Mask Detection

Matheus de Almeida
CS 542
Boston University
mfalm@bu.edu

Cyril Caparanga
CS 542
Boston University
cyrilc@bu.edu

Carolyn Wright
CS 542
Boston University
cwright4@bu.edu

Zachary Mecenas
CS 542
Boston University
zmecenas@bu.edu

Abstract

As masking has become increasingly common across the globe, the need for technologies to adapt to the change has grown too. Facial recognition systems that were so commonly studied in computer vision for a wide range of situations and circumstances in society suffered a setback due to Covid-19. Face identification algorithms were challenged to become more sophisticated in order to deal with the particularly obstructive face masks. Machine learning techniques had a significant impact on improving mask detection and masked facial recognition algorithms. The goal is to create a model that will detect whether someone is wearing a mask or not. For the purposes of this study we have decided to focus primarily on using a Convolutional Neural Network, namely VGG16.

1 Introduction

Face recognition systems had the benefit of working on non-occluded images. Essentially working on head shots, the systems could use every feature of a person's face to make it's judgements. However, with the sudden need for face masks during the pandemic, facial recognition systems must make judgements without facial features such as nose and mouth. This introduced a critical problem in places that authenticate people such as airports, immigration checkpoints, and access controlled environments (1). As such, masked face recognition (MFR) and occluded face recognition (OFR) deep learning models technologies have attracted substantial attention by researchers which have proposed various methods for improved performance (1; 2; 3; 4; 5). Nevertheless, this paper brings further benefit to the exploration and experimentation for improvement of convolutional neural networks (CNN) specifically for MFR.

2 Related Works

Research in MFR has significantly increased since the pandemic started. The goals ranged from finding the most accurate model to finding efficient/fast models to applying MFR to a real-world application.

At the onset of the pandemic, the National Institute for Standards and Technology (NIST) examined how well facial recognition algorithms performed against masked individuals (6). A Face Recognition Vendor Test (FRVT) in 2020 was used to determine how well existing technologies performed against masked data (7). The FRVT's purpose was to measure the performance of 89 commercial FR algorithms when encountering masks. The report provided many lessons about the new challenges that existing algorithms would need to tackle. Masked images increased failure rates to about 5% for the most accurate algorithms and increased failure rates to 20%-50% for other competent algorithms (6). In general, the masks made identification difficult, especially when masks covered more of a person's nose. Also, the shape and color of the mask affected the performance of the algorithms. Specifically, algorithms performed better when the face had a round, blue mask (6).

Paravision, a computer vision company, was one of the front-runners during NIST's FRVT in 2020 (7). They presented their challenges and considerations when working with MFR (8). Traditional head shots with masks were the easiest images to process. But, as additional facial occlusion was added, there were fewer critical facial features to use. For example, a tilted head with sunglasses and a patterned masks was considered one of the most difficult images to process (8). Paravision focused on face quality filtering, where they used a quality estimator and rejected images with quality values below some threshold. They also used multi-colored synthetic masks to improve the accuracy with respect to real masks. Overall, they were able to improve accuracy on real masks but still struggled with false non-match rate. One of the key conclusions was that frontal faces work best, but additional occlusions rapidly increased false non-match rate (8). A possible implication is that better images improve accuracy, but may reduce the general capability of the model once taken outside of the filtered setting.

Saravanan et. al. (2022) used image processing and a pre-trained VGG16 model. Modifying the images through transformations increased the number of images in their data set. They aimed to provide a scheme that avoided the cost of training a model from scratch. Taking advantage of VGG16's transfer learning, they used a pre-trained model and froze its layers, so only the last layer is trained (9). Against a data set of 1484 images, they achieved 96.50% accuracy and believed it could be further improved by customizing the final layer and using a larger dataset (9). By using a successful, pre-trained model, it is possible to achieve high accuracy with VGG16.

3 Data Set

The dataset we used to train, validate, and test the model came from a set of almost 12,0000 images of faces classified as masked or unmasked (10). The breakdown of the subsets can be seen in Table 1. Each subset has an almost equal proportion of masked and unmasked images. For example, the training data consists of 10,000 images that has 5,000 masked faces and 5,000 unmasked faces. The masked images were scraped from Google image searches and the unmasked images were taken from the CelebFace dataset. The images were then processed so that the images only contain a person's face with minimal background. However, the quality of the images still varied because of tiled faces, varied masks (medical masks, patterned masks, etc.), and additional occlusions such as hats or sunglasses.

The validation data is not used to train the model in any way. It acts as a checkpoint to verify which of our trained models is best and helps us make design decisions.

Table 1: Dataset Details (10)

Subset	Masked	Unmasked
Training	5000	5000
Validation	400	400
Test	483	509

4 Method and Results

4.1 Method

The foundation of our method is the VGG16 model which debuted in the 2014 ImageNet Challenge. VGG16 is a 16 layer deep convolutional neural network that classifies inputs of 224x224 images into 1,000 different categories ranging from goldfish to tandem bicycles (11). As shown in Figure 1, this is accomplished through a series of convolutional layers and pooling layers that learn different attributes of the images before determining their categories. The key point that separate VGG16 from its competition is its use of a single type of convolution filter and pooling layer. All of the convolutional layers in VGG16 use 3x3 filters and all max pooling layers are 2x2 with a stride of 2. The reason for this difference is that stacking 3x3 filters can replicate the same receptive field as larger filters while reducing the number of total computations required. These attributes made VGG16 an ideal candidate for our project as it is well equipped to handle small to medium datasets with relatively low training times (9). The version of VGG16 we worked with came from TensorFlow.

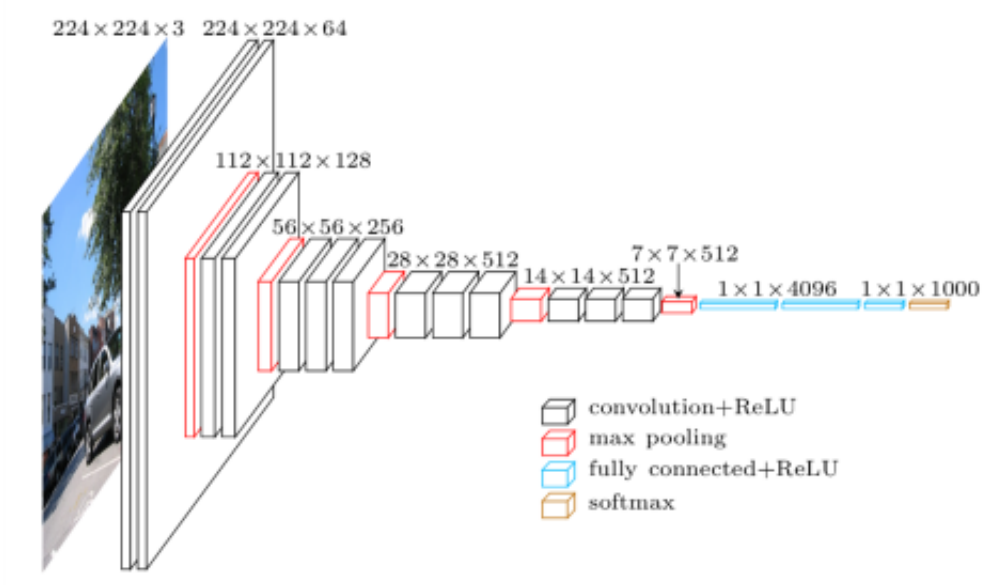

Figure 1: VGG16 architecture taken from (9)

Table 2: Dataset Details

Iteration	Input Weights	Layers	Optimizer	Loss	Accuracy
1	Randomly Initialized	- VGG16 with binary output	Adam	0.69	0.49
2	ImageNet	- VGG16 - Dense 1000 ReLU - Dense 500 ReLU - Dense 100 ReLU - Dense 2 Softmax	Adam	0.69	0.51
3-5	ImageNet	-VGG16 - Replaced final Softmax layer with binary	Adam	0.037-0.027	0.9649-0.9919

4.2 Implementation

Our process for implementing our model involved tracking our settings and the results of the training in a table. As shown in Table 2, the settings we monitored were the weights, architecture, and optimizer. The results we tracked were categorical cross-entropy loss and accuracy. Accuracy was a suitable metric for our work because of the equal distribution of our dataset into the two classes. For all of our training runs we implemented early stopping if training saw no improvement in loss within 20 epochs.

Our initial implementation utilized the VGG16 architecture with randomly initialized weights. We thought our model would be more effective if the weights for the entire model were learned from our dataset. After some experimentation, we found that the randomly initialized model performed about as well as, if not slightly worse than, random guessing.

Our next step was to use the VGG16 model with weights pre-trained on ImageNet. Using the pre-trained model meant that we had to retain the exact structure from the ImageNet classification which included the output layer with 1,000 classes. Our first solution to this issue was to expand the architecture with a series of dense layers of decreasing size, leading into a new classification layer for our two classes. This iteration remained close to 50% accuracy.

Our second solution aimed to get the most out of the pre-trained weights. We limited our randomly initialized parameters by only changing the VGG16 classification layer and freezing the layers in the pre-trained model. We accomplished this by loading the VGG16 model, then looping through its layers adding all but the last classification layer to our own architecture. We then added a 2 node softmax dense layer for our binary classification. This change led to our largest improvement scoring 96% accuracy on our testing data after training for about 120 epochs. With these same settings we trained two more versions that increased the accuracy to 98% then 99%, but we are not sure where the improvements are coming from.

4.3 Results

Figure 2 shows our accuracy rate and our loss for the training and validation set over time. This example is pulled from our Iteration 4 which had a final accuracy rate of 98% on the test set. This plot shows that our early stopping is working effectively and is avoiding over-fitting and memorization. The accuracy for the validation data does not begin to dip down over time and the loss does not begin to increase. Given this result in combination with our accuracy rate on our test set, we felt that this model was performing satisfactorily.

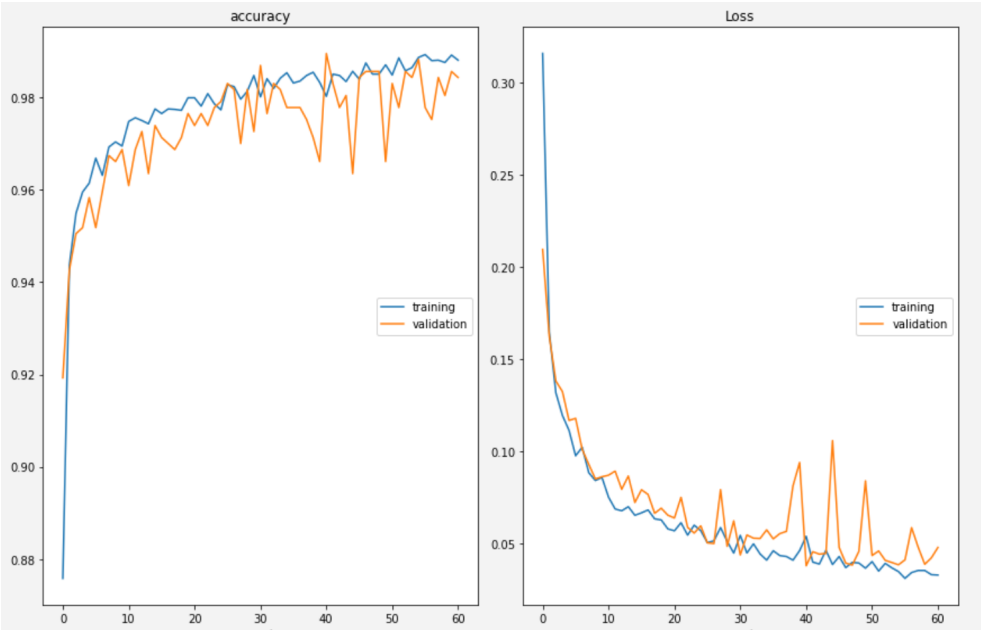

Figure 2: Plots of Loss and Accuracy

We next tested our model on 20 randomly selected images of our test set and visualized how our model was performing. Figure 3 shows an example of 20 randomly selected images along with their true and predicted labels. The true labels can be seen in parenthesis above the image and the predicted labels can be seen above without parenthesis. Images that were correctly classified have green text, while images that were incorrectly classified have red text. As can be seen here, our model performed very well on this small sample with a 95% accuracy rate.

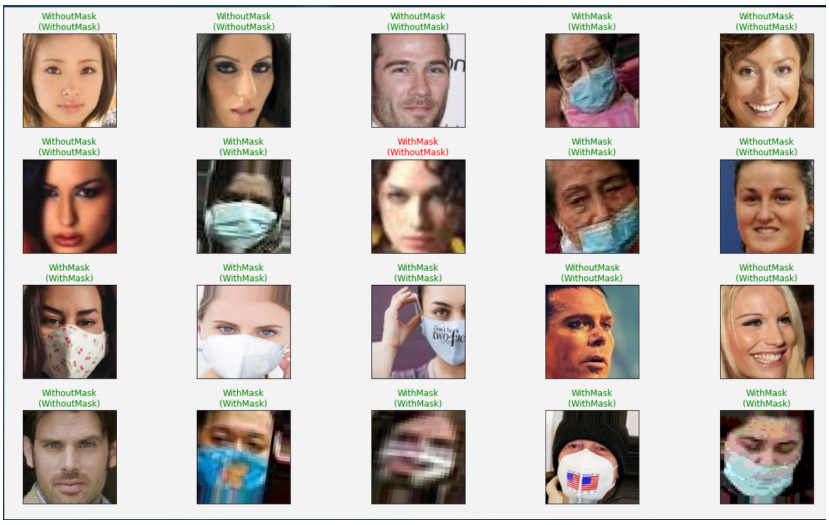

Figure 3: Example Using Test Set (n = 20)

Additionally, we tested the model on some of our own images as seen in Figure 3. Prior to cropping these images down to their current sizes, our model did not perform as well as we had expected. We found that this was due to the amount of background there was in an image and the area of pixels which VGG16 was focusing on. To improve upon the results, we decided to try to replicate the pre-processing done on the dataset by cropping these images to just the face. It proved to enhance our model's ability to assess whether an image had a mask or not. This highlights a major limitation to our model. Unless the model is fed pre-processed images that have been cropped to strictly faces and limited background noise, it will not perform as well as the current accuracy rate suggests. This is something that we would like to address in future iterations of this model to make it more generalize-able.

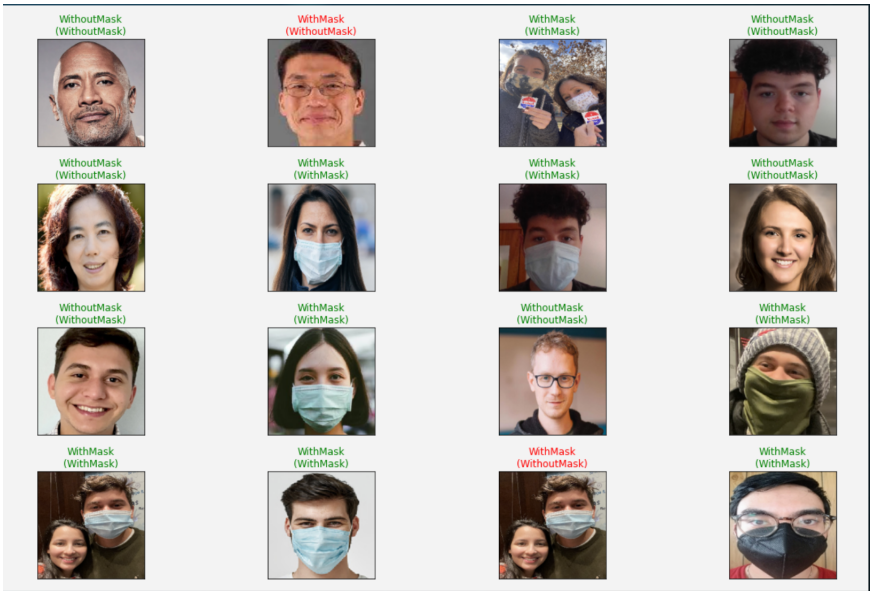

Figure 4: Example Using Personal Images (n = 16)

4.4 Future Work and Impact

Probing deeper, the results in this paper provide a strong vision towards possible future work to further our model investigation. By tailoring a face recognition model onto our current model, the

increased variety could allow the model to deal with multiple faces at once as well as address our limitation of only using cropped images as input. This could also open the possibility for real-time recognition. Another potential area would be applying the model to a dataset that has user identities and expand on the possibility of masked identification. The advancements mentioned would assist with surveillance of occluded face identification, mask monitoring in hospitals and airports, and a variety of other uses.

5 Conclusion

In this project, an effective convolutional neural network learning was explored using a VGG16 model as a base. VGG16 was chosen for its good performance on medium sized datasets and the small computation requirement, i.e. allowing for a short turnaround time when training the model. The final modified iterations, which changed the final class layer from 1000 to 2, presented a high accuracy of 96-99% with the pre-processed dataset used. The accuracy decreased slightly when tested on a custom set of images that did not go through the exact same image processing steps as the training data. Despite this, the results of our model showed improvement of about 1-3% in accuracy values when only using our chosen dataset.

References

[1] A. Ahmad, F. Albalas, T. AL-Hadhrami, B. Y. Lojin, and A. Bashayreh, "Masked face recognition using deep learning: A review," *Electronics*, vol. 10, no. 21, p. 2666, 2021.

[2] A. Chavda, J. Dsouza, S. Badgujar, and A. Damani, "Multi-stage cnn architecture for face mask detection," 2020.

[3] Z. Cao, M. Shao, L. Xu, S. Mu, and H. Qu, "Maskhunter: real-time object detection of face masks during the covid-19 pandemic," *IET Image Processing*, vol. 14, no. 16, pp. 4359–4367, 2020.

[4] P. Mohan, A. J. Paul, and A. Chirania, "A tiny cnn architecture for medical face mask detection for resource-constrained endpoints," *Lecture Notes in Electrical Engineering*, p. 657–670, 2021.

[5] K. Lin, H. Zhao, J. Lv, C. Li, X. Liu, R. Chen, and R. Zhao, "Face detection and segmentation based on improved mask r-cnn," *Discrete Dynamics in Nature and Society*, vol. 2020, p. 11, 2020.

[6] C. Boutin, "Nist launches studies into masks' effect on face recognition software," Aug 2020.

[7] M. Ngan, P. Grother, and K. Hanaoka, "Ongoing face recognition vendor test (frvt) part 6a:." `https://doi.org/10.6028/NIST.IR.8311`, Jul 2020.

[8] B. Avasarala, "Face recognition with masks." `https://pages.nist.gov/ifpc/2020/presentations/3a_NIST_IFPC_2020_Paravision.pdf`, October 2020. Presented at NIST International Face Performance Conference.

[9] T. Saravanan, K. Karthiha, R. Kavinkumar, S. Gokul, and J. P. Mishra, "A novel machine learning scheme for face mask detection using pretrained convolutional neural network," *Materials Today: Proceedings*, vol. 58, Part 1, pp. 150–156, 2022.

[10] A. Jangra, "Face mask detection 12k images dataset." `https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset`, May 2020.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations*, p. 14, 2015.

A Code Link

PDF: `https://drive.google.com/file/d/19BFPfDALd0fMOKz_YIoy9gH5vE30ID1G/view?usp=sharing`

Jupyter Notebook: `https://drive.google.com/file/d/1QByP5E2QHnhs0Pe0PvpIzjv6LRycWzjs/view?usp=sharing`