IMAGE CLASSIFICATION: FACE MASK DETECTION

Michael Davies
School of Data Science
University of Virginia
Charlottesville, VA 22903
mld9s@virginia.edu

Akeem Wells
School of Data Science
University of Virginia
Charlottesville, VA 22903
ajw3rg@virginia.edu

Preston Parrott
School of Data Science
University of Virginia
Charlottesville, VA 22903
pp3bd@virginia.edu

December 5, 2021

ABSTRACT

We present here two broad deep learning approaches applied to face mask data: Classification (mask, no-mask) and Object Detection (where is a potential face mask). First, we obtained data from Kaggle.com to implement two classification models: ResNet50 and VGG16. Both models perform exceedingly well. Second, we scrubbed images from google images and added a few images from the original data set in order to have images with greater diversity. After annotating the images, we implemented a YOLOV3 (You Only Look Once) model. Ultimately, the YOLO model also performed exceedingly well.

1 Motivation

We implemented two broad approaches to face mask data: Classification and Object Detection. We largely treat this as separate approaches because image classification very likely will not perform well on images that contain both masked and unmasked people. Therefore, our data for the classification models are divided into images of mask and no-mask. Conversely, to implement object detection, we scrubbed images from google and added a few images from the original data set in order to have images with multiple people (some with masks and some without.) In both cases, we leveraged pretrained models - imagenet and coco respectively.

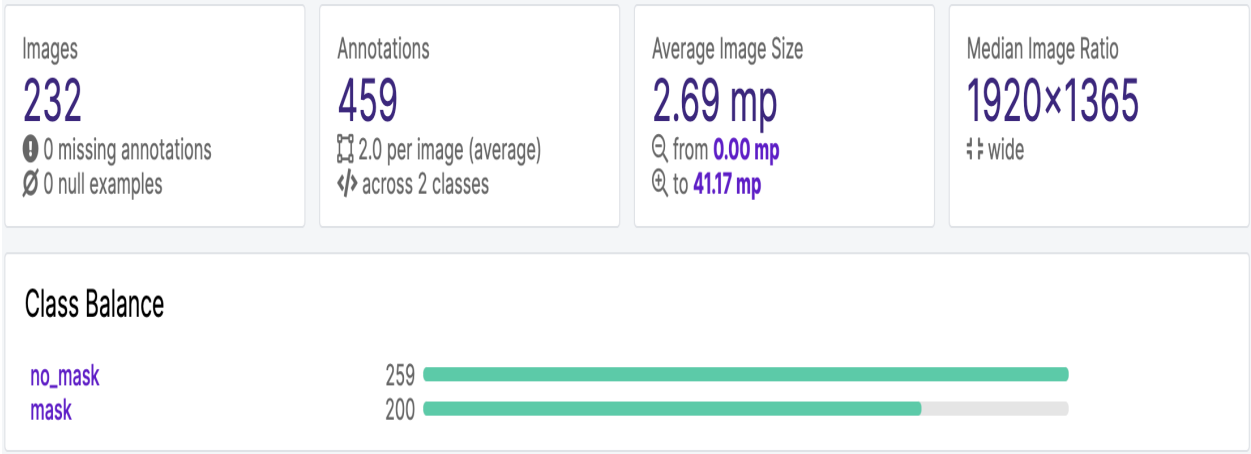
2 Data

2.1 Data for Classification

We are using the "Face Mask Detection using CNN" dataset from Kaggle. The dataset contains 5000 images, and each are labelled for binary classification: with mask and without mask. In the ResNet50, after scaling, we augmented the data with brightness, contrast, and hue.

2.2 Data for Object Detection

The initial data selection was annotated from a small subset from the data at Kaggle.com using https://roboflow.ai/. This data proved to need updating due to various reasons including lack of diversity between faces, lack of diversity between masks, image resolution, etc. Additionally, images from the initial data set only contained images with one mask to be annotated. We scrubbed several additional images from Google, which had better resolution, multiple faces (some with masks and some without), and greater age and racial diversity. We then annotated them with bounding boxes around the face masks using roboflow.



3 Methods

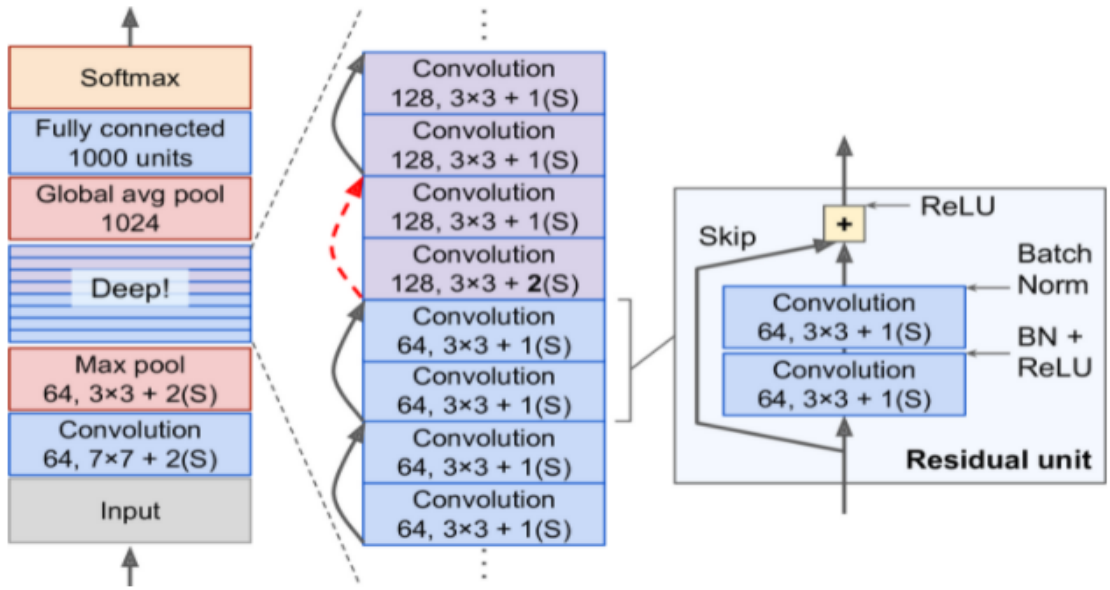
At this stage, we have implemented the following methods:

- ResNet50 (classification)
- VGG16 (classification)
- YOLO (object detection)

3.1 Classification Models

3.1.1 Resnet50

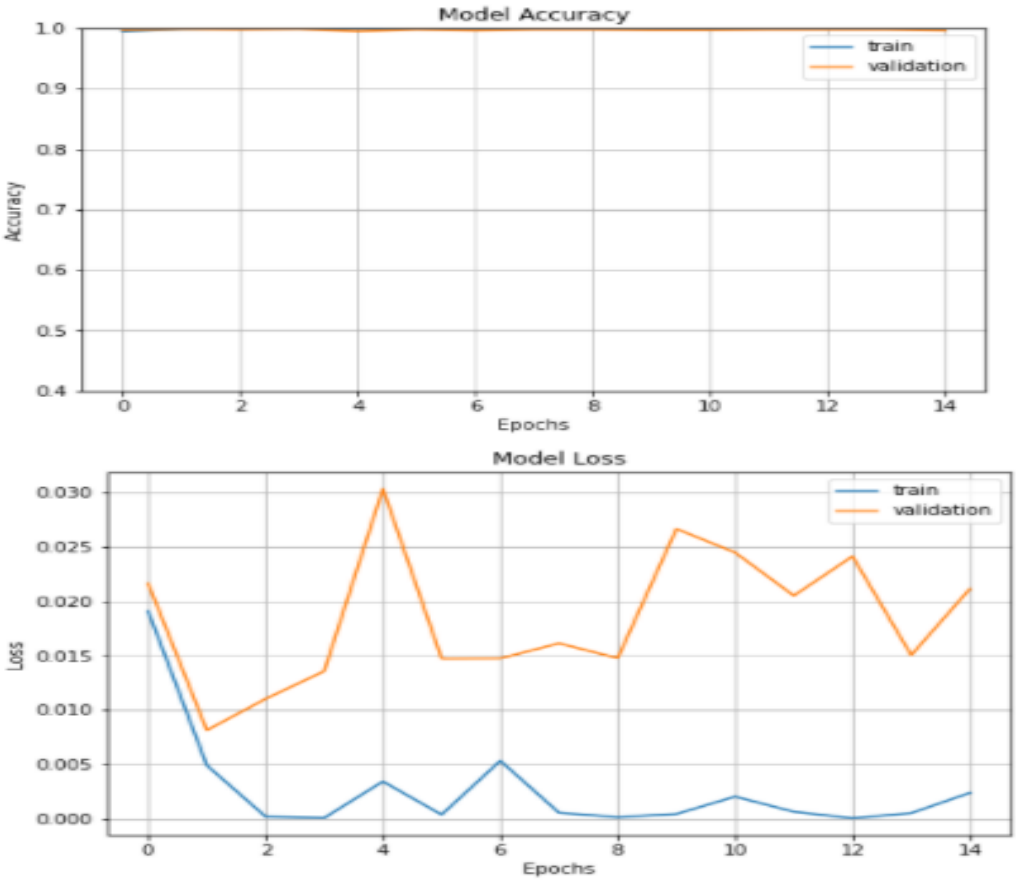
We first implemented a ResNet50 for image classification. This algorithm is well known for relatively strong performance and fast runtime. The key distinction of ResNets is its application of residual learning via skip connections, which effectively are shortcuts to jump over some layers. This avoids the problem of vanishing gradients and ensures the signal can easily make its way across the whole network.



In terms of model design, the ResNet pre-trained weights are from the imagenet. We implemented a standard optimizer (Adam), drop out (0.5), and a final dense layer of 2048. With the layers frozen, we set epochs = 5. Then we unfroze addition layers and set our epochs = 15.

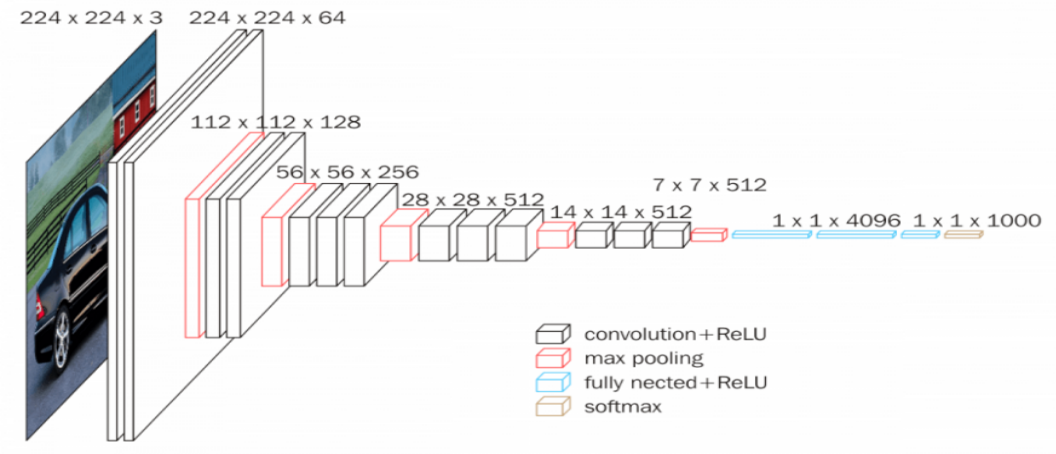
Results

The final performance results were: Accuracy: 0.9997, Val Accuracy: 0.9987, Loss: 0.0008 and Val Loss: 0.0127. Given the results (an validation accuracy over 0.99), this appears to be an easy classification problem. Of course, ResNets are famously good performers. In addition, the image data was not likely very challenging. The images were almost exclusively close up images of a single face mask, and there was very little diversity in terms of race and age.



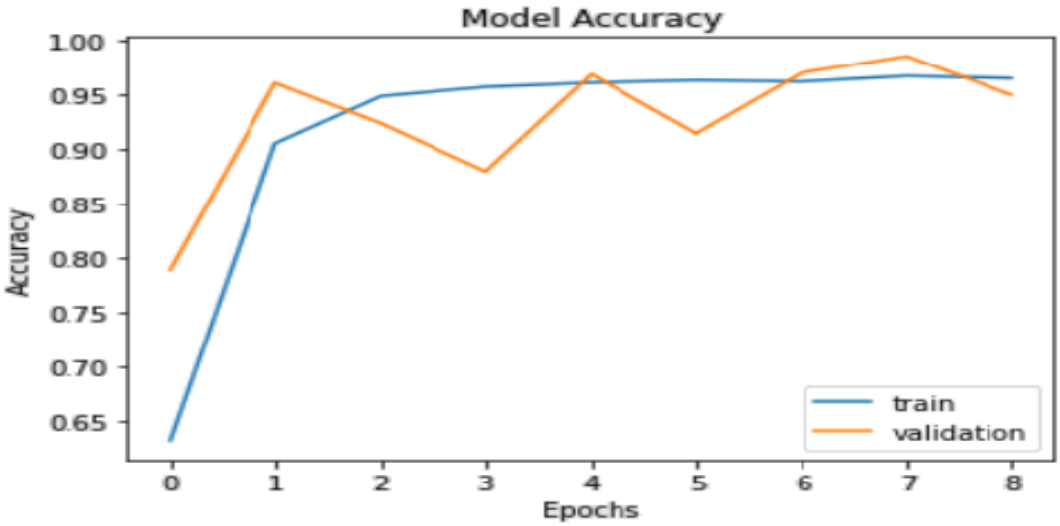
3.1.2 VGG16

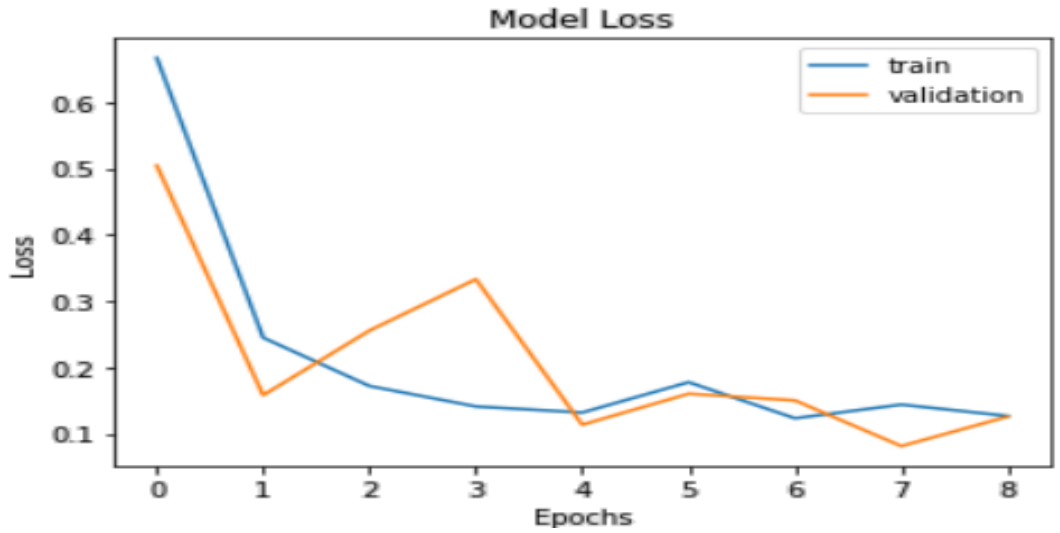
Following the implementation of the Resnet50, a VGG16 was trained. The choice of this algorithm was due to its contrast in technique and depth. The model has 138M parameters with a notably simplistic and interpretable architecture given its use of 3×3 filters stacked on top of each other in increasing depth. Reducing volume size is handled by max pooling. Two fully-connected layers, each with 4,096 nodes are then followed by a softmax classifier. However, due to its depth and the number of fully-connected nodes, the VGG16 model size is over 533MB which slows training. Additionally,the network architecture weights are large which can be concerning as it relates to bandwidth.



Results

Optimal performance was achieved after 9 epochs at 200 steps per epoch. The model performed well on both training and validation data as indicated by performance results: Accuracy: 0.973, Val Accuracy: 0.985, Loss: 0.13 and Val Loss: 0.91.





3.2 Object Detection

3.2.1 YOLOV3

As discussed above, this data appears easy to classify. Therefore, we extended this project with an object detection model. This also required additional changes to our working data set.

YOLO (You Only Look Once) is a Convolutional Neural Network (CNN) that "applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region."

The current data set contains 232 images with 459 total annotations of 259 people without masks and 250 people with masks

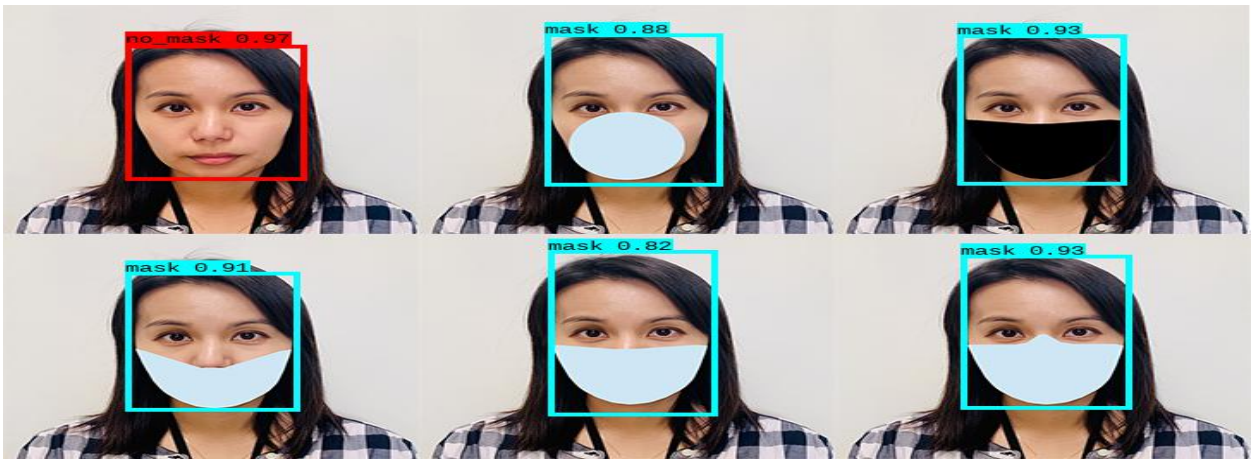
From this we applied several augments including changes in hue, contrast, brightness, exposure, and rotation. This yields us a training set of 696 images.

In order to achieve this, we implemented the YOLOV3 model using TensorFlow 1.0 (Keras 2.2.4). We've modified a version of YOLOV3 provided by (https://github.com/roboflow-ai/keras-yolo3). We pretrain our model (62,000,000 params) using weights from (https://pjreddie.com/media/files/yolov3.weights) on 557 Training Samples and 139 Validation Samples.

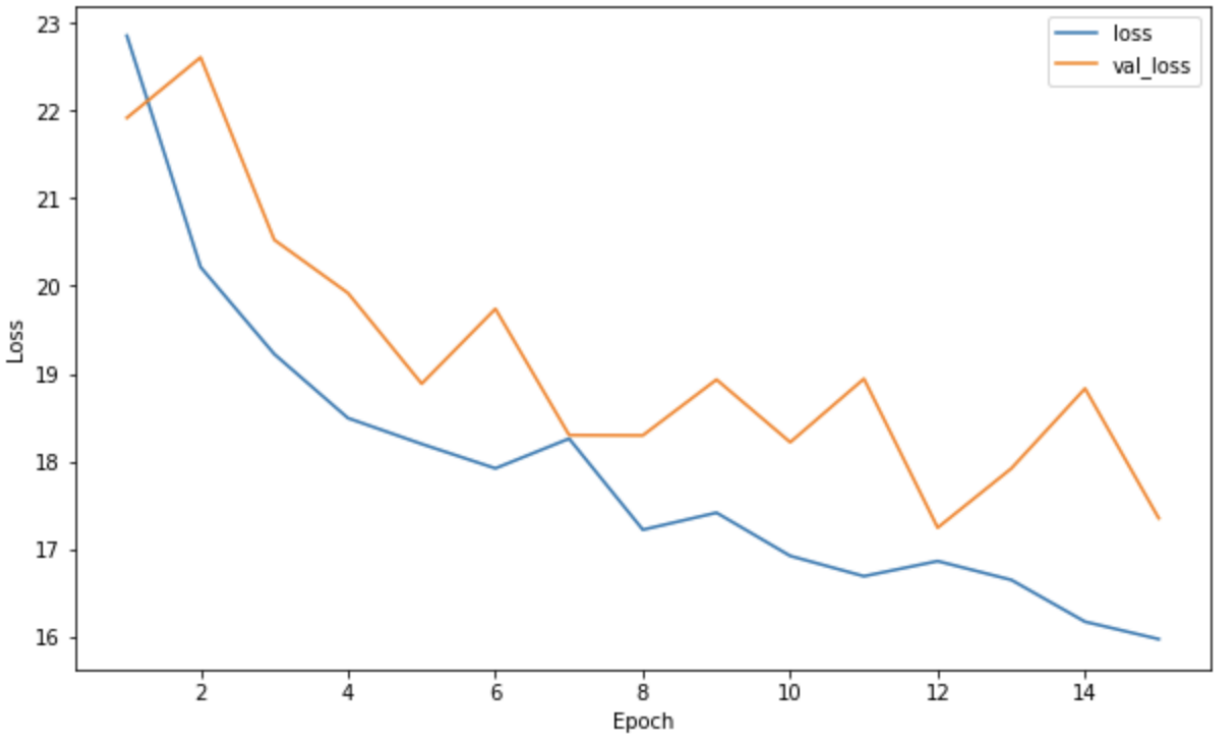
Initial Results

From our YOLOV3 model, we were able to take an test image

and produce bounding boxes with probability of the correct class.



YOLO uses sum-squared error between the predictions and the class label to calculate loss. The resulting losses for our YOLOV3 model were as follows:



4 Conclusion

Given the strong performance in the classification models (ResNet50, and VGG16), this appears to be an easy classification problem. This is unsurprising. Of course, this problem is ideally suited for deep learning algorithms such as ResNets and VGGs. In addition, the image data was not likely very challenging. The images were almost exclusively close up images of a single face mask, and there was very little diversity in terms of race and age.

Within the performance of the YOLOV3 model, we can see accurate results in detecting people wearing a mask or not. The model also is correct at not bounding standalone masks presented in an image. Due to long training times and limited resources, the model sacrifices accuracy for reduced runtimes. Therefore, the loss graph above appears to not show loss convergence.

5 Bibliography

https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/

https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96

https://learning.oreilly.com/library/view/hands-on-machine-learning/9781492032632/ch14.html#idm45022138648968

https://github.com/roboflow-ai/keras-yolo3

https://viso.ai/deep-learning/yolov3-overview/

https://colab.research.google.com/drive/1ByRi9d6_YzuOnrEKArmLMLuMaZjYfygO#scrollTo=WgHANbxqWJPa

https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b

https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

6 Member Contribution

While each member took lead on one particular model, we consulted as a group during the process.

- ResNet50 - Michael Davies
- VGG16 - Preston Parrott
- YOLOV3 - Akeem Wells
- Image scrubbing (additional images for object detection) - all
- Image annotations (labeling bounding boxes) - all
- Data hosting (github) - all but hosted Akeem's github
- Written product - all