer The convolutional layer is the core building block of CNN, which is used for feature extraction. In this layer, a mathematical operation of convolution is performed between the input image and a filter of a particular size [1]. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter. Then, a feature map [2] of the processed images from the convolution layer is generated. The mathematical formulation of two-dimensional convolution for input image X and filter H is defined as follows [3].

$$X'[i,j] = \sum_m \sum_n H[m,n] \cdot X[i-m, j-n] \tag{1}$$

Where X' represents the output image after convolution operation. The indices i and j are concerned with the image matrices X and m and n deal with the filter matrices H.

Because convolution is a linear operation and images are far from linear, non-linearity layers are often placed directly after the convolutional layer to introduce non-linearity to the activation map. The activation function we chose was the ReLU [4] which is the most popular activation function in convolution layer. It computes the function of

$$f(x) = \max(0, x) \tag{2}$$

In other words, when the input value is greater than or equal to zero, the output is the input itself, while when the input value is less than zero, the output is zero. Compared with other activate functions like sigmoid and tanh, ReLU is more reliable and accelerates the convergence.

Pooling layer In common situations, a convolutional layer is followed by a pooling layer, which refers to approximating the outputs by aggregating nearby values and reducing the size of the feature map from the latest convolution layer [2]. Two common pooling functions are used in the system are max pooling and average pooling. The size of the pooling filter is usually a 2 by 2 matrix. In max pooling, the 2 by 2 filter slides over the feature map and picks the largest value, while in average pooling, the average of the values in the filter is calculated.

Fully connected layer After flattening, the pooled feature map is transformed into a single column and is passed to the fully connected layers. Fully connected layers are used to connect the neurons in the preceding and succeeding layers and to generate the classification result for

the input image. Thus, they are placed before the output layer, forming the last few layers of the CNN structure.

The last fully connected layer is responsible for class prediction, so an activation function needs to be used. In the project, images are supposed to be classified into four categories, so softmax was chosen as the activation function for the last layer of basic CNN model to normalize the prediction result. The form of softmax is usually given as follows [5]:

$$p_i = \frac{e^x i}{\sum_j e^x j} \tag{3}$$

where x_i is the total input received by unit i and p_i is the prediction probability of the image belonging to class i.

In addition, since SVM is also a good technique for multi-class classification, we convert original softmax activation into SVM in the last fully connected layer. To do that, we implement kernel regularizer as the regularizer function with $L2$ norm and use linear activation function instead.

Our model architectures The architecture of four models we implemented are shown in Fig. 5 and Fig. 6. The green rectangles represent the input image data, the red rectangles represent the convolutional layer, the blue rectangles represent the pooling layer, the yellow rectangles represent the fully connected layer without the last layer and the purple rectangles represent the last fully connected layer with softmax or SVM as the classifier.

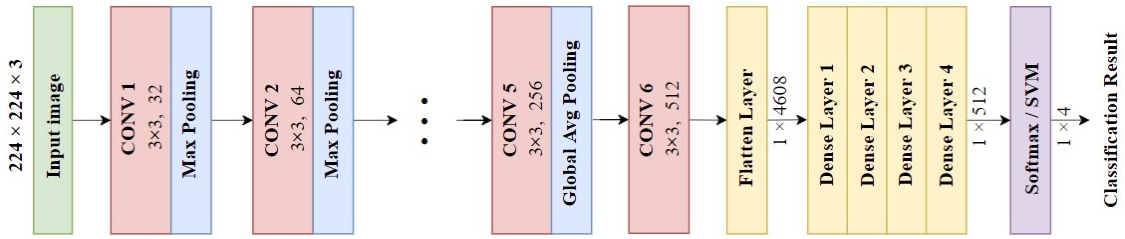


Fig. 5: Architecture of the basic CNN models we implemented

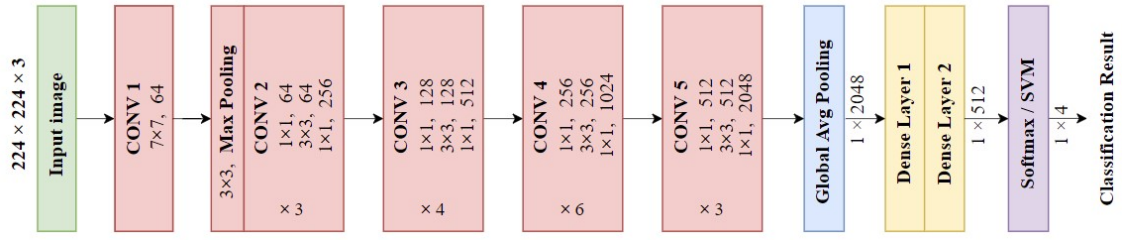


Fig. 6: Architecture of the ResNet50 models we implemented

C. Optimization and loss function

Adam optimizer The weight optimizing function chosen for all structure is Adam optimizer algorithm, which define in the figure 7 where m_t denotes the aggregate of gradients(momentum) at time t, Θ_t denotes weights at time t, α denotes Step size parameter / learning rate, $\beta_{1\&2}$ means decay rates of average of gradients in the above two methods. v_t means sum of square of past gradients. weight of each layer will keep evolve along gradients calculated and reach a stop after converge or end of training.

Algorithm 1: Adam Optimizer

Initialize $\theta_0, m_0 \leftarrow 0$, $v_0 \leftarrow 0, t \leftarrow 0$
While θ_t not converged
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
Bias Correction
$\widehat{m_t} \leftarrow \frac{m_t}{1-\beta_1^t}, \widehat{v_t} \leftarrow \frac{v_t}{1-\beta_2^t}$
Update
$\theta_t \leftarrow \prod_{\mathcal{F}, \sqrt{\widehat{v_t}}} \left(\theta_{t-1} - \frac{\alpha \widehat{m_t}}{\sqrt{\widehat{v_t}} + \epsilon} \right)$

Fig. 7: Adam optimizer algorithm

Cross entropy loss The loss function chosen for softmax for multi-classification is the cross entropy loss, which term the percentage output array into one-hot array of same size with 1 on entry with maximum percentage value and 0 on others.

Squared hinge loss The loss function chosen for SVM for multi-classification is the squared hinge loss [6], which defined as:

$$L(y, \hat{y}) = \sum_{i=0}^{N} \left(max\left(0, 1 - y_i \cdot \hat{y}_i\right)^2 \right) \tag{4}$$

where y denotes the true label of the data which is -1 or 1, and $\hat{y}$ denotes the predicted value. Thus, the value of the squared hinge loss is equal to 0 when the true and the predicted labels are the same or when $\hat{y} \geq 1$, and the value is quadratically increasing with the error when the true and the predicted labels are different or when $\hat{y} < 1$ [7].

D. Design parameters

1) Number of convolutional layers in basic CNN model: The number of convolutional layers in a CNN model is an important hyper-parameter [8]. In CNN models, the number of convolutional layers is increased to increase the depth of the feature space thus helping in learning more levels

of global abstract structures to get a better result. However, a large number of convolutional layers under specific image and filter size may lead to overfitting.

To evaluate how the number of convolutional layers of a CNN affects its performance, an analysis was done on our basic CNN model with 4, 5, and 6 convolutional layers. The number of filters in the model increased from 16, 32, 64, 128, 256 to 512 as layers increased. Other hyperparameters remain the same in all convolutional layers.

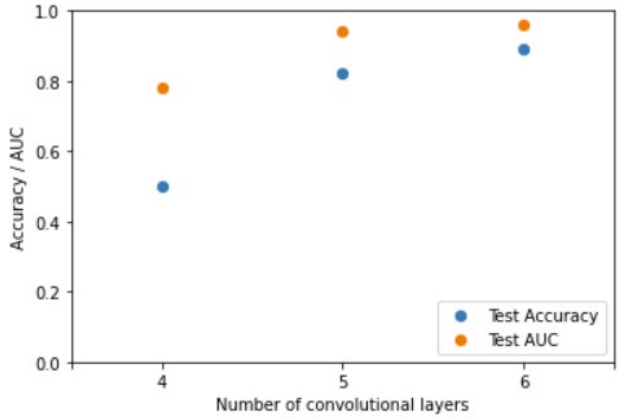
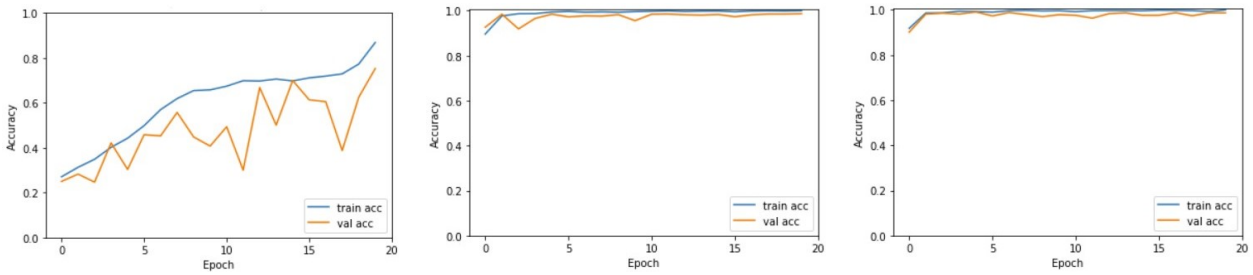


Fig. 8: Accuracy and AUC values of CNN model with different number of convolutional layers

The Fig. 8 shows that with the number of convolutional layers increased, more features are extracted, which leads to a better test accuracy and AUC value. Therefore, in our two basic CNN models, we implemented 6 convolutional layers to make the network more efficient.

2) Learning rate in ResNet50 model: The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated. Choosing the learning rate is challenging as a value too small may result in a long training process that could get stuck, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process [9].

In the project, we tested the learning rates of 0.01, 0.001 and 0.0001 in ResNet50 model. The results are shown in Fig. 9. We can see that when the learning rate is small, the network is unstable with great fluctuations generated. But as the learning rate increased, the network performed better but converged slowly and needed a longer training time. Since the learning rate of 0.001 and 0.0001 did not make any significant difference in our ResNet50 model, to decrease training time, we chose the default value 0.001 in the project.

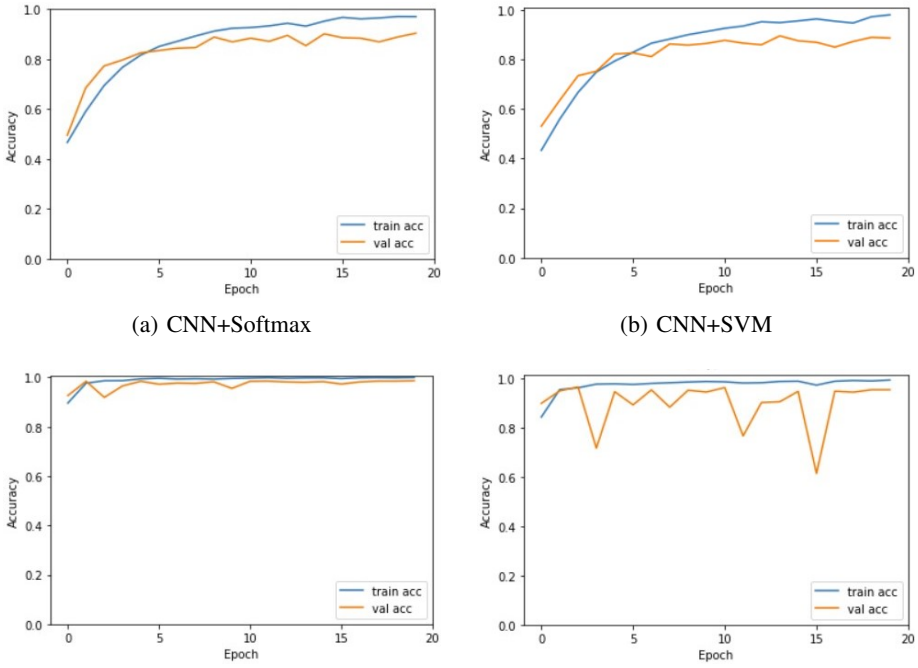


(a) lr = 0.01 (b) lr = 0.001 (c) lr = 0.0001

Fig. 9: Accuracy of ResNet50 model with different learning rates

E. Comparison of models

1) Training, validation and test accuracy: The training and validation accuracy of the four models (two basic CNN models with softmax and SVM as the classifier respectively, ResNet50 models with softmax and SVM as the classifier respectively) for 20 epochs are shown in Fig. 10. It shows that all the four models we implemented did good performance on both training and validation datasets. The accuracies on these two datasets gradually increase with the number of training epochs, and they are higher than 85% after 20 epochs, indicating that no overfitting or underfitting happened during the processes.



(a) CNN+Softmax (b) CNN+SVM

(c) ResNet+Softmax (d) ResNet+SVM

Fig. 10: Training and validation accuracy of the four models for 20 epochs

Comparing basic CNN models with ResNet50 models, we can see that ResNet50 had a better

performance on both the accuracy and convergence rate. ResNet50 models achieved the accuracy of 90% within 3 epochs, while basic CNN models needed 10 epochs to get the same results. At the same time, after 20 epochs, ResNet50 models reached higher accuracy than basic CNN models did. The reason for this phenomenon may be that ResNet 50 has deeper architecture and is more complex than CNN, which makes it a better framework for image classification. The parameters and training and testing time of these two model frameworks are shown in Table I. It can be seen that ResNet 50 has more parameters than Basic CNN, but more training and testing time.

TABLE I: Parameters and experiment time of CNN and ResNet50

Model	CNN	ResNet50
Parameters	13,780,420	28,835,204
Train time / min	7	32
Test time / sec	2	8

Meanwhile, which is also notable is that models with softmax as classifier always have a better result than the same models using SVM as classifier. The reason may be that for multi-classification tasks, the squared hinge loss function used in SVM classifier is not as stable and precise as cross entropy loss function used in softmax. Thus, softmax is more commonly used in deep learning networks for multi-classification.

TABLE II: The accuracy of each model(%).

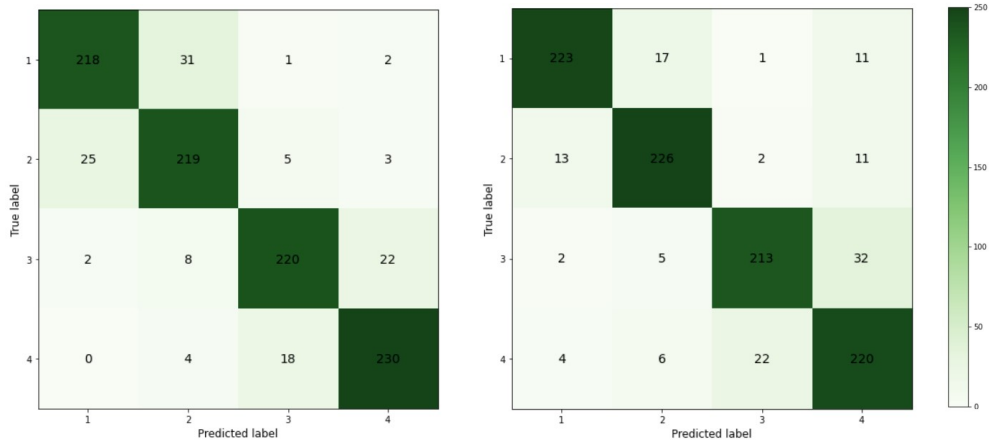
Model	CNN+ Softmax	CNN+ SVM	ResNet50+ Softmax	ResNet50+ SVM
Training	99.39	95.28	99.93	95.87
Validation	96.74	87.73	99.65	94.67
Testing	89.49	87.39	98.31	94.24

The Tab. II shows that the performance of each model on test dataset is consistent with that on training and validation dataset. The ResNet50 model with softmax as the classifier achieved the highest accuracy of 98.31%.

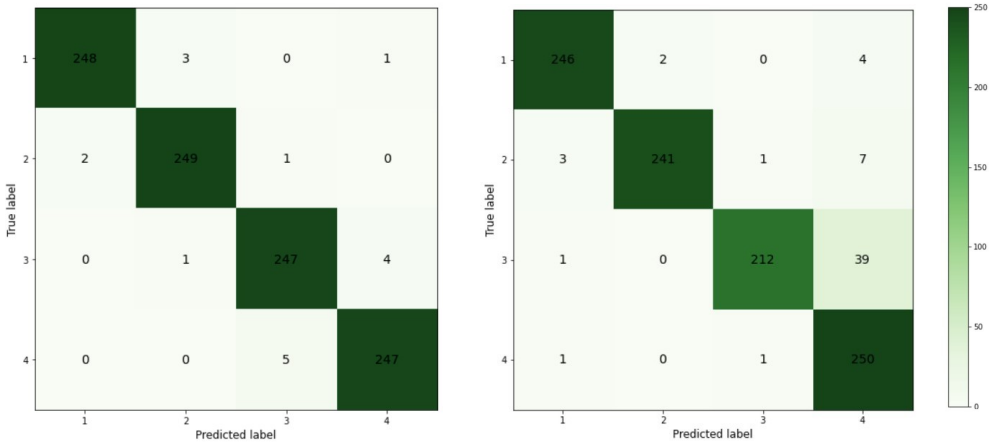
2) Confusion matrix: Confusion matrix [10] is a technique used to evaluate the performance of machine learning classification. It is represented in a matrix form and gives a comparison between actual and predicted values. The confusion matrix of our four models on test dataset are represented below. In confusion matrix, some important conceps are defines as follows:
True Positive (TP): It refers to the number of predictions where the classifier correctly predicts the positive class as positive.
True Negative (TN): It refers to the number of predictions where the classifier correctly predicts the negative class as negative.

False Positive (FP): It refers to the number of predictions where the classifier incorrectly predicts the negative class as positive.
False Negative (FN): It refers to the number of predictions where the classifier incorrectly predicts the positive class as negative.
Precision: It tells what fraction of predictions as a positive class were actually positive, which can be calculated by the formula: $TP/(TP + FP)$.
Recall: It tells what fraction of all positive samples were correctly predicted as positive by the classifier, which can be calculated by the formula: $TP/(TP + FN)$.

In Tab. III, the precision and recall values of each model are calculated. We can see that the ResNet50 model with softmax as the classifier also got higher values than other models.



(a) CNN+Softmax (b) CNN+SVM



(c) ResNet+Softmax (d) ResNet+SVM

Fig. 11: Confusion matrix of the four models

3) Processing speed: According to the design specifications, a fast-processing speed is an essential factor in the project. To evaluate the processing rate of each algorithm, we fed 10 images into the network and calculated the average time needed for generating a result. As it's

TABLE III: The precision and recall values of each model.

Model	CNN+ Softmax	CNN+ SVM	ResNet50+ Softmax	ResNet50+ SVM
Precision	0.88	0.87	0.98	0.95
Recall	0.88	0.87	0.98	0.94

shown in the Fig 12, ResNet models took nearly 0.5 second less than basic CNN model did for each image. Thus, ResNet 50 model should be our first choose.

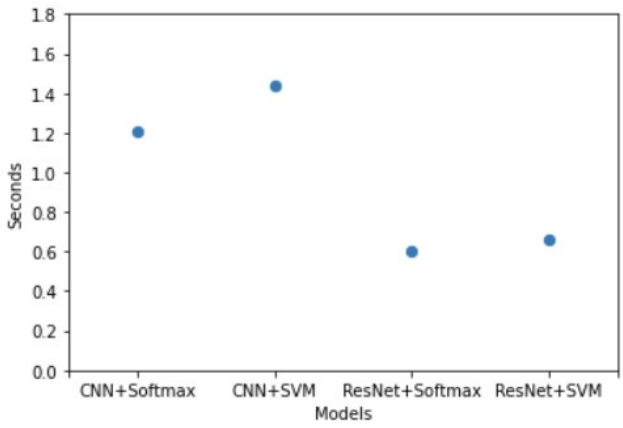


Fig. 12: Processing speed of the four models

F. Results analysis - the best model

According to the results and analysis shown above, the ResNet50 model with softmax as the classifier is the best model with the highest testing accuracy and least processing speed. Therefore, ResNet50 with softmax as the classifier is the final model we used in the project.

V. LIMITATION

One of the limitations of this project comes to the size of dataset. Our original dataset was about 80 Gigabyte, but when we started coding, we found that there was not enough space for some of our machines to download dataset in such a large size. Also, in order to reduce the processing time, we pick a relatively small dataset (around 10 thousand images). Although the system performs well on these images, it is still not sure whether the program is biased or not. In the possible future work, larger and more diverse pictures will be included into the dataset. Besides, according to our requirement scope, we are focusing on differentiating faces with mask correctly equipped from wrongly mask-wearing faces. Most of pictures in the dataset are straight face photos in different orientation. If we get the chance to continue the project, pictures with multiple people or video from surveillance camera with algorithm that extract each face feature will also be implemented. Apart from that, we have not included user interface design in this

project, the system runs directly on jupyter notebook (ipynb) files and alert message appears under output window (colab) and windows notification bubble (local machine). Users may be confused about how to operate the system. Since the above are all not in the scope of current project. These will be treated as limitation of the product.

VI. Challenge

During the decision process, our plan was to use four different methods to approach the goal., During implementing CNN+SVM structure, there is not a suitable package to connect the output of CNN layers with SVM classification layers. we firstly tried to use linear classifier as activation function of last CNN layer with square hinge loss to approach the effect of the SVM. 75% accuracy was the best result during system tuning step, and this result was not reliable due to its high type I error, then we applied the regularized hinge loss, optimization parameter, and gradient approach manually. However, the output did not get better, the accuracy is very low (about 60%) and the accuracy stop growing after 20 epochs. So, we abandon this approach. We tried some other simple model (SIFT, SVM...), but without a neuron network system, the test accuracy cannot pass 80%. And for SIFT algorithm, the execution time of the system is relative longer as we need to extract 8-direction gradient features from each image. In the end, we decided to focus on the CNN model design. Because images are not from a varied dataset. We believe Neuron Network plus classifier last page will work fine on the project goal. We used a non-trainable VGG-19 model followed by global-pooling layer as a specific filter, and a set of dense layers to separate photos with different face mask wearing condition. Second challenge comes from team cooperation. Our plan was each team member generate one detecting method. During designing process, we altered our original design plans (e.g., from CNN+SVM to another CNN+SoftMax solution). There should be a communication between teammates about the changes. However, the real situation is that we treat these changes as part of our own work and forget to inform teammates about the new plan. Which made the result of two CNN methods have similar structures. We found the problem very late before entering documentation step, So, we abandon one CNN model and use the packaged VGG model to speed up training.

VII. Conclusion

In this project, we develop a system that can differentiate the wrongly worn mask face photos from images that use face mask correctly. The system will print alert message to the screen when enough people are detected to have face mask wearing issue. The scope of this project is from the situation that pictures have been extracted from the video or multi-people images taken by a well deployed camera and sent to the system already, to the system detects enough people in the area and pop up the alert message or the potential hazard is not significant, and system prepared to receive next bunch of pictures. We try to approach a high accuracy as well as low type I type II error rate system with 4 different neuron network structure, which are

VGG+softmax, customized CNN+softmax, Resnet+softmax and Resnet+SVM, we compare the result of 4 methods and pick Resnet+softmax structure as our final solutions due to its high performance. The system reaches 98% in accuracy, recall and precision rate. it is completely software product that run and test on colab and local machine which makes it executable on all kind of hardware with python environment implemented and cost free (rule out of local machine cost and the price of colab pro if faster running speed needed). The system takes only 0.63 second to processing an image and the system itself is just an ipynb file counted less than 1 mb storage. Since the UI design is out of scope of the current product, we have not made the product into an encapsulated application. User may be confused about the result page if they have no experience in operating IDE. Apart from that, as a pure code product. The algorithm can be easily added by adding/altering functions. In the future work, we will try to make the system into a complete application with a more technical designed UI. Algorithm that extracting face images from video stream and features from face points to different directions will be implemented.

REFERENCES

[1] Gurucharan M K. Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network). https://www.upgrad.com/blog/basic-cnn-architecture/, 2020. [Online; accessed 16-July-2022].

[2] S. Zayen N. Jmour and A. Abdelkrim. Convolutional neural networks for image classification. In *2018 international conference on advanced systems and electric technologies (IC_ASET)*, pages 397–402. IEEE, 2018.

[3] Tulin Ozturk, Muhammed Talo, Eylul Azra Yildirim, Ulas Baran Baloglu, Ozal Yildirim, and U Rajendra Acharya. Automated detection of covid-19 cases using deep neural networks with x-ray images. *Computers in biology and medicine*, 121:103792, 2020.

[4] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[5] Shabir Hussain, Yang Yu, Muhammad Ayoub, Akmal Khan, Rukhshanda Rehman, Junaid Abdul Wahid, and Weiyan Hou. Iot and deep learning based approach for rapid screening and face mask detection for infection spread control of covid-19. *Applied Sciences*, 11(8):3495, 2021.

[6] Ching-Pei Lee and Chih-Jen Lin. A study on l2-loss (squared hinge-loss) multiclass svm. *Neural computation*, 25(5):1302–1323, 2013.

[7] Guibiao Xu, Zheng Cao, Bao-Gang Hu, and Jose C Principe. Robust support vector machines based on the rescaled hinge loss function. *Pattern Recognition*, 63:139–148, 2017.

[8] Syafeeza Ahmad Radzi and Mohamed Khalil-Hani. Character recognition of license plate number using convolutional neural network. In *International Visual Informatics Conference*, pages 45–55. Springer, 2011.

[9] Jason Brownlee. Understand the impact of learning rate on neural network performance. *Machine Learning Mastery*, pages 1–27, 2019.

[10] Sofia Visa, Brian Ramsay, Anca L Ralescu, and Esther Van Der Knaap. Confusion matrix-based feature selection. *MAICS*, 710(1):120–127, 2011.

APPENDIX A
TABLES

	Effectiveness at preventing											
	Ancestral		Alpha		Beta		Gamma		Delta		Omicron	
Vaccine	Severe disease	Infection	Severe disease	Infection	Severe disease	Infection	Severe disease	Infection	Severe disease	Infection	Severe disease	Infection
AstraZeneca	94%	63%	94%	63%	94%	69%	94%	69%	94%	69%	71%	36%
CanSino	66%	62%	66%	62%	64%	61%	64%	61%	64%	61%	48%	32%
CoronaVac	50%	47%	50%	47%	49%	46%	49%	46%	49%	46%	37%	24%
Covaxin	78%	73%	78%	73%	76%	72%	76%	72%	76%	72%	57%	38%
Johnson & Johnson	86%	72%	86%	72%	76%	64%	76%	64%	76%	64%	57%	33%
Moderna	97%	92%	97%	92%	97%	91%	97%	91%	97%	91%	73%	48%
Novavax	89%	83%	89%	83%	86%	82%	86%	82%	86%	82%	65%	43%
Pfizer/BioNTech	95%	86%	95%	86%	95%	84%	95%	84%	95%	84%	72%	44%
Sinopharm	73%	68%	73%	68%	71%	67%	71%	67%	71%	67%	53%	35%
Sputnik-V	92%	86%	92%	86%	89%	85%	89%	85%	89%	85%	67%	44%
Other vaccines	75%	70%	75%	70%	73%	69%	73%	69%	73%	69%	55%	36%
Other vaccines (mRNA)	91%	86%	91%	86%	88%	85%	88%	85%	88%	85%	67%	45%

Fig. 13: Prevention Effectiveness Of Different Vaccines Against Different Variants Of Covid Virus

APPENDIX B
GANTT CHART

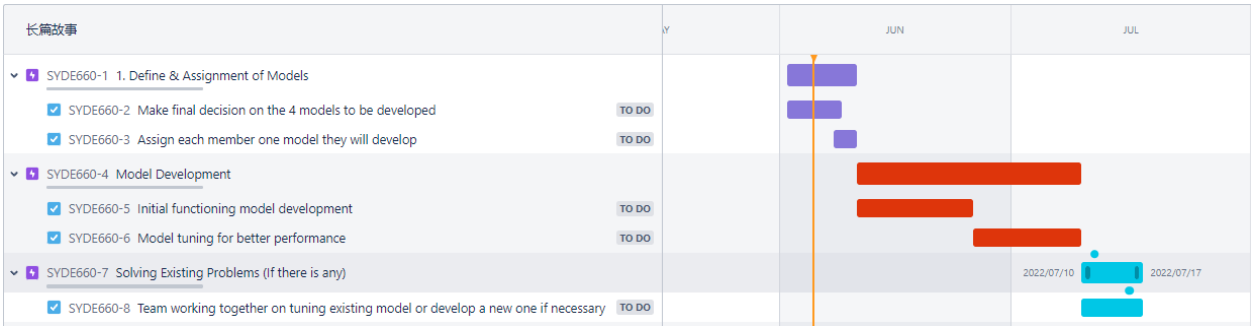
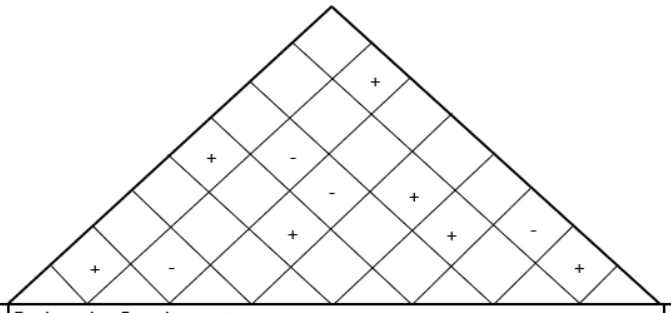


Fig. 14: Project Management Plan - Gantt Chart

APPENDIX C
QUALITY FUNCTION DEPLOYMENT (QFD) CHART



User Needs/Functions	Importance (What)	Engineering Requirements								Competitive Products		
		ML techniques with high accuracy for image classification should be employed	Should take only a few seconds to detect whether someone is wearing mask properly	Should be applicable to multiple computer hardware and operating systems	Should be friendly to people without a technical background	The algorithm's functionality should be easily to change or extend	Software and hardware need to develop the algorithm should be inexpensive	The algorithm should run stably for a long time without frequent maintenance	The algorithms should occupy only a small amount of computer memory at runtime	Security guards filtering out people without masks at entrance	Manual observation pictures of people wearing masks taken in a fixed location	Algorithms with low accuracy and delay for classify pictures of people wearing masks taken in a fixed location
High accuracy	30	9				3				1	3	4
Prompt response	20	3	9							4	2	3
Cross-platform	10			9						5	3	3
Easy to use	12		3		9			3	3	5	2	3
Adaptability	12	3		3		9				2	2	2
Low cost	8						9			4	3	4
Durability	8	3		3	3	3		9	9	1	2	3
Unit		Acc	Secs	Subj	Likert	Subj	$ CND	Years	GB			
Importance (HOW)		390	216	150	132	222	72	108	108			
Importance (%)		28	15	11	9	16	5	8	8			
Minimum Threshold		85	1	5	6	6	30	1	0.5			
Desired Target		90	3	7	8	8	100	3	2			
Maximum Threshold		99	6	10	10	10	200	5	5			

Fig. 15: QFD chart of the project.