Performance Analysis of Three kinds of Neural Networks in the Classification of Mask Images

Jinkui Chi\textsuperscript{1*}, Yuxi Liu\textsuperscript{2†}, Victor Wang\textsuperscript{3†} and Jiaxuan Yan\textsuperscript{4†}

\textsuperscript{1}College of Letter and Science College of Agricultural and Environmental Sciences, University of California Davis, Davis, California, 95616, the U.S.
\textsuperscript{2}Department of Computer Science, George Mason University, Fairfax, Virginia, 22030, the U.S.
\textsuperscript{3}Department of Mathematics, University of California, San Diego, La Jolla, California, 92093, the U.S.
\textsuperscript{4}College of Information Science and Technology, Beijing University of Chemical Technology, Beijing, Beijing, 100029, China

\*Corresponding author’s e-mail: jkchi@ucdavis.edu, yliu57@gmu.edu, vwang@ucsd.edu, 2018170010@mail.buct.edu.cn

†These authors contributed equally

Abstract. The global spread of COVID-19 has led to a massive increase in public demand for effective measures to slow the spread of the virus. In response, the World Health Organization has advised people to wear face masks every day to stop the spread of the virus. In facial mask recognition, traditional manual feature extraction methods are cumbersome and inaccurate, considering the various types of masks and other practical problems. This paper considers various neural network models comprehensively and selects three classical CNN networks, including VGG, ResNet, and DenseNet. After using the data expansion and optimization methods, good data results are obtained for the three network models. Remarkably, the improved DenseNet accuracy can reach 99.55%.

1. INTRODUCTION

The global spread of coronavirus disease 2019 (COVID-19) has resulted in a mass increase in public demand for effective means to slow the spread of the virus. Fortunately, Howard et al. found that using facial masks to stop the spread of COVID-19 is effective when compliance among the populace is high [1]. The World Health Organization (WHO) has also recommended using face masks as part of a package of health measures to stop the spread of COVID-19 [2]. In order to ensure the efficiency and safety of mask wearing detection, it is necessary to develop an automated high-performance detection machine. Deep learning provides such support, combined with knowledge of computer vision, to better perform automated detection tasks.

Since the introduction of AlexNet in 2012, deep convolutional neural networks have become the dominant approach for image classification [3]. The basic structure of a neural network with convolution layers, pool layers, connection layers, and output layers has achieved great strides in image classification, object detection, attitude estimation, and image segmentation [4].

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
The VGG-19 model was first proposed in 2014 by Simonyan et al. to investigate the effect of Convolutional Network (ConvNet) depth on image recognition performance [5]. The VGG network was trained on the ImageNet database with 1.3 million images and 1000 classes. VGG-19 is a modification of the VGG network that uses 19 layers to achieve performance comparable with other state-of-the-art models. The model uses a sequence of convolutional layers, max-pooling layers, and fully connected layers with classification handled by a SoftMax activation function.

The concept of the residual network (ResNet) was first proposed in He et al. [6] and extended in He et al. [7]. They followed a principled approach to add shortcut connections every two layers to a VGG-style network (Simonyan and Zisserman) [5]. The new network has better performance and achieves both lower training and test errors. ResNet has gradually matured after several years of development and can be used in many life scenarios.

The introduction of ResNet started the era of creating more profound and more connected neural networks. DenseNet inspired by ResNet is one of this kind. The main innovation of this neural network is to have a closer connection between these input layers and output layers. This innovation allows the network to give more accurate results with more minor features generated.

Our work focuses on analyzing the differences in performance of a facial mask detector when utilizing VGG-19, ResNet-34, and DenseNet-121 to serve as the backbone. At the same time, this paper will not be limited to the model parameters obtained by transfer learning. However, it has achieved good simulation results. He et al. once proposed training procedure refinements, including changes in loss functions, data preprocessing, and optimization methods also played a significant role [6]. This paper draws on those findings, using methods such as modifying the stride size of a particular convolution layer or adjusting the learning rate schedule to improve the model’s accuracy.

The rest of the paper is divided into three parts and is organized as follows. Section 2 introduces the selected neural networks as well as the introduction of experimental settings. Section 3 shows the experimental results and the analysis of our findings. Finally, Section 4 concludes this paper and discusses opportunities for future experiments.

2. METHODS

We try various neural networks to train and finally decide to utilize VGG, ResNet, and DenseNet, three neural networks, to develop image recognition on training data. The following is an introduction to the specific methods of three neural networks.

2.1. VGG

The VGG-19 network, as initially proposed, takes a 224x224 RGB image as input [5]. VGG-19 uses minimal filter size of 3 x 3 for its convolutional layers and a 2 x 2-pixel window with stride 2 for its max-pooling layers. The stacking of these tiny filters allows for multiple convolution layers to incorporate more significant non-linearity in the decision function and implement regularization by reducing parameters from a single large layer to multiple stacked layers. The VGG-19 model consists of 16 convolution layers, five max-pooling layers, three fully connected layers, and one output layer. Two of the fully connected layers used a ReLu activation with 4096 channels. The last layers had 1000 channels to handle the ImageNet classification. The final output layer uses a softmax activation.

Although VGG-19 is not as deep as models like ResNet-34, VGG-19 has still been proven to have comparable results in image recognition tasks while also requiring less time and resources to train than more complex models [8].

The base VGG-19 model we trained had no alterations on either the data or the model and used a batch size of 32, epochs of 45, and a learning rate of 0.01. Note that in all experiments, the VGG-19 architecture layers were kept the same. Data augmentation was done on the dataset by flipping images horizontally and vertically, randomly shearing the images, and randomly zooming in. Our base VGG-19 model used an 80-10-10 split of the data into training, validation, and test sets. For our experiments, we introduced a 70-15-15 split to observe the effects on performance and overfitting. Hyperparameter tuning was performed using random search. The hyperparameters tuned were units in the fully
connected layers, activation function in the fully connected layers, learning rate, epochs, and batch size. Dropout and early stopping were introduced into the model to see if overfitting decreased.

2.2. ResNet
The ResNet model accepts a 3 x 224 x 224 (Channel, Height, Width) image and outputs a list of numbers representing the probability. It contains five blocks to deal with image of different sizes. In the first block, the image of 3 x 224 x 224 will go through a convolutional layer whose size is halved, a batch normalization layer, a real layer, and be max pooled with three as the convolution kernel size. After the first block, a 64 x 112 x 112 image will be output. Then this image will pass the second block three times. The second block includes two convolutional layers with three as the kernel size. The image will be processed by the third, fourth, and fifth blocks. The composition of these three blocks is very similar. They all contain two convolution layers with three as the kernel size. The first convolution layer will set the stride to 2 to half the size of the input image. The image will pass the third, fourth, and fifth blocks four times, six times, and three times respectively. Another important feature of Resnet is that there is a direct skip path. Suppose the convolutional layer in the block does not affect the final probability calculation result. In that case, the incoming image can skip the block and be transmitted directly. Generally speaking, the input image is in size of 224 x 224 at first, then becomes 112 x 112, 56 x 56, 28 x 28, 14 x 14, 7 x 7, and 1 x 1 in size finally. The final image will also pass through the average pool, 1000-d fully connected, and softmax layer.

We employ the normalization method to optimize the images. This can ensure that all images have similar distribution and are more accessible to convergence, which helps training faster and better.

2.3. DenseNet
The dense net, mainly influenced by ResNet, is invented by adding the dense block to the net to improve its performance. However, the architecture approach is not only creating a deeper network but also reusing parameters formed in the earlier layers [10].

The default input size of it should be 224*224. Before entering the first dense block, an input graph will pass through a 7*7 convolution layer with stride2, and the picture is downsize to 112*112. Since the input of the preceding layers is not only convolution results but also the parameters generated from the previous layers, we need to control the number of parameters generated by each layer. Therefore, a hyperparameter k is introduced to decrease the number of parameters to k in each layer. The 3*3 max pool generates the local maximum in the next pooling layer, and the picture is further squeezed to 56*56[9].

Aside from it, these features generated from the previous two layers are stacked by concatenation rather than adding. It is the way of the passing parameters of the preceding layers. Then the result goes through the first dense block. Every dense block has a 1*1 convolution layer as a bottleneck to reduce the number of parameters generated and a pooling layer. After that, these results will enter a transition layer. The 1*1 convolution layer will conduct a nonlinear transformation based on ReLu. Then the result will pass through the average pooling layer. This will be the pattern for this neural network. After the fourth dense block, the result will be globally pooled, and the features generated will be ready to be used [9].

3. EXPERIMENTAL ANALYSIS
We respectively considered the results of three methods and performed hyperparameter tuning to improve the results further.

3.1. VGG
The base VGG-19 model without any tuning or data augmentation has a training loss of 0.0330, training accuracy of 0.9906, validation loss of 0.3991, and 0.9375. We will use these accuracy and loss measurements to benchmark our changes against and see if the model improves.
Table 1 shows the results of our experiments with data augmentation and 70-15-15 split. We observed that with data augmentation, the validation loss and training accuracy decrease during the validation accuracy and training loss increase. This suggests that data augmentation may help improve the performance of the model while also reducing overfitting. The results from the 70-15-15 split and a result from combining data augmentation with the 70-15-15 split are less promising. Validation accuracy decreased and validation loss increased for both experiments, suggesting that these two options worsened model performance. In addition, the data augmentation and 70-15-15 split increased overfitting in the model. Therefore, we used data augmentation with an 80-10-10 split for our following experiments. To tune the hyperparameters, we performed a random search with 50 trials to maximize the validation accuracy. For the fully connected layers, we selected neuron counts between 100 and 4100 with a step of 200. The activation function choices were either a ReLu or SoftMax.

Table 2 summarizes the result of using dropout, using early stopping, and using both on the model accuracy and loss. With early stopping only, the training stopped on epoch 16, with the best result appearing at epoch 14. With both dropout and early stopping, the training stopped on epoch 29, with the best result appearing at epoch 24. Our findings suggest that both dropout and early stopping can help reduce overfitting. Therefore, the final VGG-19 uses dropouts after each fully connected layer as well as early stopping. The final model performance is summarized in Table 3.

Table 3: The final model performance of VGG

| Operation       | Train Loss | Train Accuracy | Validation Loss | Validation Accuracy |
|-----------------|------------|----------------|-----------------|---------------------|
| Base            | 0.3484     | 0.8463         | 0.2098          | 0.9375              |
| Final           | 0.3036     | 0.8969         | 0.1832          | 0.9688              |
The improvement in both validation and training accuracy with a corresponding decrease in loss suggests that our model performance has improved. We can also see from a graphical analysis model loss and training graphs in Figure 1 and Figure 2 respectively that there does not seem to exist a significant drop in validation accuracy nor an increase in validation loss that would indicate overfitting. However, we found that the test accuracy for the final model is 0.9487 while the test loss is 0.2078. The addition of dropout and early stopping has reduced the potential for overfitting. However, the difference between the validation accuracy and test accuracy suggests that there may still be some overfitting in the final model.

3.2. Evaluation of ResNet

We use the 80-10-10 split for the train, validation, and test dataset, respectively. The setting of the parameter of learning rate is 0.01, and the learning batch size is 16. In the parameter tuning below, we use an algorithm, “train accuracy + validation accuracy - train loss - validation loss,” to choose the best performance epoch and use the model trained by this epoch to test on the test dataset. This algorithm includes the most critical four result values in hyperparameter tuning. It is helpful to prevent choosing overfitting epoch as our best performance epoch. After training, validation, and final testing without parameter tuning, we get the result in Table 4.
TABLE IV.

| Metric     | Result |        |
|------------|--------|--------|
|            | train  | validation | Base test |
| Loss       | 0.3233 | 0.2631   | 0.2389    |
| Accuracy   | 0.8803 | 0.8989   | 0.9030    |

The base ResNet model without any tuning is shown in Table 4. From the graph, we can see that the test accuracy only reaches 90.3%. It means that the model needs to conduct hyperparameter tuning to improve its performance on the final test. We try many ways to do it.

First, we try to choose the best epoch, increasing the epoch from 10 to 100 to find the best epoch in this range. Only ten epochs used in the base situation may not be enough for model training, so we try to do more epochs training. The train and validation results as shown in Figure 3 and 4.

We use our algorithm, “train accuracy + validation accuracy - train loss - validation loss,” to select the model of the best performance epoch, the 98th epoch, to test on the test dataset and get the results as shown in Table 5.
Compared to the base result, the validation accuracy increases 8.44% and reaches 98.33%. It shows that after increasing epochs, the performance of the neural network increases a lot.

To tune the hyperparameters more, we start to try some different learning rates. We choose four learning rates: 0.1, 0.01, 0.001, 0.0001 and run 10 epochs respectively. We choose the best epoch’s model in every learning rate of tuning to test on the test dataset and present the result data in Table 6.

From Table 6, the effect on the test accuracy and validation accuracy from different choices of learning rate is clear to see. When the learning rate is 0.1 and becomes smaller one-tenth each time, the train and validation accuracy increase simultaneously until the learning rate reaches 0.0001. After that, the train and validation accuracy drop gradually if the learning rate continues to decrease. The best learning rate to choose should be 0.0001, which obtains the highest train accuracy of 91.53% and validation accuracy of 97.11%.

Then, we want to find some improvement on the model performance from batch size’s tuning. We train the data in batch sizes 4, 16, 64, 128 to find the best batch size and let it fit our ResNet model. The learning rates are all set to be 0.0001 because it is the best choice in the learning rates’ tuning. The tuning epochs are 10 for every batch size. We choose the best epoch in every batch size’s tuning, and the tuning result is shown in Table 7.

Table 7 summarizes the result of using different batch sizes on the model accuracy and loss. Surprisingly, different choices on the batch size lead to similar results on train accuracy and validation accuracy. The difference in validation accuracy between any two groups is less than 0.5%. We also try
to train these four batch sizes in bigger epochs, 50 epochs, to see whether it may affect the result greater. However, the result of 50 epochs training is similar to our 10 epochs’ result. Different batch sizes only affect the outcomes slightly.

**TABLE VIII. FINAL RESULTS**

| Batch size | Final Results | Final Results |
|------------|---------------|---------------|
|            | Train Loss    | Train Accuracy| Validation Loss | Validation Accuracy | Final Test Loss | Final Test Accuracy |
| 4          | 0.0660        | 0.9748        | 0.0350          | 0.9933              | 0.0380          | 0.9889             |
| 16         | 0.0704        | 0.9737        | 0.0256          | 0.9956              | 0.0151          | 0.9955             |
| 64         | 0.0638        | 0.9758        | 0.0317          | 0.9956              | 0.0202          | 0.9933             |
| 128        | 0.0701        | 0.9747        | 0.0376          | 0.9956              | 0.0241          | 0.9900             |

After the trial of the combinations of 100 epochs, 0.0001 learning rate, and four different batch sizes, we get our best model after parameter tuning. Table 8 shows that when the batch size is 16, the final test accuracy can achieve best to 99.55%, and compare with our base test accuracy in Table 5, it improves more than 9%, from 90.33% to 99.55%. This is the best performance and best model we can get after hyperparameter tuning.

### 3.3. Evaluation of Dense Net

During the experiment, though a higher resolution of input may increase the quality of our model, we used 64*64 as our input due to the limitation of our devices. In order to further improve the performance of this model, we tested several possible combinations of hyperparameters (learning rate and batch size). The learning rate audits the step size of our model, and batch size controls the number of samples passed to our model. We included the learning rate 0.001 0.001 0.01 0.1 and batch size 2 32 64 128, which offers us 16 possible combinations. We use validation accuracy and overall loss as our indicators to visualize the results, as shown in Figure 5.

![Figure 5. Validation accuracy(upper)step loss(lower)/step epoch Batchsize/learning rate](image-url)
As shown in Figure 6, with limited batch size and a relatively large learning rate, the model will change drastically and not generate a convincing result. On the other hand, with a larger batch size and a low learning rate, the model will not generate features in later interactions efficiently, but the accuracy with a batch size of 128 and a learning rate of 0.0001 yields a better result. It could be interpreted that in the first several interactions, the model has learned a large number of essential features, and many features are reused in later learning processes due to the feature of this neural network. In addition, the learning rate at 0.001 and batch size at 32 or 64 is the effective combinations of the hyperparameters for our model. To further explore the possible combinations with larger batch sizes, we included 256 in our batch size.

As shown in Figure 7, batch sizes 32, 64, and learning rate 0.001 generate models with high accuracy and limited loss. It could be understood that these models are learning in an appropriate footstep and with a decent feature updating rate. In order to further explore the influences of the number of epochs, we continue to train the models for 50 epochs.
Figure 8. Validation accuracy (upper) step loss (lower)/step epoch
32 Batchsize/0.001 learning rate

Figure 9. Validation accuracy (upper) step loss(lower)/step epoch
32 Batchsize/0.001 learning rate
Based on the Figure 8 and 9, there is overfitting in the second model since the validation accuracy of our model stopped increasing. A tremendous difference between these models' inaccuracy and loss based on the graphs above. Based on performance stability and running time, we should adopt the model with learning rate 0.001, batch size 64 and train it 81 times. Table 10 shows the performance of the final model.

| Test result | Description | Batch size | Learn rate | Test Accuracy | Loss   |
|-------------|-------------|------------|------------|---------------|--------|
| Model      |             | 64         | 0.001      | 0.99          | 0.00004|

The final model reaches 99% accuracy and become a relatively most practical situations.

4. DISCUSSION

We found that a base VGG-19 model with no modifications could generate a validation accuracy of 0.9375 but may suffer from overfitting. We found that the best way to increase model performance and reduce overfitting on the dataset side was to use an 80-10-10 split of the data and perform data augmentation. Our findings suggest that using both dropout and early stopping reduced overfitting in the model. Using random search, we discovered that the best performing model used 30 epochs, batch size of 64, the learning rate of 0.001, and utilized a (3500 neurons, Relu) - (3000 neurons, Relu) - (3900 neurons, Relu) sequence of fully connected layers. The final model had a test accuracy of 0.9487 and loss of 0.2078, which shows improved performance but may also indicate that there could still be overfitting in the model.

In Resnet model optimization, this paper adjusted a single variable every time and achieved good results. This paper does not train the complexity and variability of the linear combination of multiple factors. The following work can use Principal Component Analysis equal-weight model simulation to get the optimal parameter combination.

In addition, Kaiming He mentioned in his paper that the bn layer was used in the ResNet model to reduce the problem of over-fitting [7]. Although this paper has good results in the accuracy and overfitting of the model, it cannot be ruled out that methods such as dropout and the early termination can achieve better training results. In the following work, different methods of over-fitting can be considered to evaluate its influence on the model comprehensively.

As for now, the test accuracy of dense net final model is above 99 percent, but the model is suffering from overfitting. We could apply some data augmentation or random dropout in the neural network to improve its performance. Additional further work could be using a network with more layers and using grid search, random search to tune the model.

5. CONCLUSION

This paper conducts a comprehensive study on the classification task of facial mask recognition by the CNN model. Considering the complexity of traditional feature extraction and supervised classification methods and the applicability of other CNN networks to image size and data set size, we selected three more suitable networks and achieved good results[11]. In improving and perfecting the model, we adopted random search and multi-objective optimization methods to analyze the influence of the choice of hyperparameters on the experimental results. Finally, we established a good balance between training accuracy and overfitting. The experimental results show that the three models' training accuracies are all above 96 percent, and the result of Dense Net and Res Net are at 99 percent.

References

[1] J. Howard, A. Huang, Z. Li, et al, “An evidence review of face masks against COVID-19,” Proceedings of the National Academy of Sciences, vol. 118, no. 4, 2021
[2] World Health Organization, Mask use in the context of COVID-19: interim guidance, 1 December 2020, 2020

[3] Krizhevsky, I. Sutskever, G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” Advances in neural information processing systems, vol. 25, pp. 1097-1105, 2012

[4] A. A. M. Al-Saffar, H. Tao, M. A. Talab, “Review of deep convolution neural network in image classification,” in 2017 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET), 2017, pp. 26-31

[5] K. Simonyan, A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” arXiv preprint arXiv:1409.1556, 2014

[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, 2016, pp. 770-778

[7] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in European conference on computer vision, 2016, pp. 630-645

[8] K. K. Bressem, L. C. Adams, C. Erxleben, et al, “Comparing different deep learning architectures for classification of chest radiographs,” Scientific Reports, vol. 10, no. 1, pp. 1-16, 2020

[9] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, “Densely connected convolutional networks,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700-4708

[10] Resnet, DenseNets & UNets. [Online]. Available: https://medium.com/swlh/resnets-densenets-unets-6bbdbcfdf010

[11] G. Mateo-García, L. Gómez-Chova and G. Camps-Valls, “Convolutional neural networks for multispectral image cloud masking,” in 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), 2017, pp. 2255-2258