CIS 522 Spring 2021, Project Report

Face Mask Detection Classification Using CNN and ResNet

Team Members:

- Xi He; Email: `xhe93@seas.upenn.edu`
- Yijie Fan; Email: `yjf@seas.upenn.edu`

Abstract

COVID-19 is a contagious and epidemic disease that is threatening human lives currently. Wearing masks in public settings protects people from COVID-19 transmission. Moreover, the incorporation of machine learning and deep learning algorithms shows significance in face recognition and image classification. In this project, we aim to build a Face Mask Detection model in order to identify whether the person is wearing a mask or not based on face images. We constructed the Logistic Regression as our baseline model. We created the Vanilla CNN model with hyperparameter tuning. We also implemented transfer learning with ResNet18 architecture. As a result, transfer learning with ResNet18 architecture shows the highest test accuracy–0.976 and is considered as the best.

1 Motivation

Recently, COVID-19 is an epidemic disease that is threatening human lives. Based on the COVID-19 information provided by World Health Organization in April, about 50,000 people in the United States are still diagnosed with a COVID-19 every day[1]. We are still in the midst of an ongoing pandemic. Even though COVID-19 vaccines are available recently in the United States, scientists are still tracking the emergence of virus variants that can cause a long-term threat to humanity. As Jaimie Meyer says, a Yale Medicine infectious disease specialist, "Until we're able to vaccinate most people against COVID-19, we have to continue to wear facial coverings."[2] Centers for Disease Control and Prevention (CDC) points out that wearing a face mask is a convenient and effective method to prevent people from getting and spreading COVID-19 since wearing a mask provides a barrier that keeps respiratory droplets from spreading[3]. CDC highly encourages people to wear masks in public settings.
In order to follow the recommendation of the CDC and ensure the safety of customers and employees, most indoor places, such as restaurants, markets, and shopping malls, require customers to wear masks. However, some people still do not want to wear masks in public areas. Their behavior can risk other people in public areas, especially in a closed indoor place. Therefore, a lot of indoor places send workers to stand at the front doors to confirm customers are wearing masks. This method is time-consuming and can be a waste of human resources. Recently, AI models have shown promising results in the finding of an object in images. Therefore, the goal of this project is to construct several Machine Learning and Deep Learning models in order to identify whether the person is wearing a mask or not based on face images. These machine learning and deep learning models can help restaurants, hospitals, and many other indoor places to detect people who are not wearing a mask, save human resources and time, and provide a safe environment for customers and employees under the COVID-19 pandemic.

2 Social Impact

With an AI system that could classify whether a customer is wearing a mask, we could save human resources, especially for places where there is a large customer flow. The Face Mask Detection model not only can automatically find people who are not wearing masks but also can perform this task quickly. This AI system will help stores, hospitals, and other indoor places to save employees and time to check whether customers are wearing masks. Also, AI systems could generate anonymous statistical data that will help authorities anticipate future outbreaks of COVID-19. For example, it can bring answers to questions like what percentage of people are wearing masks. It can predict areas at high risk of COVID-19 outbreaks. The AI system may also have some drawbacks and biases, one is that the classifier may discriminate against people having scars or tattoos on faces. The classifier may also cause biases toward people who are from minority classes. As a result, the classifier may misclassify and cause inconvenience for them. Therefore, we should try our best to get rid of these disadvantages during both the training and development of the AI system.

3 Data set

3.1 Initial Data Statistics and Visualization

This project is using a face mask image dataset from the Kaggle competition, and the dataset was created by Ashish Jangra, with part of the dataset scrapped from Jessica Li's Celebface dataset [4].The dataset contains 11792 face images, half of them are wearing masks and half of them are not wearing masks. All the images with face masks are scrapped from google search, and all the images without face masks are gathered from CelebFace dataset generated by Jessica Li. The size of images varies from image to image, but many of them have pixel size 128×128. The dataset is diverse, given that these dataset include images of people from different genders and races background, having different gestures, and wearing different kind of masks. Some of images are even from cartoons, which enables to train a model that performs well on many different situations.

Before running machine learning codes, the detailed data inside is analyzed. We would like to take a look at some of images, making sure these dataset is valid for training. In addition, we need to check that images from wearing mask class is similar from images from not wearing mask class. For imbalance problem, since it the dataset contains about 6k images wearing masks and 6k images not wearing masks, the dataset is balanced.

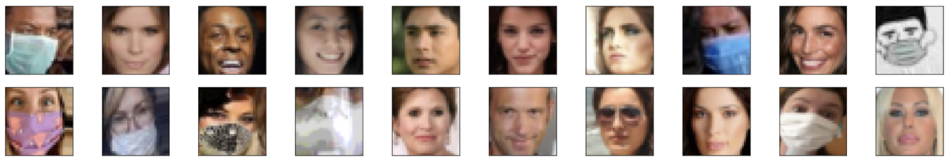


Figure 1: Sample images from dataset

3.2 Data Pre-processing

For this project, since we are performing two kinds of classification methods: non-deep learning benchmark and CNN related deep learning methods on our dataset, we need to perform different ways of pre-processing for two kinds methods. To ensure consistency, and for convenient training process, we first resize all images into 32×32 pixels. With images of same size, we could perform both Logistic Regression method and CNN methods on them. Then, we divide images into training, validation, and testing. Training set contains 10k images, 5k of them wearing masks and 5k of them not wearing masks. Validation set contains 800 images, 400 wearing masks and 400 not wearing masks. Test set contains 992 images, 483 wearing masks and 509 not wearing masks.

For Logistic Regression, since each image has RGB channels, we first perform greyscale operation onto the image, so that the result image has $32 \times 32 = 1024$ elements. Doing so could avoid the overfitting problem, since too many features would increase the complexity of the model. Given that we have about 10k images for training, 1024 features should be enough.In addition, as Logistic Regression model does not require validation process, so we combine the training set and the validation set to form a training set of size 10800 for Logistic Regression.

For methods using CNN, we resize each image into size of 32×32, normalize the dataset, and form three dataloaders with batch size of 128. We use images in the training dataloader to train the CNN model in each epoch, use images in validation dataloader to have an understanding about the performance of the model in the training process during each epoch, and use images in the test dataloader to test the model after the training is finished.

4 Problem Formulation

4.1 Target Variable Y and Supervised Learning

We formulate a Face Mask Detection Classification as a supervised binary classification problem. Y-target in this problem is whether or not the person in the image is wearing a mask. The dataset contains 6883 images of people wearing masks and 6909 images of people not wearing masks. In this project, 0 represents wearing masks and 1 represents not wearing masks, since we are interested in people not wearing masks.

4.2 X Features

The original X features are just images with different sizes, separating into wearing masks and not wearing masks category. We resize the images into 32×32. For Logistic, we use the greyscale image, so that X feature has size of 1024 for each image. For CNN methods, X features is the whole image, with size $32 \times 32 \times 3$.

4.3 Evaluation

For measuring the accuracy of models, we will just use accuracy. For Logistic Regression, we measure the number of images classified correctly. Since the dataset is balanced, we do not need to use measurements such as F1 score.

$$accuracy = \frac{\#\ of\ images\ classified\ correctly}{\#\ of\ total\ images}$$

For Logistic Regression, we measure only the test accuracy, while for CNN methods, we measure training accuracy and validation accuracy for each epoch, and then we measure test accuracy after training is finished.

4.4 Loss Function

For this project, we will use the binary cross-entropy loss function for all CNN methods, since this project is a binary classification problem and outputs of CNN methods will be probability values between 0 and 1.

5 Methods

The machine learning models used are Binary Logistic Regression, Vanilla CNN, and ResNet18 transfer learning.

5.1 Baseline: Binary Logistic Regression

Logistic Regression is widely used in the binary classification method. It is a supervised machine learning model. It is a special class of generalized linear model[5]. The Binary Logistic Regression model is simple and very efficient to train. We use the pixels from greyscale image to train the binary logistic regression model. We use it as a baseline for this project.

5.2 Vanilla CNN

A Convolutional Neural Network (CNN) is a Deep Learning algorithm that can take in an input image, assign importance to various objects in the image. Therefore, CNN is most commonly applied to identify and classify images. In this project, we select Vanilla or Classic CNN because it is currently the most widely used method for image classification. A Vanilla CNN contains 2 convolutional layers, an activation operation following each convolutional layer, 2 pooling layers especially the Max Pooling layer, and several fully connected layers. The basic Vanilla CNN's architecture is shown in Figure 2. We use this model as a baseline model to indicates whether CNN methods work for this problem. This deep learning architecture is commonly used because it contains several pooling layers to reduce dimensionality and it is not a very deep and quick image classification algorithm.

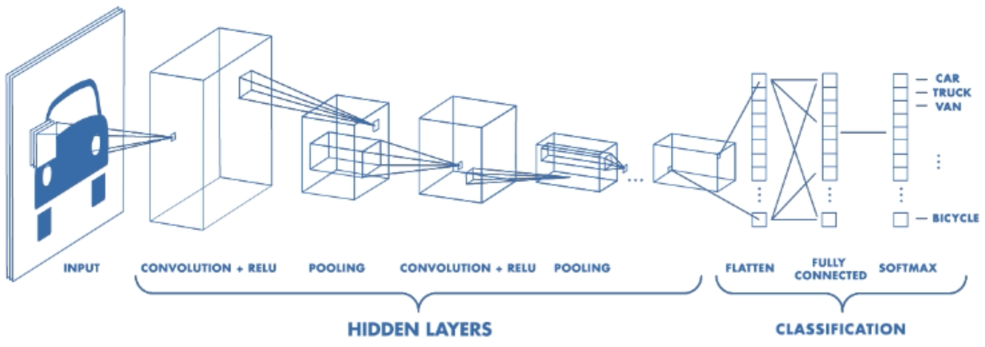


Figure 2: Vanilla CNN architecture

5.3 ResNet18 Transfer Learning

A residual network, or ResNet for short, is an artificial neural network that helps to build deeper neural network by utilizing skip connections or shortcuts to jump over some layers. Skipping helps build deeper network layers without falling into the problem of vanishing gradients. Therefore, using ResNet arcitecture allows us to build deeper neural networks, benefiting more from the power of deep neural network [6]. Using transfer learning allows us to save time, since we only need to train the fully connected layer at the end of ResNet18 and still get a high accuracy.

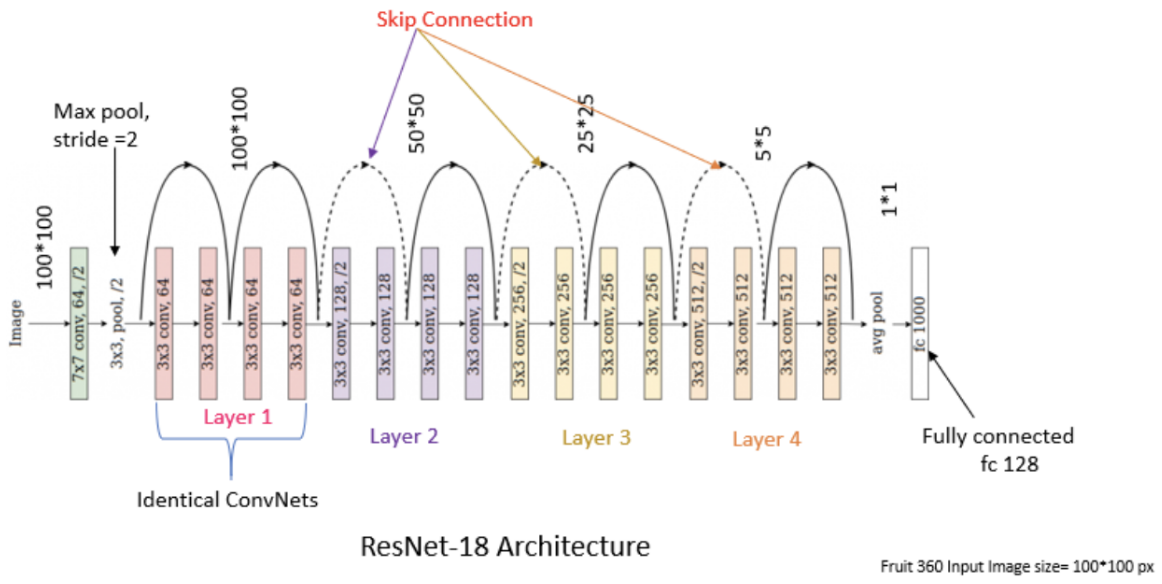


Figure 3: ResNet18 architecture

6 Experiments and Results

In the experiment, the training set is 10800 for Binary Logistic Regression, and 10000 for both Vanilla CNN and ResNet18 Transfer Learning. For CNN and ResNet18, we also have a validation set of size 800. The test set for all three models contains 992 images. We evaluate both train accuracy and test accuracy to show the performance of the models. For Vanilla CNN and ResNet18, the input images after data preprocessing have shape of 32 x 32 x 3 with batch size of 128.

6.1 Binary Logistic Regression

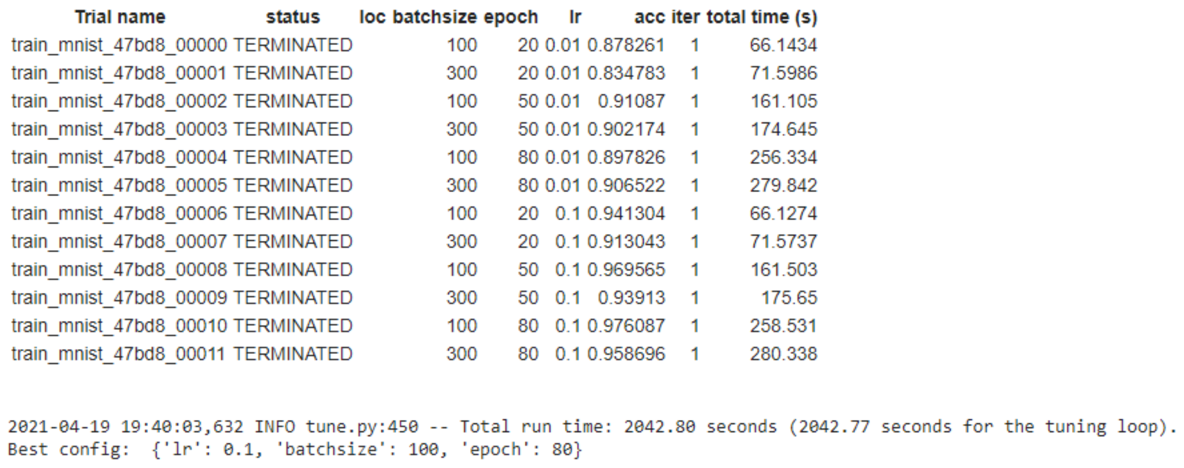
We use the normalized greyscale image to do Logistic Regression with the Logistic Regression method from sklearn package. We apply L2 regularization to the Logistic Regression to avoid overfitting. Since each image is of size 32×32, the feature size of each input image is 1024, and we have 10800 images for training.

Model	Train Accuracy	Test Accuracy
Logistic Regression with greyscale image	0.916	0.877

Table 1: The train and test accuracy of logistic regression models

6.2 Vanilla CNN

In this project, our Vanilla CNN contains 2 convolutional layers, a ReLU operation following each convolutional layer, 2 max-pooling layers with a kernel size of 2, and 3 fully connected layers with 1 hidden layer. There are total of 1152 input parameters and 2 output parameters for fully connected layers. We use stochastic gradient descent (SGD) as the optimization model.
We also perform hyper-parameter tuning on a batch size of 100 and 300, epoch size of 20, 50, and 80, and learning rates of 0.01 and 0.1 using the Ray Tune method. The result of hyper-parameter tuning is shown in figure 3. It is clear to see that Vanilla CNN with a batch size of 100, learning rates of 0.1, and epoch size of 80 has the best performance on the validation set.



```
        Trial name              status   loc batchsize epoch   lr      acc iter total time (s)
train_mnist_47bd8_00000 TERMINATED             100    20 0.01 0.878261    1        66.1434
train_mnist_47bd8_00001 TERMINATED             300    20 0.01 0.834783    1        71.5986
train_mnist_47bd8_00002 TERMINATED             100    50 0.01  0.91087    1        161.105
train_mnist_47bd8_00003 TERMINATED             300    50 0.01 0.902174    1        174.645
train_mnist_47bd8_00004 TERMINATED             100    80 0.01 0.897826    1        256.334
train_mnist_47bd8_00005 TERMINATED             300    80 0.01 0.906522    1        279.842
train_mnist_47bd8_00006 TERMINATED             100    20  0.1 0.941304    1        66.1274
train_mnist_47bd8_00007 TERMINATED             300    20  0.1 0.913043    1        71.5737
train_mnist_47bd8_00008 TERMINATED             100    50  0.1 0.969565    1        161.503
train_mnist_47bd8_00009 TERMINATED             300    50  0.1  0.93913    1         175.65
train_mnist_47bd8_00010 TERMINATED             100    80  0.1 0.976087    1        258.531
train_mnist_47bd8_00011 TERMINATED             300    80  0.1 0.958696    1        280.338


2021-04-19 19:40:03,632 INFO tune.py:450 -- Total run time: 2042.80 seconds (2042.77 seconds for the tuning loop).
Best config:  {'lr': 0.1, 'batchsize': 100, 'epoch': 80}
```

Figure 4: Hyper-parameter Tuning

We train and validate the Vanilla CNN model which has batch size of 100, learning rates of 0.1, and epoch size of 80. The figure 5 shows how training and validation losses vary according to the current epoch. As you can see, the validation loss is slightly higher than then training loss. Both training and validation losses are less than 0.1 at the end of training process. The figure 6 shows how training and validation accuracies vary according to the current epoch. Both accuracies gradually increase as the number of epoch increases.

At the end of training and validating process, both training and validation accuracies are higher than 95 percent. The validation accuracy is not extremely less than training accuracy, which means overfitting is not a huge issue in our Vanilla CNN model.

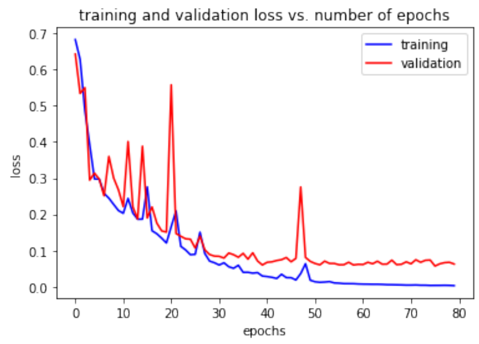


Figure 5: training and validation loss vs. number of epochs

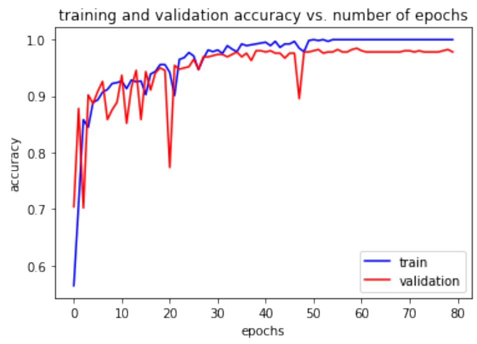


Figure 6: training and validation accuracy vs. number of epochs

Then, we apply our Classic CNN model on the test set and the result is shown in table below. The Classic CNN model can achieve an accurancy of 0.9623.

Model	Train Accuracy	Test Accuracy
Vanilla CNN	1.00	0.9623

Table 2: Test Accuracy of Vanilla CNN Model

6.3 ResNet18 Transfer Learning

As shown in the method part, ResNet18 contains a 7×7 convolutional layer at beginning followed with a max pooling layer. Then, it contains 4 sequence of layers, each layer in the sequence is a convolutional layer with size 3×3. In addition, there exists skip connections every two layers apart. At the end, it has an average pooling layer, followed by a fully connected layer.

We use the ResNet18 from Pytorch.models, which is trained with 2012 ImageNet Classification Dataset[7] from paper Deep Residual Learning for Image Recognition[8]. We freeze all weights in the convolutional layers of the ResNet18 model, and retrain the fully connected layer with our given dataset. We use 10000 images to train the model for 10 epochs, and use validation set of size 800 to check the performance of our model during training. The graphs of loss and training/validation errors corresponding to training epochs is given below in Figure 7 and Figure 8.

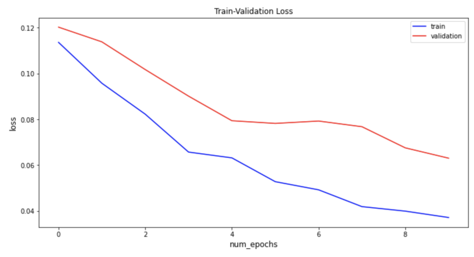


Figure 7: training and validation loss vs. number of epochs

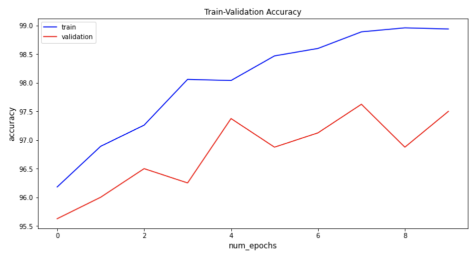


Figure 8: training and validation accuracy vs. number of epochs

During the first 4 epochs, the validation accuracy has achieved 97% accuracy, and it fluctuates in the remaining 6 epochs. Although the training error still increases after 4th epoch, the validation accuracy is still increasing. However, we see that the cross-entropy loss both training and validation sets are decreasing for the 10 epochs. It seems that the error remains flat during epoch 4 to epoch 7, and then begins to decrease after that epoch. Therefore, the accuracy may still increase if we train more than 10 epochs. Eventually, we achieve a training accuracy of 98.94%, and a test accuracy of 97.58%.

Model	Train Accuracy	Test Accuracy
ResNet18 Transfer Learning	0.989	0.976

Table 3: ResNet18 Transfer Learning Result

7 Conclusion and Discussion

In thie project, we explore best Face Mask Detection model using Logistic Regression, Vanilla CNN ,and Transfer Learning with ResNet18. We use the public masked face datasets from the Kaggle competition which is created by Ashish Jangra. During the data preprocessing process, we resize each image into size of 32 x 32 and we split the original dataset into training, validation, and test sets. We implemented three models, including Logistic Regression, Vanilla CNN, and Transfer Learning with ResNet18.
Based on the result, the logistic regression achieves a test accuracy of 0.877. The Vanilla CNN with hyper-parameter tuning achieves a test accuracy of 0.9623. The ResNet18 gets a test accuracy of 0.976. Comparing these three models, using transfer learning with ResNet18 achieves the highest accuracy. We observe that deep learning models perform much better than the logistic regression model in image detection and classification problems. ResNet18 Transfer Learning has a slightly higher test accuracy than Vanilla CNN. In other words, deep learning models can make high accurate predictions and deeper models may perform better than shallow models in face mask image detection problem.
Our ResNet18 Transfer Learning model can be used to confirm if the customer is wearing a mask or not at the entrance of restaurants and other indoor places. Our model can still be improved. Future work may include: 1) Our face mask dataset is not balanced split with respect to gender and race. This would be worthy effort to constructed less biased the model for minority classes. 2)Data augmentation can be used to generate more images and to add some noise, such as scars or tattoos, to images.

Acknowledgments

We want to show great thanks to Dr.Konrad, Dr.Lyle and Rongguang Wang for their kind help and understandings during this project.

References

[1] World Health Organization. Maho coronavirus (covid-19) dashboard. https://covid19.who.int/. Accessed: 2021-04-30.

[2] KATHY KATELLA. 5 things everyone should know about the coronavirus outbreak. https://www.yalemedicine.org/news/2019-novel-coronavirus. Accessed: 2021-04-28.

[3] CDC. Masks protect you & me. https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-sick/masks-protect-you-and-me.html. Accessed: 2021-01-07.

[4] Face mask 12k images dataset. https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset, June 2020. Accessed: 2021-4-29.

[5] Peter McCullagh. *Generalized linear models*. Routledge, 2018.

[6] Gaurav Singhal. Transfer learning with resnet in pytorch. https://www.pluralsight.com/guides/introduction-to-resnet. Accessed: 2021-4-29.

[7] O. Russakovsky, J. Deng, J. Su, H.and Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, and et al. Imagenet large scale visual recognition challenge. 2014.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. 2015.