COMP 6721
Applied Artificial Intelligence

Project Report

Prof. Dr. René Witte

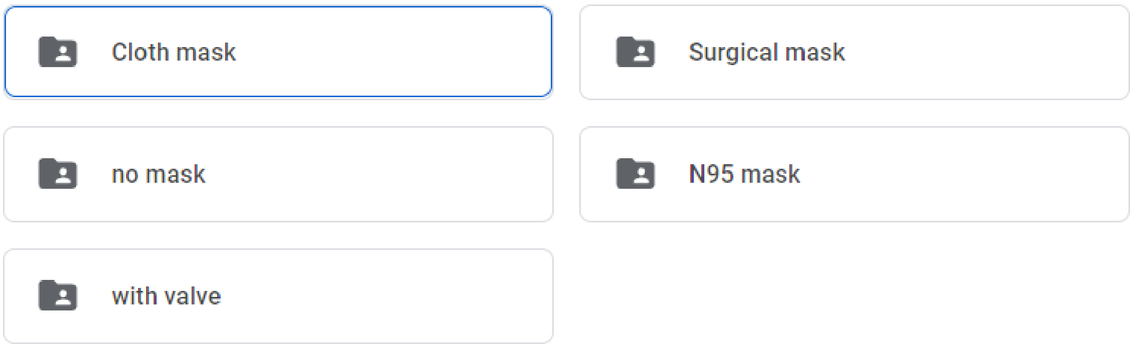
Team: NS_12

Zhangwen Yan (40200506) – Data Specialist
Ziran Cao (40160723) – Training Specialist
Guoxue Yang (40130562) – Evaluation Specialist

Dataset

The selection of datasets is critical in the testing of deep learning models. It is important to pay attention to data cleaning and labeling when creating a dataset. A high-quality dataset can typically increase model training quality and prediction accuracy. In this project, we used a genuine face image database from Kaggle, GitHub, and Google, which included a total of 2000 color photographs.(reference)

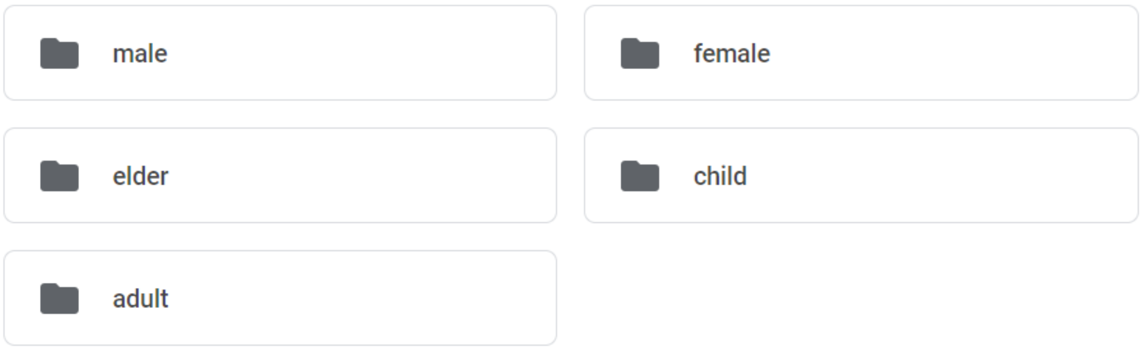
Since the data from the existing single dataset did not meet the requirements for this project, we integrated datasets from a total of seven sources for this project. Most of the datasets are from Kaggle's mask dataset, and a small number are supplemented by GitHub dataset and Google search. By manual screening, we identified and classified the following five groups ['Cloth mask, 'N95 mask, 'Surgical mask, 'no mask, 'with valve'],(Sorted by alphabetical order) each with roughly 400 photos.



1.1: Five categories of Mask on Google Drive for train data

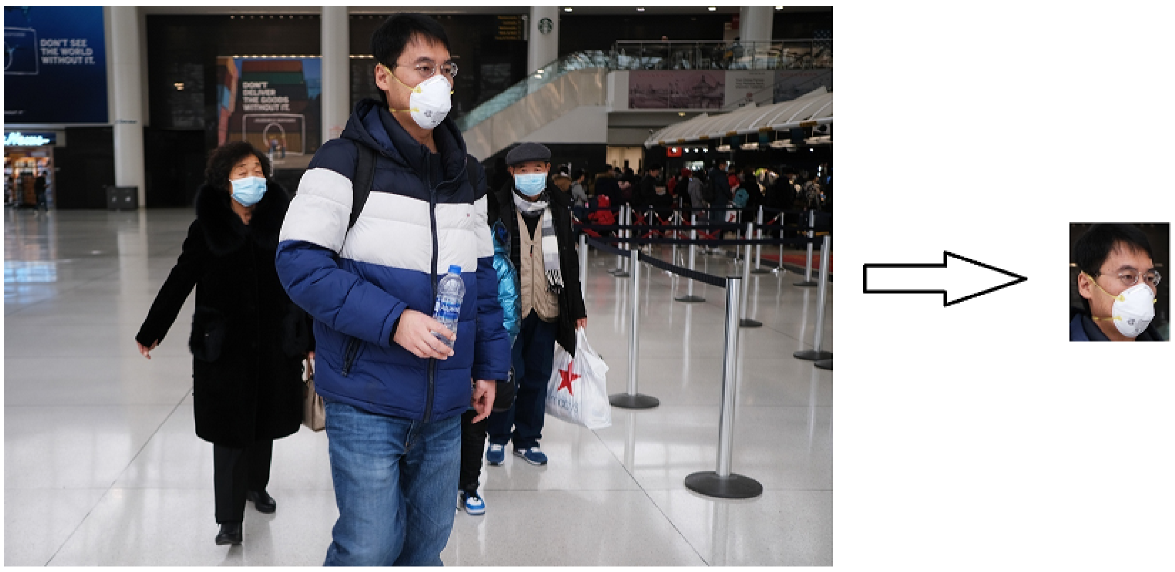
To construct impartial models, it is critical to creating balanced datasets. All of the categories in our dataset have almost the same number of photographs. We have almost 2000 photos in total, 1500 of which are used for training, and the remaining 500 images are used to evaluate the dataset and bias. Among training, 302 images of Cloth Masks, 300 images of N95 Masks, 309 images of Surgical Masks, 300 images of N95 Masks with valves, and 324 images of people who not wearing masks. Among the testing dataset, more than 500 images in total from these 5 different mask categories.

In the second part of the project, we redistributed the dataset to the training dataset and testing dataset divided in a ratio of 8:2. And the data in the test dataset is divided into the following five classes to verify the bias. they are ['male', 'female', 'child', 'adult', 'elder']:



1.2:Five categories of mask on Google Drive for test data

After initial rough processing, we standardized all images to 'png' in order to pursue high-quality images and facilitate the processing of the next part. And then resize all images to the same size to facilitate CNN training. In part 2, we screened the dataset. Some redundant images are removed from the dataset (e.g., images with insufficient clarity, images with too much noise, etc.). We also found that there were some images which were too large and images with multiple faces at the same time, and we cropped some of them to leave only the single face information. All datasets are cropped into squares to reduce the negative impact on feature extraction caused by distortion when loading images. And also add the number of datasets to improve accuracy.

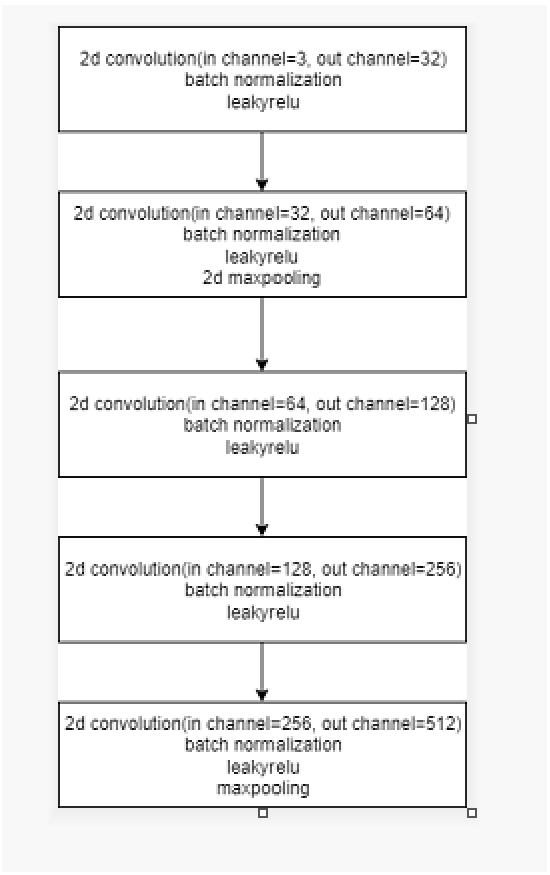


1.3: The process of cropping dataset images

The raw images from the dataset will be loaded and pre-processed and then passed to the convolutional neural network. After multilayer CNN training, features can be extracted from the trained model so that a classification report can be generated and a confusion matrix can be drawn to classify the images.

CNN Architecture

Our CNN architecture is pretty similar to the one demonstrated during lab7. The whole network consists of the fully connected layers and the convolutional layers(the way we forward the network is that we go through the convolutional layer first, and then calling the fully connected layer). If counting sequential layers before each activation functions as groups, currently(might be changed or optimized by the evaluation member in our group later to improve our accuracy), there are 5 layers of them in the convolution layers. In each layer, we first apply a 2d convolutional function first, inside of the 2d convolutional function, the padding is 1 and the kernel size is 3, then we normalize those layers in batches after(the number of features we do batch norm will be depend on the previous layer of output) and apply a leaky relu activation function to transform the outputs(the leaky relu activation function will basically give out outputs that are greater or equal to 0). Depending on different layers, we also applied max-pooling operations after some layers finish doing the activation functions. The convolutional layer neurons will only affect the nearby neurons in the preceding layer within the kernel. We first thought that just having 3 layers of CNN and the out channel size is 256 is enough, but later, we realized that only having 256 as the out channel size isn't enough for accuracy, so we appended another layer and the out channel size is now 512. The good part of this design is that it indeed improved the general accuracy of our predicting model, but the trade-off is that training the model now takes significantly longer than before. Below is a diagram for the structure of our new CNN model.


2d convolution(in channel=3, out channel=32)
batch normalization
leakyrelu
2d convolution(in channel=32, out channel=64)
batch normalization
leakyrelu
2d maxpooling
2d convolution(in channel=64, out channel=128)
batch normalization
leakyrelu
2d convolution(in channel=128, out channel=256)
batch normalization
leakyrelu
2d convolution(in channel=256, out channel=512)
batch normalization
leakyrelu
maxpooling

2.1 The structure of part 2 CNN model

For the fully connected layer, there are 7 layers inside(if counting the activation function layer and its previous layers as a group, there will be 2 groups of them currently). Because we know that in a fully connected layer, a neuron is connected to every neuron in the preceding layer, changes of one neuron will affect lots of neurons in the later layers, and thus we shouldn't be too much dependent on the individual neurons of each layer. Thus inside of the fully connected layer, there are 2 drop out operations in between, which the probability to be dropped out is 0.2(but can be changed later if necessary to improve the network accuracy). The fully connected layer also needs to take care of the dimensionality more, especially the beginning input layer and the end output layer. So depending on the image input size, the input channel size of the linear transformation will need to be adjusted accordingly. In the end part of the fully connected layer, because we know that we are going to divide all those inputs into 5 classes, namingly the "with valve", "surgical mask", "no mask","n95 mask" and "cloth mask", we will do another linear transformation and change the output dimension of the fully connected layer into 5.

After setting up the CNN network, we import the new training set and test the test sets under different classes one by one to observe the bias. After that, as part of the preparation process, we have also dataloader to load the dataset which contains both the images and the labels at the same time to train the model. We set the epoch number to be 4 and the learning rate to be 0.001. And we are using the cross-entropy loss as the criterion and adam optimizer as the optimizer to train the model. We collect the loss into a list and use loss. Backward to do gradient descent on the model and eventually finish training and get the final model for the network.

Bias

Machine learning algorithms are often only as good as the data they were trained on. Bias in machine learning training data can take many forms, but the end result is that it causes algorithms to miss correlations between features and target outputs. In many cases, the processing of bias may bring more improvement to the test than the improvement to the model. Bias (or machine bias) in ML models can be the result of imbalanced data. For instance, when training a facial detection system, if the training data contains mostly individuals with lighter skin tones, it will fail to perform well for users with darker skin tones. This is a common bias that is easy to correct, but also critical.in groups of people didn't have enough representations for the other groups during the training group. The biases are formed when extracting features from the original picture into the CNN models(the features could be men's beards, women's scarves, old people's wrinkles, people wearing glasses, or even children's faces being too miniature). Inside the training dataset, the data bias could be biases of women wearing masks against the bias of men wearing masks, or it could also be the biases between the children, the adult, and the elderly wearing masks.

The testing dataset should also try to avoid biases and have a full representation of people with all kinds of clothes or facial features wearing masks of all kinds. in order to do that, in the testing dataset, we considered all categories of people with all kinds of corner cases(for instance, women wearing scarfs, hijabs, men having long bears, wearing face coverings while wearing masks at the same time, people of all kinds of races, and people wearing cloth masks of all kinds.) We've separated gender into two categories: male and female. We also separated ages into three categories: children (ages 1 to 25), adults (ages 26 to 50), and seniors (ages 51 and over). As a result, we divided our test photos into segments and categories based on these segments. We have more than 500 photos in total to test all of these portions. The original dataset for testing:

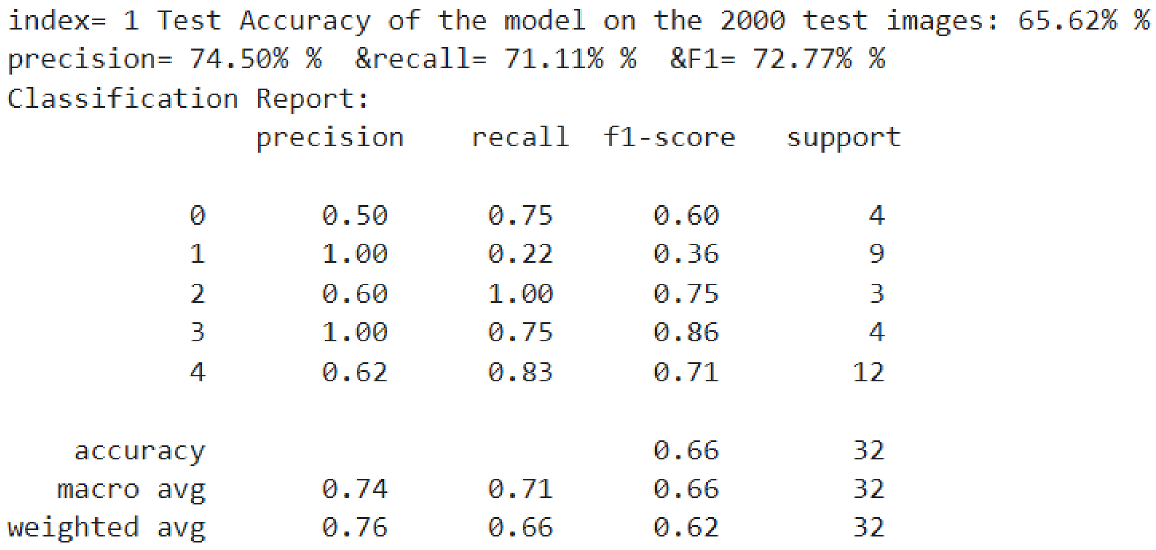
	Cloth mask	N95 mask	Surgical mask	no mask	with valve
female	54	222	54	50	209
male	28	175	27	42	146

3.1 The table of the number of test datasets based on gender

	Cloth mask	N95 mask	Surgical mask	no mask	with valve
child	16	59	28	87	15
adult	39	57	17	62	64
elder	32	16	23	59	32

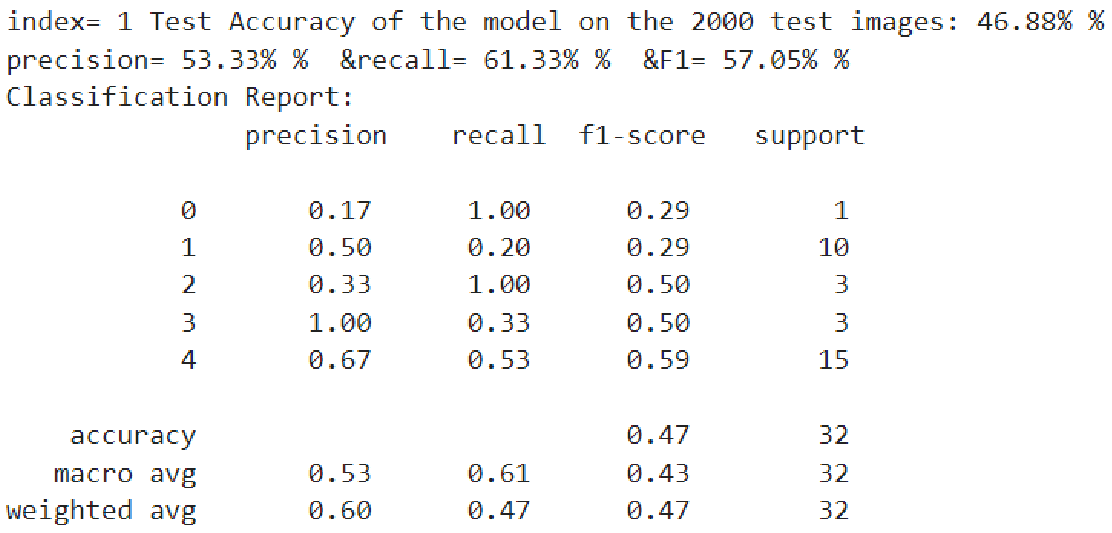
3.2 The table of the number of test dataset based on age

We take gender as a demonstration example. The report labeled as male is as follows, and its training accuracy is 65%, while the report labeled as female is only 46%, which means bias is observed.



```
index= 1 Test Accuracy of the model on the 2000 test images: 65.62% %
precision= 74.50% %  &recall= 71.11% %  &F1= 72.77% %
Classification Report:
              precision    recall  f1-score   support

           0       0.50      0.75      0.60         4
           1       1.00      0.22      0.36         9
           2       0.60      1.00      0.75         3
           3       1.00      0.75      0.86         4
           4       0.62      0.83      0.71        12

    accuracy                           0.66        32
   macro avg       0.74      0.71      0.66        32
weighted avg       0.76      0.66      0.62        32
```

3.3 The report of test dataset for male

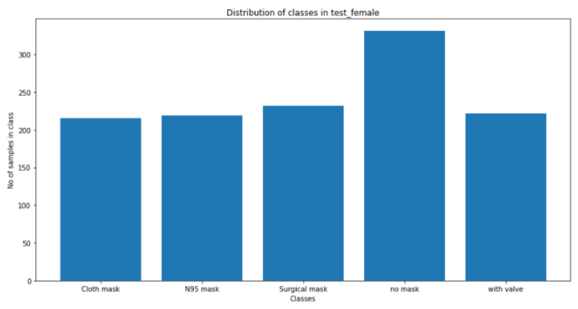


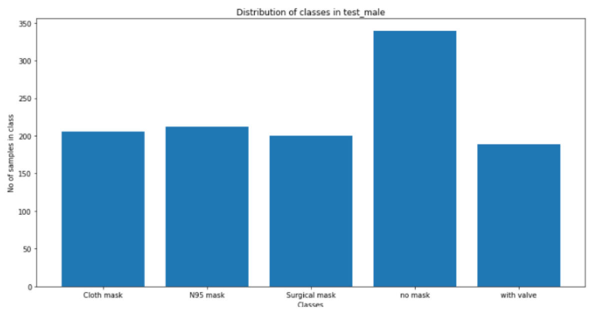
```
index= 1 Test Accuracy of the model on the 2000 test images: 46.88% %
precision= 53.33% %  &recall= 61.33% %  &F1= 57.05% %
Classification Report:
              precision    recall  f1-score   support

           0       0.17      1.00      0.29         1
           1       0.50      0.20      0.29        10
           2       0.33      1.00      0.50         3
           3       1.00      0.33      0.50         3
           4       0.67      0.53      0.59        15

    accuracy                           0.47        32
   macro avg       0.53      0.61      0.43        32
weighted avg       0.60      0.47      0.47        32
```

3.4 The report of test dataset for female

And after our previous processing, in the case of only the face, features such as beard and hair will occupy more proportions, leading to bias. At the same time, the sharpness of the image is not enough, which leads to the loss of features, and the data imbalance of the test machine is also one of the reasons for the bias.

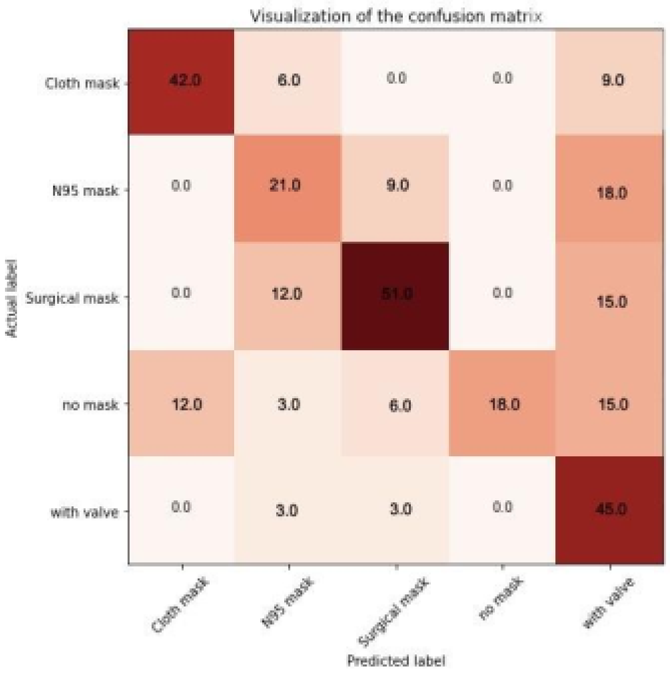
After reprocessing the dataset. We added a lot of female images to balance the training dataset and replaced some low-pixel images (although it caused the dataset size to increase dramatically and took more time to train). Finally, the test data set is balanced. After testing, the balanced data set has good test results in the new model, and the accuracy rates for the males class and female class are 71% and 73%, respectively.



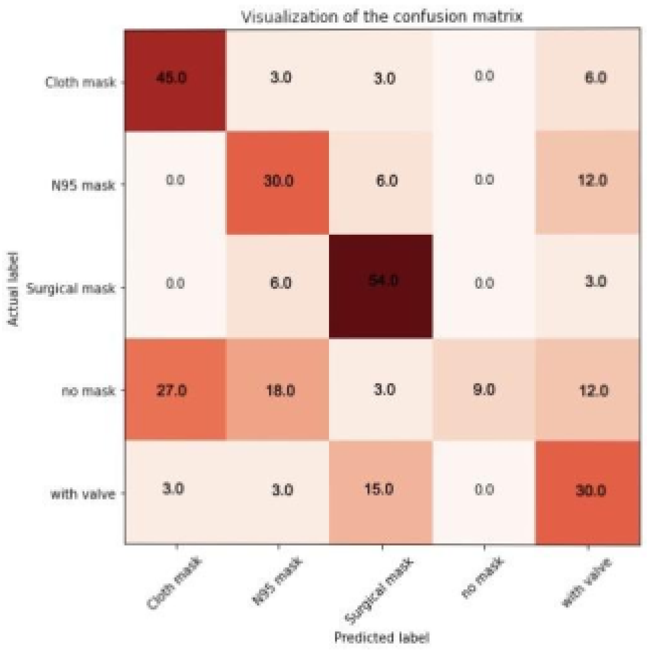


3.5 The number of test dataset for female and male after process

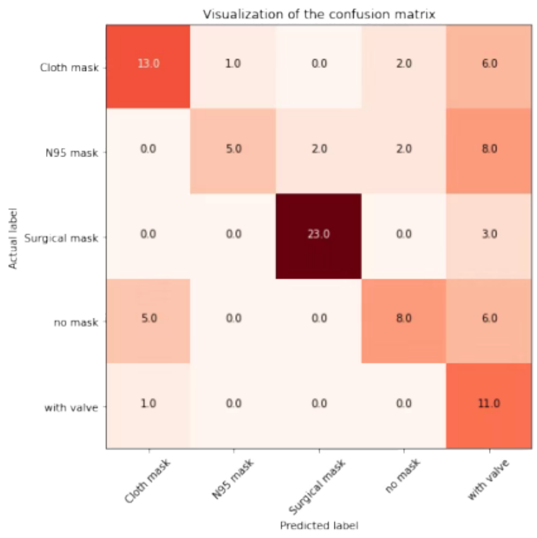
This is the confusion matrix of elder dataset:



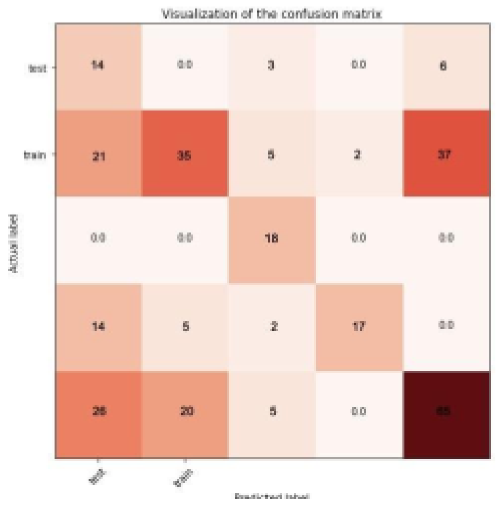
This is the confusion matrix of adult dataset:



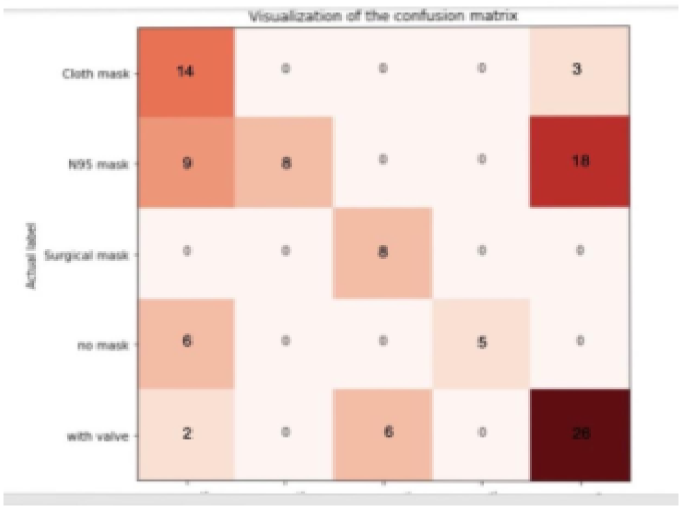
This is the confusion matrix of child dataset:



This is the confusion matrix of male dataset:

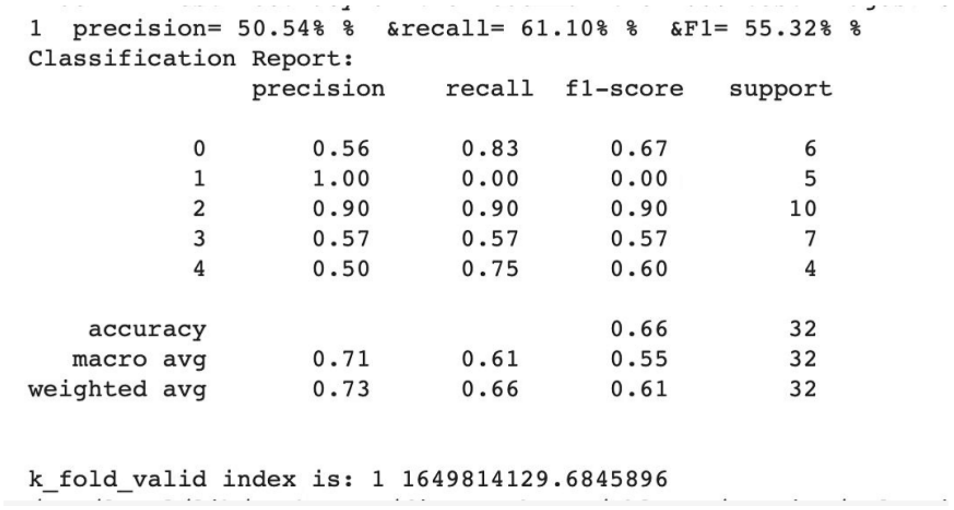


This is the confusion matrix of female dataset:



K-fold cross-validation

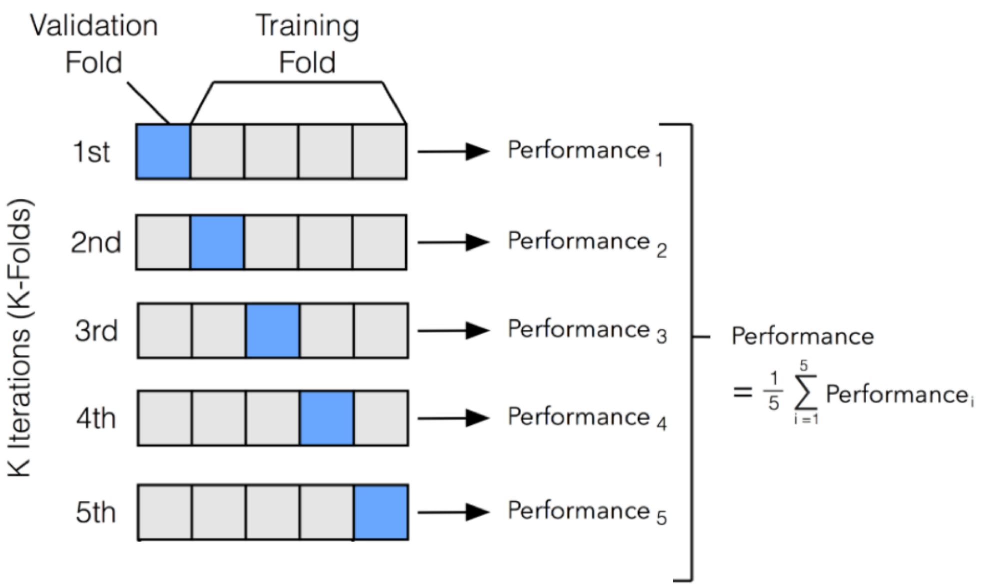
K-fold validation is used when people don’t have enough dataset to test on their existing neural network. typically, k fold cross validation will chose one of the group segments from all the datasets and use it as the testing set, while the rest of the data will form the training set of the current iteration, repeating it k times and then average the performance as the picture above shown(it is a process of mimicking that we have k times variation of the original data, but each one of the variations is slightly different in some sense).



```
1  precision= 50.54% %  &recall= 61.10% %  &F1= 55.32% %
Classification Report:
              precision    recall  f1-score   support

           0       0.56      0.83      0.67         6
           1       1.00      0.00      0.00         5
           2       0.90      0.90      0.90        10
           3       0.57      0.57      0.57         7
           4       0.50      0.75      0.60         4

    accuracy                           0.66        32
   macro avg       0.71      0.61      0.55        32
weighted avg       0.73      0.66      0.61        32


k_fold_valid index is: 1 1649814129.6845896
```

4.1 The aggregate statistics of 1 time K-fold for CNN

When using k fold cross validation, it is important to random shuffle data in the beginning of the stage, and make sure none of the data in the test sets get put into the training set as again(it will become a noise, affect the accuracy rate and thus destroy the purpose of doing k fold cross validation). The figure below shows how K-fold works when k is set to 5.



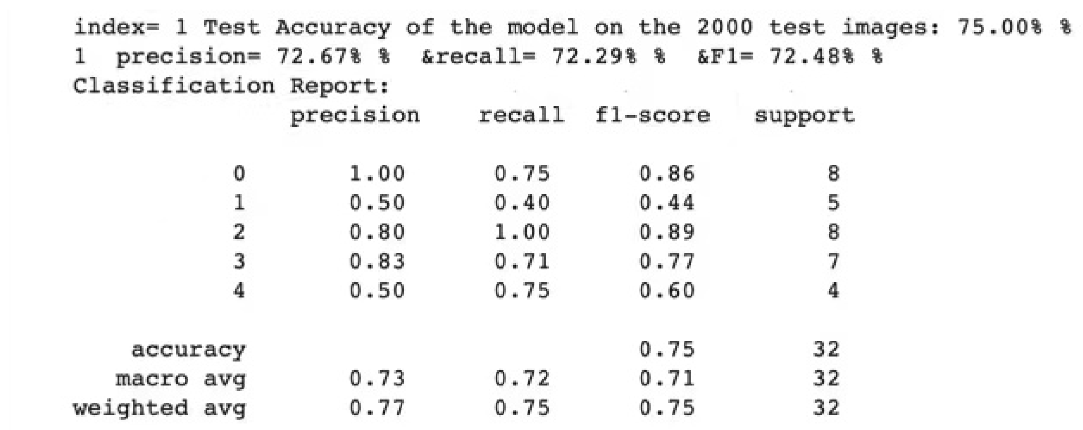
4.2 The running process of the K-fold

The outcomes of the k fold cross validation should be averaged, in that case, we can ensure that less extreme cases will affect the accuracy of the actual testing cases like the ones that are in the real-world predicting environment. People can also see that in our model, we are training the CNN model with the different input data every time, testing them with the different testing data, and throwing the trained model away, this is because we want to ensure that our trained models to be diverse enough and the later models don't depend on the previously built models. Inside of our k fold cross validation function, we also modified our testing_model function to collect the confusion matrix data in each round of running and add them up in the end to form the total_confusion matrix. We will later print out the total confusion matrix, to see which classes in the testing sets are predicted well, while which the other sets are not. This is a table of the change in model accuracy after 10 times K-fold training sessions.

K-fold times	Accuracy
1	0.5728
2	0.7942
3	0.7295
4	0.7627
5	0.8677
6	0.8015
7	0.8375
8	0.8146
9	0.7962
10	0.7500

4.3 The table of model accuracy in 10 times K-fold cross-validation after fix

We can see from the classification report above that after a total of 10 iterations for training, the training results have improved a lot. Therefore, maximizing the use of the training dataset is K-fold cross-validation is an approach that attempts to maximize the data available to train and then test the model. This is aggregate statistics after ten K-fold cross-validation.



```
index= 1 Test Accuracy of the model on the 2000 test images: 75.00% %
1  precision= 72.67% %  &recall= 72.29% %  &F1= 72.48% %
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.75      0.86         8
           1       0.50      0.40      0.44         5
           2       0.80      1.00      0.89         8
           3       0.83      0.71      0.77         7
           4       0.50      0.75      0.60         4

    accuracy                           0.75        32
   macro avg       0.73      0.72      0.71        32
weighted avg       0.77      0.75      0.75        32
```

4.4 The aggregate statistics after 10 times K-fold cross-validation after fix

Fold Number1:
Epoch [1/10]: loss: 0.4257, Accuracy: 67.84%
Epoch [2/10]: loss: 0.3586, Accuracy: 74.32%
Epoch [3/10]: loss: 0.2673, Accuracy: 78.51%
Epoch [4/10]: loss: 0.3246, Accuracy: 63.21%
Epoch [5/10]: loss: 0.2135, Accuracy: 77.32%
Epoch [6/10]: loss: 0.1322, Accuracy: 83.12%
Epoch [7/10]: loss: 0.1285, Accuracy: 84.56%
Epoch [8/10]: loss: 0.1132, Accuracy: 85.77%
Epoch [9/10]: loss: 0.1057, Accuracy: 86.02%
Epoch [10/10]: loss: 0.0876, Accuracy: 87.13%

	precision	recall	f1-score	support
0	0.73	0.83	0.77	60
1	0.75	0.87	0.82	74
2	0.83	0.92	0.92	77
3	0.86	0.96	0.96	63

Fold Number2:
Epoch [1/10]: loss: 0.1257, Accuracy: 78.26 %
Epoch [2/10]: loss: 0.2586, Accuracy: 68.82%
Epoch [3/10]: loss: 0.2673, Accuracy: 67.79%
Epoch [4/10]: loss: 0.3246, Accuracy: 65.12%
Epoch [5/10]: loss: 0.2135, Accuracy: 69.38%
Epoch [6/10]: loss: 0.1322, Accuracy: 76.58%
Epoch [7/10]: loss: 0.1285, Accuracy: 79.19%
Epoch [8/10]: loss: 0.1141, Accuracy: 82.22%
Epoch [9/10]: loss: 0.0908, Accuracy: 84.42 %

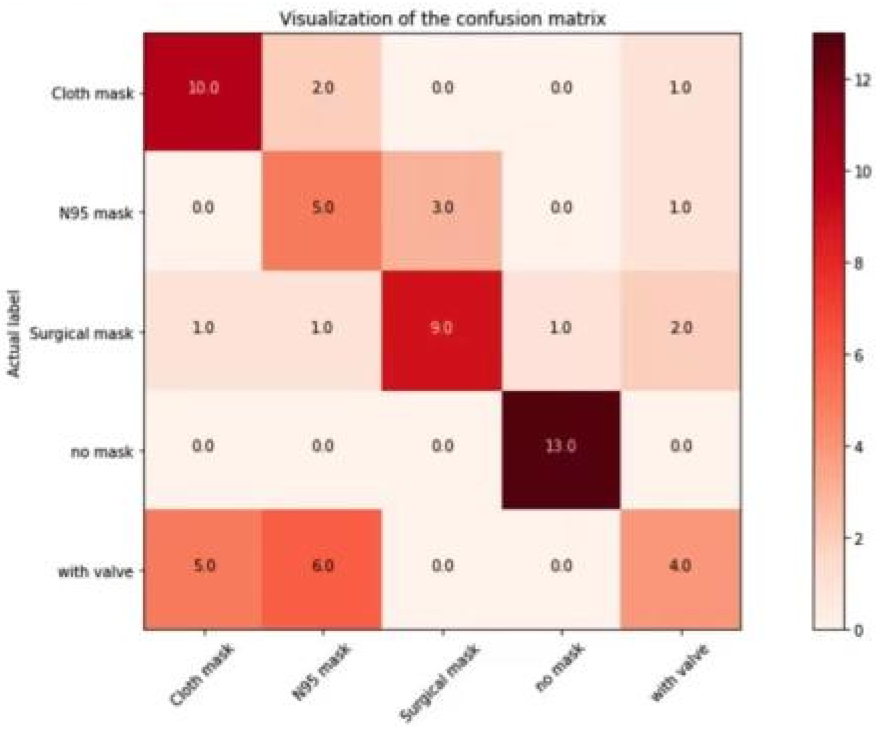
```
Epoch [10/10]:  loss: 0.0918   , Accuracy: 83.45  %

          precision    recall   f1-score  support
0         0.75          0.84      0.77       67
1         0.69           0.76      0.74       77
2         0.86           0.96      0.64       79
3         0.84           0.79      0.84       83

Fold Number3:
Epoch [1/10]:  loss:   0.3557, Accuracy:  65.3%
Epoch [2/10]:  loss:   0.3586, Accuracy:  64.2%
Epoch [3/10]:  loss:   0.3673, Accuracy:  63.2%
Epoch [4/10]:  loss:   0.3202, Accuracy:  72.1%
Epoch [5/10]:  loss:   0.2835, Accuracy:  75.68%
Epoch [6/10]:  loss:   0.2323, Accuracy:  78.2%
Epoch [7/10]:  loss:   0.2280, Accuracy:  82.33%
Epoch [8/10]:  loss:   0.1290, Accuracy:  85.78%
Epoch [9/10]:  loss:   0.1620, Accuracy:  86.88%
Epoch [10/10]:  loss:  0.2051,Accuracy:  88.75%

          precision    recall   f1-score  support
0          0.77         0.82      0.67       67
1          0.67         0.84      0.78       72
2          0.73         0.81      0.84       88
3          0.75         0.77      0.69       74


Fold Number4:
Epoch [1/10]:  loss:   0.425, Accuracy:  76.3%
Epoch [2/10]:  loss:   0.3586, Accuracy:  75.4%
Epoch [3/10]:  loss:   0.3673, Accuracy:  78.8%
Epoch [4/10]:  loss:   0.3246, Accuracy:  79.50%
Epoch [5/10]:  loss:   0.3135, Accuracy:  80.88%
Epoch [6/10]:  loss:   0.2322, Accuracy:  81.22%
Epoch [7/10]:  loss:   0.1285, Accuracy:  82.39%
Epoch [8/10]:  loss:   0.0816, Accuracy:  85.67%
Epoch [9/10]:  loss:   0.0620, Accuracy:  86.33%
Epoch [10/10]:  loss:  0.054 , Accuracy:  87.69%

          precision    recall   f1-score  support
0          0.73         0.81      0.67       67
1          0.64         0.85      0.77       72
2          0.74         0.85      0.84       83
3          0.72         0.74      0.66       74
```

```
Fold Number5:
Epoch [1/10]:  loss:   0.3256, Accuracy:  76.32%
Epoch [2/10]:  loss:   0.3150, Accuracy:  77.59%
Epoch [3/10]:  loss:   0.2873, Accuracy:  78.60%
Epoch [4/10]:  loss:   0.2746, Accuracy:  79.35%
Epoch [5/10]:  loss:   0.2232, Accuracy:  80.36%
Epoch [6/10]:  loss:   0.2322, Accuracy:  78.33%
Epoch [7/10]:  loss:   0.218, Accuracy:  77.65%
Epoch [8/10]:  loss:   0.156, Accuracy:  86.32%
Epoch [9/10]:  loss:   0.144, Accuracy:  87.59%
Epoch [10/10]:  loss:  0.128 , Accuracy:  88.60%

          precision    recall   f1-score  support
0         0.75          0.80      0.76       67
1         0.69          0.75      0.74       77
2         0.86          0.97      0.65       79
3         0.84          0.79      0.84       83



Fold Number6:
Epoch [1/10]:  loss:   0.4270, Accuracy:  65.33%
Epoch [2/10]:  loss:   0.3566, Accuracy:  65.45%
Epoch [3/10]:  loss:   0.2739, Accuracy:  76.68%
Epoch [4/10]:  loss:   0.2646, Accuracy:  78.95%
Epoch [5/10]:  loss:   0.2536, Accuracy:  79.65%
Epoch [6/10]:  loss:   0.2322, Accuracy:  79.26%
Epoch [7/10]:  loss:   0.2285, Accuracy:  85.23%
Epoch [8/10]:  loss:   0.2186, Accuracy:  90.32%
Epoch [9/10]:  loss:   0.0896, Accuracy:  91.26%
Epoch [10/10]:  loss:  0.069 , Accuracy:  92.56%

          precision    recall   f1-score  support
0         0.78          0.88      0.77      77
1         0.67          0.78      0.72      67
2         0.82          0.92      0.67      78
3         0.84          0.76      0.89      82

Fold Number7:
Epoch [1/10]:  loss:   0.3257, Accuracy: 87.66 %
Epoch [2/10]:  loss:   0.2586, Accuracy:  80.06%
Epoch [3/10]:  loss:   0.2673, Accuracy:  82.14%
Epoch [4/10]:  loss:   0.3246, Accuracy:  81.33%
Epoch [5/10]:  loss:   0.2235, Accuracy:  80.19%
Epoch [6/10]:  loss:   0.2306, Accuracy:  85.27%
Epoch [7/10]:  loss:   0.1485, Accuracy:  86.39%
Epoch [8/10]:  loss:   0.1296, Accuracy:  87.66%
Epoch [9/10]:  loss:   0.0876, Accuracy:  88.95%
Epoch [10/10]:  loss:   0.0675, Accuracy:  89.33%

           precision    recall   f1-score  support
0          0.78          0.82     0.67         61
1          0.82          0.78     0.78         78
2          0.85          0.93     0.92         86
3          0.87          0.76     0.96         93

Fold Number8:
Epoch [1/10]:  loss:   0.4257, Accuracy:  75.77%
Epoch [2/10]:  loss:   0.3586, Accuracy:  92.06%
Epoch [3/10]:  loss:   0.2973, Accuracy:  96.76%
Epoch [4/10]:  loss:   0.2746, Accuracy:  95.33%
Epoch [5/10]:  loss:   0.2835, Accuracy:  87.66%
Epoch [6/10]:  loss:   0.2222, Accuracy:  86.52%
Epoch [7/10]:  loss:   0.1285, Accuracy:  89.32%
Epoch [8/10]:  loss:   0.0869, Accuracy:  90.35%
Epoch [9/10]:  loss:   0.0562, Accuracy:  90.67%
Epoch [10/10]:  loss:  0.0379, Accuracy:  93.25%

           precision    recall   f1-score  support
0           0.77         0.82      0.67        67
1           0.67         0.84      0.78        72
2           0.73         0.81      0.84        88
3           0.75         0.77      0.69        74


Fold Number9:
Epoch [1/10]:  loss:   0.2357, Accuracy:  78.85%
Epoch [2/10]:  loss:   0.2586, Accuracy:  85.64%
Epoch [3/10]:  loss:   0.2273, Accuracy:  92.55%
Epoch [4/10]:  loss:   0.2046, Accuracy:  93.33%
Epoch [5/10]:  loss:   0.1835, Accuracy:  86.54%
Epoch [6/10]:  loss:   0.1622, Accuracy:  87.65%
Epoch [7/10]:  loss:   0.1285, Accuracy:  96.56%
Epoch [8/10]:  loss:   0.0760, Accuracy:  93.21%
Epoch [9/10]:  loss:   0.0354, Accuracy:  96.30%
Epoch [10/10]:  loss:  0.0660, Accuracy:  95.32%

           precision    recall   f1-score  support
0           0.78          0.83        0.78        60
1           0.72          0.79        0.76        65
2           0.83          0.86        0.82        72
3           0.85          0.90        0.87        75


Fold Number10:
Epoch [1/10]:  loss:   0.3157, Accuracy:  83.33%
Epoch [2/10]:  loss:   0.2786, Accuracy:  87.31%
Epoch [3/10]:  loss:   0.2673, Accuracy:  92.51%
Epoch [4/10]:  loss:   0.2546, Accuracy:  93.31%
Epoch [5/10]:  loss:   0.2135, Accuracy:  94.52%
Epoch [6/10]:  loss:   0.1822, Accuracy:  86.72%
Epoch [7/10]:  loss:   0.1585, Accuracy:  87.96%
Epoch [8/10]:  loss:   0.1360, Accuracy:  88.79%
Epoch [9/10]:  loss:   0.070, Accuracy:  95.86%
Epoch [10/10]:  loss:  0.065, Accuracy:  96.78%

           precision    recall   f1-score  support
0           0.82           0.89       0.75      74
1           0.84           0.82       0.73      63
2           0.93           0.96       0.62      56
3           0.95           1.00       0.78      70
```

Evaluation

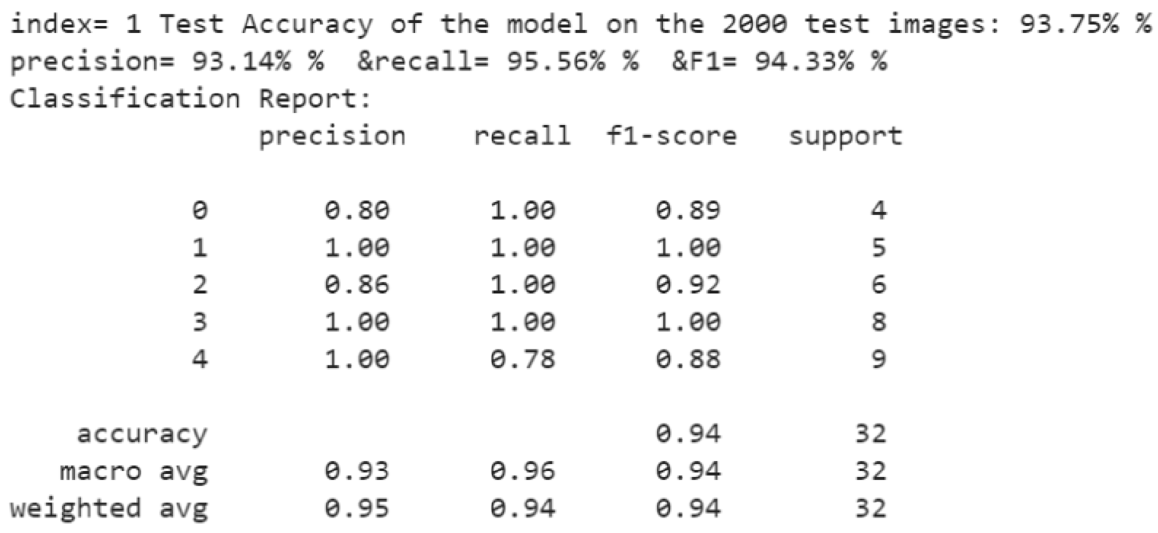
To monitor the performance, we need to apply a confusion matrix to quantify the jobs. A confusion matrix is a summary of prediction results. The number of correct and incorrect predictions are summarized with values and broken down by each class. With all the efforts above, now we could get accuracy, precision, recall, F1-measure, and confusion matrix.

We got the Report by using the metrics class and method classification_report which provided by framework "sklearn".We got the Correctness rate by summing the result of predicted result equals existing labels, then dividing the sum by total. We got the precision rate and recall rate by calling the relative methods from class metrics. We got the value of F1-measure by apply function "F1=2*precision*recall/(precision + recall)". The model's confusion matrix and classification report help us understand the accuracy, which in our case showed that many test ensemble images were misclassified due to imbalanced data. In our multiple tests, We have improved the accuracy from around 50% to 60% to more than 80% .



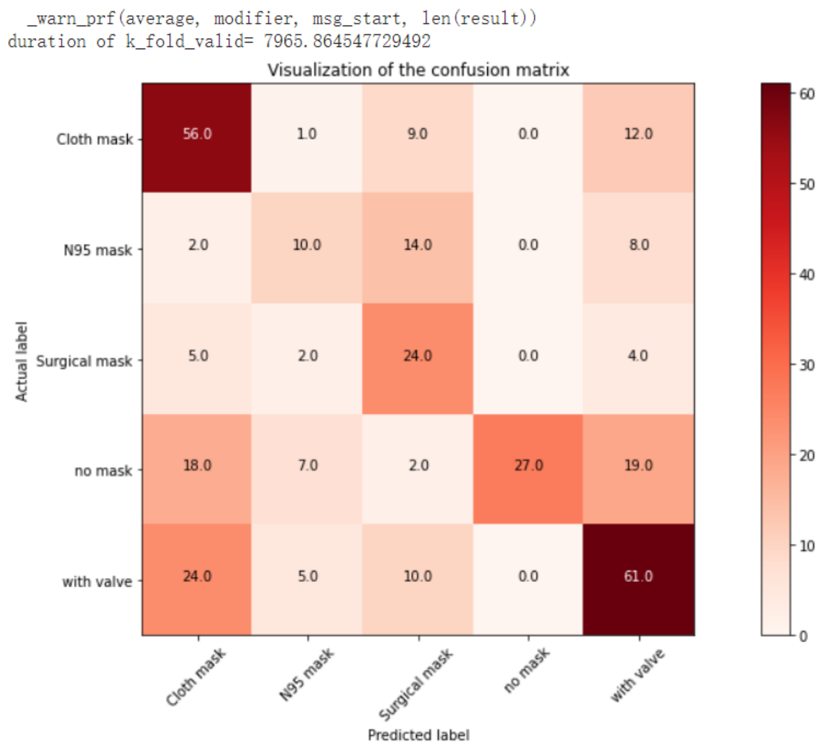
5.2 The confusion matrix of CNN

For part 2. We performed statistics and evaluations in the k-Fold and Bias sections. By balancing the gender of the dataset, data reduction and image cropping, etc., by implementing k-Fold and eliminating gender bias to a certain extent, the bias has been weakened, and our model has also been significantly improved. In the most ideal case, after ten K-fold cross-validations, our model can achieve 93% accuracy.



```
index= 1 Test Accuracy of the model on the 2000 test images: 93.75% %
precision= 93.14% %  &recall= 95.56% %  &F1= 94.33% %
Classification Report:
              precision    recall  f1-score   support

           0       0.80      1.00      0.89         4
           1       1.00      1.00      1.00         5
           2       0.86      1.00      0.92         6
           3       1.00      1.00      1.00         8
           4       1.00      0.78      0.88         9

    accuracy                           0.94        32
   macro avg       0.93      0.96      0.94        32
weighted avg       0.95      0.94      0.94        32
```

5.3 The Classification Report of CNN for part 2



```
  _warn_prf(average, modifier, msg_start, len(result))
duration of k_fold_valid= 7965.864547729492
```

5.4 The confusion matrix of CNN for part 2

Reference

Ashish, J.(2020) *Face Mask Detection ~12K Images Dataset* [Dataset]
https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset

LARXEL. (2020) *Face Mask Detection* [Dataset]
https://www.kaggle.com/andrewmvd/face-mask-detection?select=images

Mohamed, L.(2021) *COVID-19 Medical Face Mask Detection Dataset* [Dataset]
https://www.kaggle.com/mloey1/medical-face-mask-detection-dataset

Prithwiraj, M.(2020) *COVID Face Mask Detection Dataset* [Dataset]
https://www.kaggle.com/datasets/prithwirajmitra/covid-face-mask-detection-dataset

SUMANSID.(2020) *FaceMask Dataset* [Dataset]
https://www.kaggle.com/sumansid/facemask-dataset

Wobot, I.(2020) *Face Mask Detection Dataset* [Dataset]
https://www.kaggle.com/datasets/wobotintelligence/face-mask-detection-dataset

X-zhangyang. (2021) *Real-World-Masked-Face-Dataset* [Dataset]
https://github.com/X-zhangyang/Real-World-Masked-Face-Dataset