Comparative Study of Visual Face Mask Detection Using Different Models under COVID-19

Yutong Zhang
yzhang2433@wisc.edu

Bob Dai
zdai38@wisc.edu

Jacky Chen
chen878@wisc.edu

Abstract

Under the circumstance of COVID-19, wearing a mask is proved to be an efficient method to protect people from getting infection. The purpose of this paper is to find an appropriate face mask detection model that could contribute to human society. In this paper, five models will be discussed, Multi-layer Perceptrons(MLP), AlexNet, ResNet-34, VGG-16 and transfer learning of VGG-16. There will be 10000 pictures used for the training process of each model and 800 for validation and 992 for test. Pictures with and without masks contribute similar percentage to each set. After experimenting with different model architecture, we found that MLP, AlexNet, ResNet-34 and VGG-16 showed a rapid increase of accuracy rate from the first ten epochs and they could reach a test accuracy of higher than 98%. The transfer learning model performed high accuracy rate which is higher than 98.5% since first epoch and the final test accuracy reached higher than 99%. The results showed that the overall performance of five models for detecting face masks were similar according to test accuracy. In this case, MLP could be a better choice as it needed the shortest time.

1. Introduction

From 2019 to the present, COVID-19 has led to great crises all over the world, causing the deaths of nearly 2.5 million people. Since then, "how to prevent infection with COVID-19" has also become a focus of daily concern. Centers for Disease Control and Prevention (CDC) tells us that COVID-19 is mainly spread from person to person through respiratory droplets, when we communicate with others, the respiratory droplets will enter people's mouth and nose through the air, thereby infecting the virus. [2] Therefore, wearing face masks plays a vital role in slowing down the spread of COVID-19. Masks help us prevent the spread of the virus between the person wearing the mask and those around them. Under this inspiration, our team decided to do a face mask detection project to identify people who do not wear masks to help reduce the spread of COVID-19.

Our goal is to use the deep learning knowledge we have learned to train and compare a multi-layer perception and 4 different convolutional neural network models through the 12000 pictures of people wearing and not wearing masks provided in the Kaggle datasets, which can identify whether a person is wearing masks. This seems to be a very challenging task. The accuracy of the model may be affected by the difference in face coverage due to the way people wear masks and the diversity of masks (colors, types, etc.), but we will try our best to achieve an accuracy of over 90%.

We are committed to creating a face mask detection model that could contribute to human society. Currently, many state governments of the United State require people to wear face masks in public, but there are still many people who do not follow the rules and enter crowded places like supermarkets without wearing masks.[9] Face mask detection is necessary for such areas. Since manual detection is both time-consuming and labor-intensive, face mask detection technology by computer is what we need. We could finally promote our best model to a comprehensive artificial intelligence device to detect the face mask wearing in public space.

2. Related Work

With the increasing development of deep learning, facial recognition technology has been applied to all walks of life. A study by Nieto-Rodríguez and Mucientes[9] introduces a system that detects whether medical staff wears medical masks in the operating room. This system can only send out an alarm to medical staff who do not wear medical masks. While there are various colors, shapes, and categories of masks, there are also many precedents where use different technologies or both deep learning together to solve this problem. Loey et al[7] used a combination of deep learning and machine learning to label and locate medical mask objects in the image, they first used ResNet 50 for feature extraction, and then used decision trees, support vector machines (SVM) and integrated algorithms to design the classification process of masks. This paper[5] proposes masked and unmasked face recognition methods based on a statistical method suitable for unmasked face recognition and masked face recognition technology, principal component analysis (PCA), to analyze faces with different occlusions or masks. In our project, we will focus on using deep learning models and compare the testing accuracy of different deep models to select the best model.

3. Proposed Method

3.1. Multi-layer Perceptrons (MLP)

We start with Multi-layer Perceptron, which is considered as the benchmark for our project. The architecture it introduced has paved the way for further advanced neural networks. MLP belongs to the class of feed-forward neural networks, which means the data is transmitted from the input layer to the output layer in the forward direction, and consists of three types of layers: the input layer, output layer, and hidden layer, as shown in figure 1.

$$z = \sum_{i=1}^{n} w_i x_i + b = \mathbf{x}^T\mathbf{w} + \mathbf{b}$$

The connections between the layers are called weights. We use sigmoid as the activation function in the process of forward propagation.

$$Sigmoid(z) = \frac{1}{1+e^{-z}}$$

The neurons in the MLP are trained with the back propagation algorithm. Back propagation is a technique used to update the weights and bias using the output as inputs to minimize the error of an MLP. We choose to use cross entropy to compute the loss and stochastic gradient descent (SGD) to optimize the loss[3].

The Feature Normalization is also an essential part in our model. We apply batch normalization, which is conducted in two step: normalize net input and pre-activation scaling. It helps normalizing the hidden layer input, voiding the exploding/vanishing gradient problems, and increasing the training stability and convergence rate.

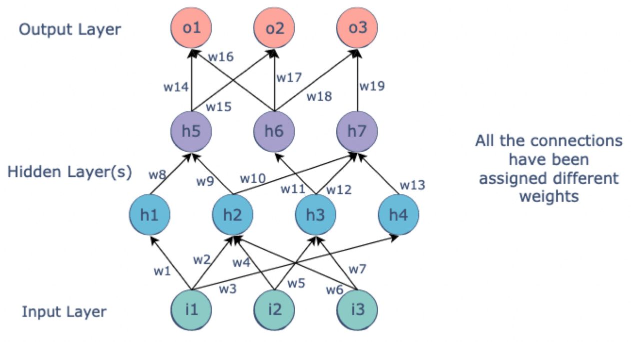


Figure 1. Image source: `https://www.educative.io/edpresso/what-is-a-multi-layered-perceptron`

3.2. Convolutional neural network (CNN)

Then, we choose to experiment with Convolutional Neural Networks, which is the most widely used architecture for image detection and image classification. In the following section, we will discuss the AlexNet, VGG-16, ResNet-34 and transfer learning.

3.2.1 AlexNet

AlexNet is the main breakthrough for convolutional neural networks back in 2012. The overall architecture contains eight layers with weights. The first five are convolutional layers and remaining three are fully connected layers. There are three max pooling layers that summarize the outputs of neightboring groups of neurons in the same kernel map. The ReLU activation function is applied in each hidden layers and softmax is applied in the ouput layer[4].

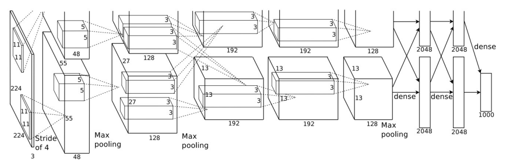


Figure 2. An illustration of AlexNet's architecture.

3.2.2 VGG-16

VGG-16 is a relatively simple convolution neural net (CNN) architecture without too many hyper-parameters. The number 16 represents that there will be 16 layers with weights. This architecture was proposed by Karen Simonyan and Andrew Zisserman when they used to win

ILSVR(Imagenet) competition in 2014[10]. VGG-16 has 3x3 convolution layers and 1 stride with same padding and 2x2 maxpool layer through the whole architecture. In VGG-16, ReLU is activation function for each hidden layer. It also uses the softmax loss function for output layer to do classification. The overall architecture is shown in Figure 3.

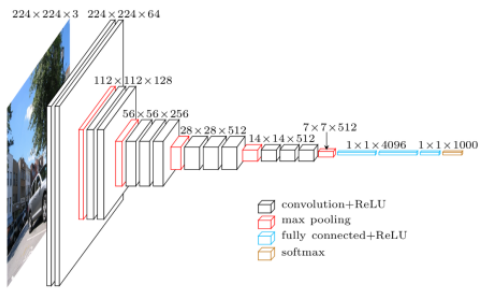


Figure 3. VGG-16 Structure Image source: `https://www.cs.toronto.edu/~frossard/post/vgg16/`.

3.2.3 ResNet-34

ResNet is currently the most widely used CNN feature extraction network. In deep learning, blindly increasing the number of layers after the deep CNN network reaches a certain depth will not bring further improvement in classification performance, but will cause the network to converge more slowly, and the classification accuracy of the test dataset will also become worse. However, ResNet solves the problem of the vanishing gradients in very deep neural network training, which achieves this by using skip connections or shortcuts to skip certain layers. He, Zhang, Ren, and Sun (2016) proposed a new deep residual network, which defines a building block:

$$y = F(x, \{Wi\}) + x$$

where x and y are the input and output vectors of the layer. The core part of the network is in Figure 4.

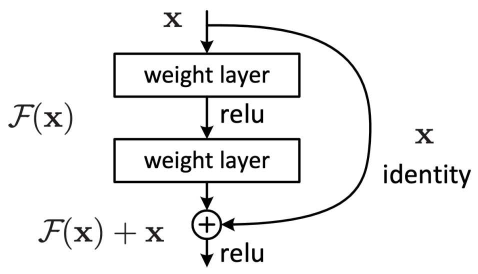
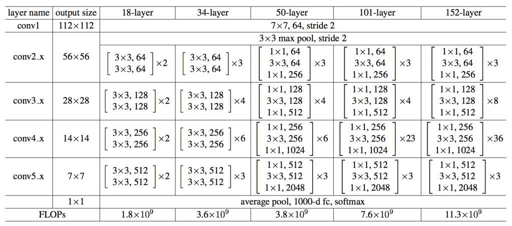


Figure 4. Residual learning: a building block. Image source: [6].

The input x is superimposed on the weights according to the conventional neural network and then passed through the activation function. After the secondary weights are superimposed, the input signal and the output at this time are superimposed, and then pass the activation function.

He, Zhang, Ren, and Sun (2016) explained that shortcut connections neither introduce external parameters nor increase computational complexity, which can compare networks with the same number of parameters, depth, width, and computational cost at the same time[6]. ResNet has many different sizes, in our project, we chose to use ResNet-34 to train our model because it has simple 34-layer residual neural network structures and at the same time has sufficient accuracy. The architecture of ResNet-34 is shown in Figure 5.



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 5. Architectures for ResNet. Image source: [6].

3.2.4 Transfer Learning

Since our data set is relatively small, we choose to use transfer learning to pre-train the model on a larger object detection data set, and then make adjustments on our data set to speed up training and improve the performance of the model. Transfer learning refers to reusing the weights of one or more layers of a pre-trained network model in a new model, and then adjusting the weights when training the model.

3.3. Activation Function

These CNN architectures use the softmax function for output layer to do classification.

$$Softmax(z_t^i) = P(y = t|z_t^i)\frac{e_{ti}^{z}}{\sum_{j=1}^{k} e^{z_j^i}}$$

where $t \in \{j...k\}$ and k is number of class labels.

And ReLU is commonly used activation function for each hidden layer.

$$ReLU(z) = \begin{cases} z & if\ z > 0 \\ 0 & otherwise \end{cases}$$

3.4. Minibatch Mode

In our models, Minibatch mode is used to avoid too much noisy compared to "on-line" mode and too little updates compared to "batch" mode. In Minibatch mode, training dataset will be divided into batches according to batch size. Gradients and costs will be calculated and weights and bias will be updated after each batch followed by:

$$\bigtriangledown_{\mathrm{w}}\mathcal{L} = (y^{[i]} - \hat{y^{[i]}})x^{[i]}$$

$$\bigtriangledown_{b}\mathcal{L} = (y^{[i]} - \hat{y}^{[i]})$$

$$\mathrm{w} = \mathrm{w} + \eta \times (- \bigtriangledown_{\mathrm{w}} \mathcal{L})$$

$$b = b + \eta \times (- \bigtriangledown_{b} \mathcal{L})$$

where η represents learning rate and negative gradient is used.

3.5. Regularization and Optimization

Dropout Dropout is used for our models with a dropout rate equals 0.5 which means 50% of units will be dropped randomly for each model. With dropout, model for each minibatch is different through which we could reduce overfitting and average over weights for each unit.

Learning Rate Decay Learning rate decay is used for our models in order to dampen oscillations near the end of the training.

Momentum Momentum is used to make convergence faster by dampening oscillations using "velocity". It will accelerate the training in the direction of the gradient which could also help to escape local minima traps.

4. Experiments

4.1. Dataset

Our project will use Face Mask dataset on Kaggle[1] which contains 11792 images with two classes, with or without mask. In this image set, we have 5000 pictures with mask and 5000 pictures without mask for training set, 400 and 400 for validation as well as 483 and 509 for test. These images contain people's whole faces with or without mask, cartoon faces with or without mask and parts of people's face such as foreheads. People in different races, different sexes, different ages and with different emotions are included in this image set. Examples of image set are presented in Figure 6.

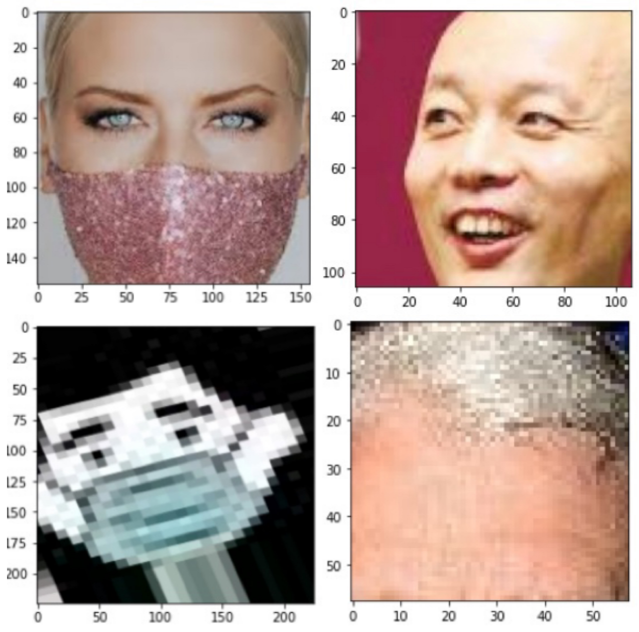


Figure 6. Image examples: with and without mask

All images are RGB images and we create our own data loader to resize them into a 70 $\times$ 70 pixels images. They are cropped randomly into 64 $\times$ 64 pixels and normalized to the [0,1] range in data preparation process. The random crop is a data augmentation technique wherein we create a random subset of an original image. This helps our model generalize better because the object(s) of interest we want our models to learn are not always wholly visible in the image or the same scale in our training data[8]. These images in training, test, and validation dataset are shuffled.

4.2. Model Implementation

4.2.1 MLP Model

We designed a MLP model with two hidden layers to avoid the varnishing and exploding of the gradient. Since all images are RGB images, the number of feature as the input for the model is 3 $\times$ 64 $\times$ 64. The second hidden layer takes the output of the first layer with 100 input features and produces the 50 output features. Then the model applies the linear transformation taking 50 input features and producing 2 output features. For each layer the batch normalization, ReLU activation function, and dropout=0.5 is applied, since we found the result improved compared without applying these techniques. The model is trained using stochastic gradient descent (SGD) with learning rate = 0.1. In addition, we used learning rate schedulers in PyTorch to decay the learning rate during the training process, which greatly reduce the training time and improve the stability. After the experiment, we find the batch size = 256 is a reasonable size in our models. Finally, we trained the model for 50 epochs.

4.2.2 CNN Models

AlexNet The AlexNet model has five convolutional layers and three fully connected layers. We used ReLU in every layers and used max-pooling in each convolutional layers. We set the batch size to 256. After the convolutional layers, we used average-pooling over the input signal and it produces output of size 6×6. In the following fully connected layers, we dropped out 50 percent nodes to prevent overfitting. In the last layer, we have 4096 features output and use linear transformation converting it into number of class, which is two.

VGG-16 We used standard VGG-16 model and set the number of classes = 2. For the hyper-parameters, after doing many experiments, we set the momentum = 0.9 and learning rate = 0.01 using SGD optimizer. We set batch size = 256. We also applied learning rate scheduler with factor = 0.1 to decay the learning rate during training.

ResNet-34 Our ResNet 34 architecture uses 7×7 and 3×3 kernel sizes to perform initial convolution and max pooling, respectively. After that, the first stage of the network starts with 3 residual blocks, and each block contains 2 layers. For deeper network, Bottleneck places the stride for downsampling at 3x3 convolution while original implementation places the stride at the first 1x1 convolution. Finally we add an average pooling layer. For the hyper-parameters, after doing many experiments, we set the momentum = 0.9 and learning rate = 0.01, and use SGD with a mini-batch size of 256.

Transfer learning VGG-16 For transfer learning using VGG-16, the first step is to freeze the model. We want to use VGG-16 to fine-tune the last 3 layers, so we set all the classifier's auto gradient except last 3 layers to false. Then, for the last layers, we replace the output layer with our own output layer = 2 because the number of class labels differs compared to ImageNet. Then, we train the VGG-16 as usual.

4.3. Software

We perform all experiments on Python through Jupyter Notebook, which provides various tools and libraries and nice visual images. In order to make our experiment reproducable, we provide the reference for the version of our software, including: Python implementation: CPython, Python version: 3.8.5, IPython version: 7.19.0, torch: 1.8.1

4.4. Hardware

For computer hardware, we perform most experiments on Jacky's Alienware laptop with an i7-6700HQ CPU and NVIDIA GTX 980M GPU.

5. Results and Discussion

5.1. Results

After training our different models, we found that AlexNet with 2×2 max-pooling and 0.5 dropout has the highest test accuracy, up to 99.7%. The test accuracy of ResNet-34, VGG-16 and VGG-16 using transfer learning are also very high, all of which are above 99%, respectively 99.40%, 99.50% and 99.29%. In contrast, Multi-layer Perceptrons (MLP) has the lowest test accuracy of all models, which is 98.68%

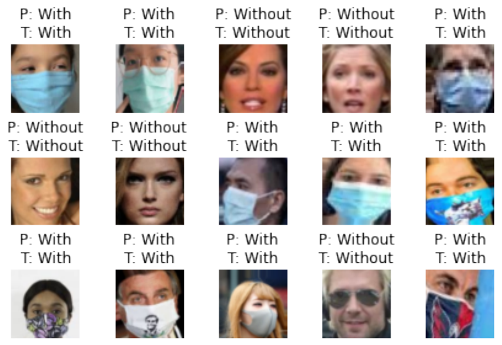
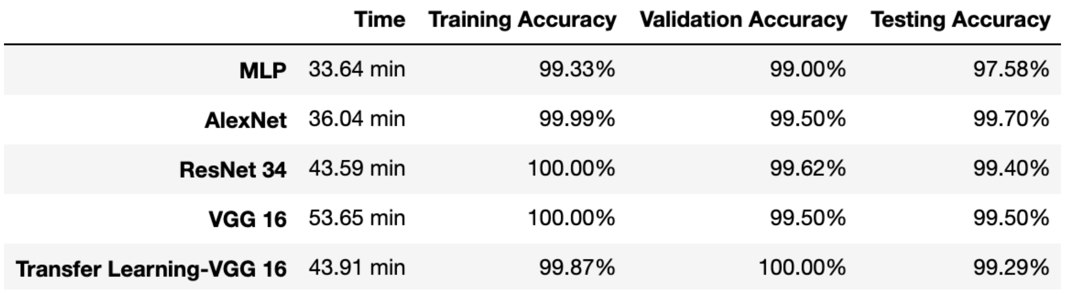


Figure 7. Model Training Results: examples

In terms of training time, although MLP has the lowest accuracy rate, it only takes 33.64 minutes to train, which is very efficient. AlexNet's training time is relatively short, taking 36.04 minutes. Besides, ResNet-34 and VGG-16 both took a longer time, which are 43.59 minutes and 53.65 minutes respectively. Interestingly, transfer learning will greatly shorten the training time, after we apply transfer learning to VGG-16, the training time is reduced to 43.91 minutes. The results are in Figure 8.



	Time	Training Accuracy	Validation Accuracy	Testing Accuracy
MLP	33.64 min	99.33%	99.00%	97.58%
AlexNet	36.04 min	99.99%	99.50%	99.70%
ResNet 34	43.59 min	100.00%	99.62%	99.40%
VGG 16	53.65 min	100.00%	99.50%	99.50%
Transfer Learning-VGG 16	43.91 min	99.87%	100.00%	99.29%

Figure 8. Model Training Results

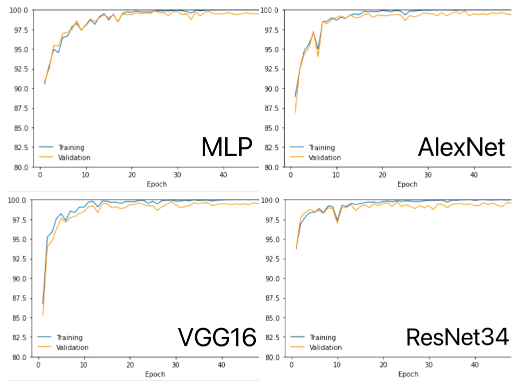


Figure 9. Model Training Results: Accuracy Rates

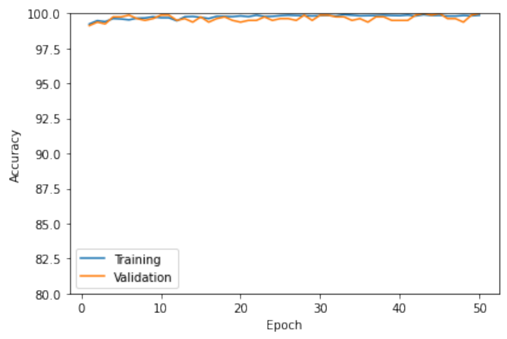


Figure 10. Model Training Results: Transfer Learning Accuracy Rates

5.2. Discussion

The results generated from five models did not show much difference for test accuracy. The lowest accuracy rate was 97.58% from MLP model. As our MLP model consists only two hidden-layer, its learning ability is more limited than CNN architectures. And due to its simpler structure, MLP shows a faster training process than other models. MLP also shows a lower learning process as it achieves stable high validation accuracy rate after completing more than 20 epochs. The test accuracy for all CNN models is above 99% which indicated that the face mask detection task with only two classes is not complicated enough to distinguish those architectures. Alexnet is the simplest CNN model we chose with eight layers and it trained within shortest time. We got a highest test accuracy from Alexnet which is 99.7%. VGG-16 is also a simple architecture but it has too large number of parameters, which gives it longest training time to progress which is 53.65 minutes, 10 minutes more than second longest training time. ResNet-34 provides us an opportunity to implement deeper architectures with 34 layers and it has higher efficiency by allowing skipping useful layers. These features gave this model a high accuracy rate and an acceptable number of minutes to progress. The learning processes were faster for the above three architects. Their validation accuracy rate became stable after around 10 epochs of training. Transfer learning is used because we did not have sufficient images inputs. With a pre-training model trained from a larger dataset, the training process for our own imageset became model modification in essence. Therefore, we got a validation accuracy rate of 99.12% from the first epoch and it remained high for the rest of 49 epochs.

Before our experiment, we resized and cropped randomly our images input and we used dropout during all model training process. Therefore, our models seemed fine without much over-fitting.

6. Conclusions

In this project, we trained various architecture models for helping face mask detection. However, this task was not difficult enough to distinguish five models we chose. After getting all results, we found it hard to select the most efficient one by comparing test accuracy. As all five models got their test accuracy of above 97%, we may conclude that each model could be an option for the detection. Although CNN architectures such as VGG-16, ResNet34 and transfer learning are thought to have better performance during more complicated deep learning, MLP and Alexnet performed better for this specific task. With 33.64 minutes of training time and 97.58% of test accuracy, MLP is qualified to be counted as one of the best options. For Alexnet, the training time is 36.04 minutes and the test accuracy rate is 99.7%. It is also a best choice to get 2% higher accuracy rate with a sacrifice of 2.4 minutes of training time.

7. Acknowledgements

First, we would like to express my deepest gratitude to our professor Sebastian Raschka, who taught us STAT 453: Deep Learning. In his courses, we learned various tools and techniques to help us explore the world of deep learning. We truly appreciate his deep learning code resources, which allow us conduct experiment conveniently. Second, we would also thank to Kaggle community, which provides our this interesting dataset.

8. Contributions

Jacky Chen found the original dataset, Face Mask 12K Images Dataset, in Kaggle.

For the computational side of the project, Bob and Jacky were responsible for the data loader and preprocessing the images dataset and building the deep learning model. Yutong was mainly responsible for the training part and developing an evaluation model.

For the writing task for the project, Yutong was responsible for Introduction, Related Work and Result; Bob handled for Dataset, Model Implementation and Acknowledgements; Jacky was responsible for Abstract, Discussion and Conclusion. The remaining parts were the result of the joint efforts of the three of us.

References

[1] Face mask 12k images dataset.

[2] Guidance for wearing masks.

[3] What is a multi-layered perceptron?

[4] G. E. H. Alex Krizhevsky, Ilya Sutskever. Imagenet classification with deep convolutional neural networks. *Information*, page 4, 2012.

[5] M. S. Ejaz, M. R. Islam, M. Sifatullah, and A. Sarker. Implementation of principal component analysis on masked and non-masked face recognition. In *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, pages 1–5, 2019.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

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

[8] J. Nelson. Why and how to implement random crop data augmentation.

[9] A. Nieto-Rodríguez, M. Mucientes, and V. M. Brea. System for medical mask detection in the operating room through facial attributes. In R. Paredes, J. S. Cardoso, and X. M. Pardo, editors, *Pattern Recognition and Image Analysis*, pages 138–145, Cham, 2015. Springer International Publishing.

[10] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv*, 1409:1556, 2014.