Mask/No-mask - A Face Mask Detection System for places with High Infection Risk

Yathukulan Thayabaran
San José State University
Department of Computer Engineering
San Jose, California
yathukulan.thayabaran@sjsu.edu

Sean Wu
San José State University
Department of Computer Engineering
San José, Santa Clara
shao-an.wu@sjsu.edu

Rajeev Sebastian
San José State University
Department of Computer Engineering
San Jose ,California
rajeev.sebastian@sjsu.edu

Aayushi Gupta
San José State University
Department of Computer Engineering
Sunnyvale, California
aayushi.gupta@sjsu.edu

***Abstract*—This paper documents the approaches and their respective results of the authors' attempt to classify images of human faces as wearing masks or otherwise. Several machine learning algorithms were employed to produce a great degree of accuracy. The intended application of this project is at the gateways of indoor locations with high visitor frequency. This model can be used to deny entrance to individuals that don't abide by the mask regulations of a venue. In this application, false positives for masks have to be minimized while keeping the false negatives at a bearable level.**

Keywords— Machine learning, Mask detection, Image classification, Convolutional Neural Networks, ResNet

I. Introduction

Starting late 2019, Covid-19 pandemic has affected our lives a great deal and we have been dealing with new challenges everyday. Slowly this is becoming the new normal - the changing way of life. As instructed by the authorities, we should wear a mask when we are in public as wearing a mask will help in stopping the spreading of this virus. However, enforcing the said laws has been a tedious task for businesses and governments alike. For example: at big stores such as Safeway, thousands of people visit daily but it is very difficult to keep a track of who is wearing a mask and who is not. Therefore, we propose a system that identifies if an individual in front of a camera is wearing a mask or not. The problem is twofold: Identifying one's face in a picture, and determining if they have masks on. Masks come in various shapes and colors, further compounding the problem. The final algorithm would run on image snippets of the cameras based on entrances of venues where our solution will be implemented.

Various research papers on the topic were read and analyzed for any critical insight on the topic. Based on this research, we identified the state of the art methodologies in this domain. The labelled dataset was pre-processed per our requirements. Given the complexity of the data, feature extraction was a challenge. This was followed by the design of the model and the algorithms used, model training, and model evaluation.

II. Related Work

In the beginning of the machine learning era, neural networks were utilized extensively to perform classification, especially with images as other machine learning algorithms such as SVM, which suffer the disadvantage of not being able to deal with input features with high dimension. To research into how to detect whether a person is wearing a face mask or not, it was worth digging into face detection and face recognition first; In this paper [1] an algorithm was proposed, they use a technique to subsample the input image and then downscale it to perform classification. However [1]could only classify faces that are upright, frontal and uncovered. In another paper published on IEEE[2] comes a method that takes advantage of multiple networks to detect faces even if it has been rotated. As of researching into face recognition, which helps even more of this paper, Convolutional Neural Network(CNN) has been researched extensively by previous researchers, such as in this paper[3], a hybrid method for human face recognition has been proposed, which it combines local image sampling, a self-organizing map (SOM) neural network, and a convolutional neural network. Other than the hybrid method of CNN above, neural networks that are facilitated by probabilistic approaches [4] and deep neural networks [5] have also been discussed.

In this project the dataset that we use is from Kaggle, a leading machine learning website for engineers to compete, learn and grow. "With/Without Mask" is the repository we do testing classification for development purposes, later on "Face Mask ~12K Images Dataset" is the repository we will be training our model in large batch size which contains roughly 5000 images with masks, 5000 images without masks for training cases and 400 images with masks, 500 images without masks for test cases.

III. Approaches and tools

A. Vanilla Neural Network

Right when the acquisition of the "With/Without Mask" dataset, there was only around 500 images containing faces with mask and without mask, therefore data augmentation was required; first we read the dataset with Pandas library, then we extracted the paths for the files that is in the training set. With the training set in hand, data preprocessing was done, we then read the image with openCV python module to produce the inverse grayscale image object, then the inverted grayscale object was thrown into a scalar to scale down the image that was originally read into the program. By applying the method

ImageDataGenerator, different orientations, flips and stretches of the image was obtained, the result of the augmented image could be seen as follow:

Figure 1. Data augmentation through rotation

With the newly enhanced data, a simple vanilla neural network (NN) was spun up to start classifying. To transform the 2 dimensional array of image into a flat input vector, a function provided by Tensorflow Keras called "flatten" was called, then the result of the flattened input vector was fed into three layers of Dense neural network each with 1000, 20000, 500 and 1 neuron as the final output layer. For the activation function that has been chosen is 'relu', 'swish', 'swish' for hidden layers, and 'sigmoid' for the final output layer; the reason of doing this is that the 'relu' activation function suffers from the "disappearing gradient" so 'swish' is substituted for it, then the 'sigmoid' will be used as the output activation function as a very common case.

Moreover, the learning rate was set to be dynamically decayed during every iteration, while the optimizer was chosen to be Stochastic Gradient Descent (SGD), which is found to be more effective than other optimizers during the development of the model, for example, SGD optimizer improved 10% in accuracy compared to that of AdaGrad.

In the final hypothesis, NN yielded the result of 94.28% in sample accuracy and 91.19% out of sample accuracy, it is worth noticing that this is done without any layers of convolution nor pooling.

B. Convolutional Neural Network

Based on our research and according to what we learned in the class, convolutional neural networks or CNNs are typically used to perform image detection tasks. We decided to implement a simple CNN to understand the results on our dataset. Using approximately 10k images to train, 10% as validation and another 10% as our test set, we proceeded to build this model. Using Tensorflow's ImageDataGenerator and flow_from_directory method, we performed real time data augmentation to generate variations of images by rotation, zooming, rescaling, etc as done in a vanilla neural network, explained above. Then using a function, converted each batch of data coming from the ImageDataGenerator into an array. From here, we obtained training, validation and testing sets with X and y arrays pr shape 120 x 120 x 3.

A convolutional neural network consists of convolutions or filters that detect specific features for example edges. In CNNs, multiple filters are taken to slice through the image and map them one by one to identify different parts of the input image.

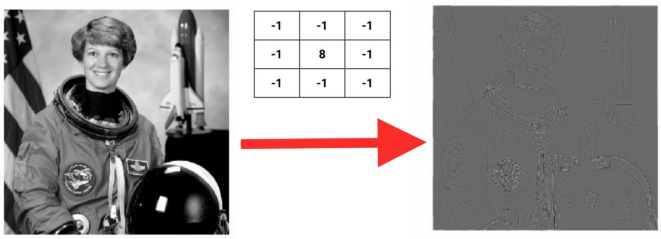

Figure 2. Convolution operations (source)

As indicated in the above figure, the image to the left is filtered using the 3 x 3 matrix(filter) to get the boundary edges, as shown in the picture to the right.

We built our CNN model using python's keras library. This is a sequential model with a convolutional layer with a filter of size 2 x 2. Applied a "Relu" activation function to introduce non-linearity and performed a dropout to avoid overfitting. This was followed by a max pooling layer to summarize the features and reduce the dimensionality for efficient calculations.

A max pooling layer(or in general a pooling layer) helps in down sampling feature maps by summarising the presence of features in patches of the feature maps.

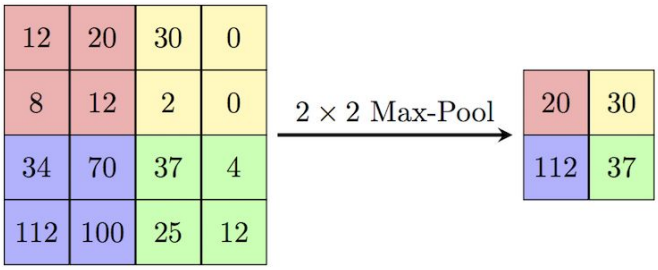

Figure 3. Max pooling operation

This sequence was repeated again to perform further feature extraction. The output sequence from the above layers was flattened and passed through two dense layers with "Relu" activation. The output layer consists of a dense layer with "sigmoid" activation, since this is a binary classification problem. We tuned the hyper-parameters using the validation set and performed an early stopping at a minimum loss of 10^-13 to avoid overfitting.

Using sklearn's evaluation matrix, we obtained a confusion matrix that looked as below:

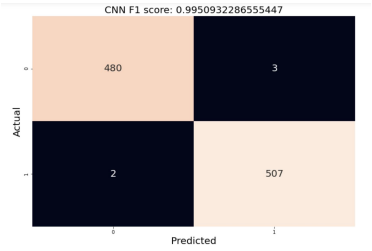

Figure 4. confusion matrix of the CNN

C. ResNet34 through Fastai

Approaches using ResNet34 were employed. The fastai v2 library was used to run the resnet models. Given the amount of computational resources required and the possibility of parallelizing big operations, it was deemed better to run the model on the GPU. The pytorch library (v10.1) was used to implement Nvidia CUDA enabled processing of certain calculations. The code was run on google colab to utilize the powerful AI optimized Tesla P4 GPUs available through the said service.

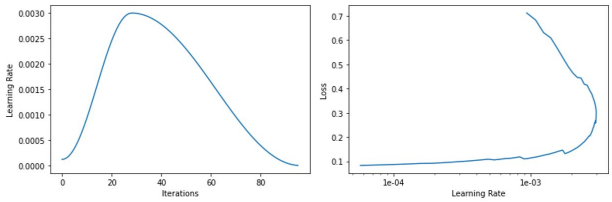

Figure 5. Cyclic learning rate and its effect on the loss function

The graphs above show the learning rate change throughout the iterations and how it affected the loss function. The training used the cyclical learning rate (model.fit_one_cycle) approach, where instead of training the model with a fixed learning rate, the learning rate was varied throughout the process of training. Therefore, the cyclic learning rate approach used here sweeps from the lowest to the highest and back down to the lowest learning rate during one cycle. This variance of the learning rate during training helped achieve more accurate models faster. The batch sizes were also tweaked during learning. Bigger batch sizes produced smoother loss curves and overall better in sample accuracy. However, given the possibility of overfitting, the batch sizes were set to 25 samples per epoch. Note that the batch sizes were tweaked before the test dataset was evaluated; with the results of the classification shown below.

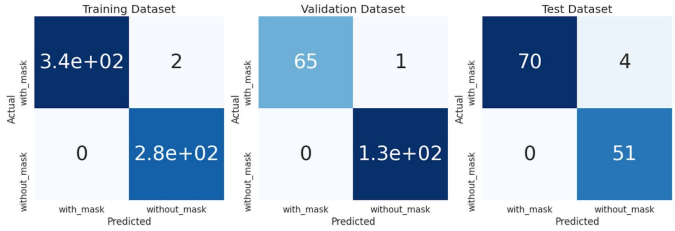

Figure 6. Confusion matrix of the ResNet34 model

It should be noted that the ResNet34 model produced no false positives for all training, test, and validation datasets. See figure above for the confusion matrix for the various datasets used.The model produced a test set accuracy of 0.968 with an f1 score of 0.972. The validation and the test sets both had an accuracy of 0.99 or better with a simile high f1 score. Given the application context of this model, the false positive rate would be the most important measuring metric with the recall metric coming in second. Shown below are some of predictions of the model.

Figure 7. The classification output on hand picked images of unseen data chosen for their variety.

Some of the images shown above are from the test dataset. However, some are chosen purely out of curiosity. The model was able to handle various hair and skin colors. It was also not affected by hair or facial hair. When given pictures of just masks without their wearers (see bottom left), the model was still able to classify the images as masked. It appears that we trained a classifier to predict the appearance of a mask on an image. This also applies to masks that are incorrectly worn (not covering the nose, cut outs for the mouth(See top right image)). Therefore, this classifier assumes the masks are always worn correctly. The color of the masks also seem to be a non issue for the classifier. Through further testing, it was found out that any article of clothing or equipment on the upper part of the subject's face does not affect the classification. For example, in the pictures of Batman and the skateboarder, the classifier ignored the said features. The images of just masks, Batman, Darth Vader etc. are not part of the datasets and were fed from a dataset that is custom made for "borderline" examples. The classification of the borderline examples changed chaotically for each run of the notebook, thus the name of the examples that were somewhat chosen, as discussed, just out of curiosity.

As to answer the question how the model classifies an image, further experimentation was required. One can catalog the steps of the backpropagation of the network and possibly fetch the activations and the gradients associated with a specific input image. This part helps greatly with the intuitive understanding of the model's operation. However, one can read too much into the said outputs and maybe mislead as to the real performance of the model. Seen below is an example of the extracted information overlaid on top of the original image. The heatmaps are color coded to distinguish from the lowest (blue) the highest (red) values.

Figure 8. Without mask - Average activations and gradients of the image overlaid as heatmaps. Notice how the strongest average activations are always around the subject's face.

Figure 9. With mask - Average activations and gradients of the image overlaid as heatmaps. The strongest gradients are around the mask.

When considering the images seen previously, one can clearly notice that the activations of the layers tend to "track" the subject's face. It is not the case with pictures of multiple subjects as this model was trained mostly on single subject images. The model discussed later handles multiple subjects. The gradients seem to cluster around the edges of the mask. The astute reader might have noticed that the resolution of the heatmap seems lower compared to that of the image. This is because the heatmap is only a 7 x 7 matrix. This is possibly because of the structure of the ResNet34 model where the kernel is set to a 7x7 matrix. This is an assumption given that the pooling layer is only 2x2 with a stride of 2.

The activations and the gradients were extracted with the help of the hook_output function of the fastai library. With access to the tensor values of a prediction, one can peel back each layer of the network and obtain the activations for the specific layers. A cuda call is made to transfer the CUDA tensor to the processor where the hooks are calculated. The average activations and the gradients are averaged to find the matrix for each image. The figure below shows all four misclassifications in the test dataset.

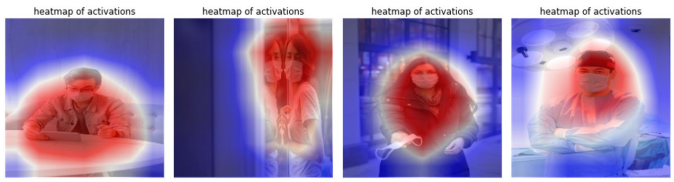

Figure 10. Activation gradients of all the misclassifications of the test dataset.

All the misclassifications were false negatives. For images 2 and 3, we can notice more than two masks in the picture. For image 4, the color of the scrubs of the surgeon matches that of the mask. Possible reasons for the misclassification of image 1 are still unidentified.

Upon taking up the suggestion made by the professor during presentation day, other applications of the model were considered. The predictions were applied to pictures of animals to gauge the effectiveness of this application on non-human subjects. The classifications often produced worse-than coinflip accuracies for images of cats and dogs. Some examples are shown below. The model had a high bias towards predicting the snout of the dogs, perhaps given the high contrast between the snout and the rest of the face, as masks.

Figure 11. The results of the attempted classification on non-human subjects. No animals were harmed in the process of obtaining these pictures per the authors' knowledge.

D. ResNet50 through PyTorch

Resnet 50 is another architecture of a residual network with 50 layers in it which was used to train on a new dataset. This data set consisted of true masks instead of the ones where the mask was added through a photo editing software. We made use of the PyTorch library to train on this new dataset. A pretrained model on the COCO dataset was used as the base model. The training was done for 25 epochs. Before training the images were converted into tensors which are general matrices with more than two dimensions. In this case they were a grid of 3 * 224 *224 pixels values. Training was done with stochastic gradient descent with a learning rate of 0.005. The weight decay parameter was 0.0005. Along with this it was necessary to find what part of the figure was the model finding it interesting. For this reason, another predictor was used to find the regions which the ResNet model found interesting. Around those regions bounding boxes were created. We found that it was the mask

that the model found interesting rather than just the face.

Figure 12. ResNet 50 bounding boxes on masks

Figure 13. ResNet50 Multiple bounding boxes

IV. Results

The ResNet34 model showed similar properties to the ResNet50 model for the loss function. The function for the training set appears to be able to handle more data before it saturates near its eventual horizontal asymptote. Therefore, the model performance standards improve with more data. (See image below). The function became smoother for higher batch sizes.

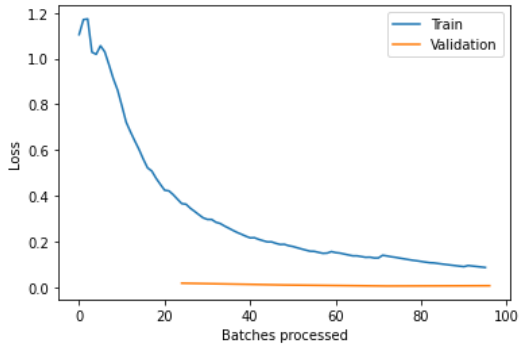

Figure 14. The loss function of the ResNet34

For the ResNet 50 implementation we trained the network for 25 epochs.to check if we are converging as in we are reducing the error ate each epoch we plotted the training error versus iteration as the training was going on.

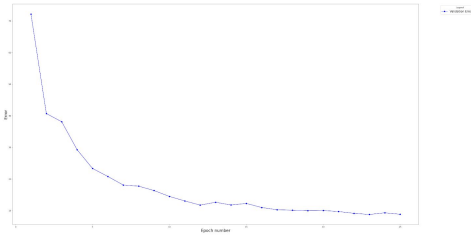

Figure 15. Training error

In order to see if the model was learning the faces with mask on them we also trained an auxiliary model. It was a pretrained model which detected heads of individuals specifically.

V. Conclusion

In this project we have experimented with various kinds of machine learning techniques to identify if a person in an image is wearing a mask. Starting from vanilla neural networks, where we used dense layers to predict the labels, we could achieve ~ 94% accuracy. We then worked with a Convolutional neural network to train on ~ 10k images to obtain an F1 score of ~ 0.99. A more advanced technique that we tried were the ResNet (ResNet34 and ResNet50) models, are faster to train and it is possible to understand from these models which features are the most important.

Future work includes creating an ensemble model of all the models pioneered by the individual teammates. One can also try to reduce the assumptions that we made in the predictions: that all subjects are human, All masks are worn correctly, and the images are right side up. Training the models with bigger datasets and using the previously mentioned flipping methods to create more samples from the given dataset can also be used. Another possible route, especially given the performance of the ResNet50 model, would be to create a real-time video classifier.

VI. References

[1] H. A. Rowley, S. Baluja and T. Kanade, "Neural network-based face detection," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 1, pp. 23-38, Jan. 1998, doi: 10.1109/34.655647. https://ieeexplore.ieee.org/document/655647

[2] H. A. Rowley, S. Baluja and T. Kanade, "Rotation invariant neural network-based face detection," Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231), Santa Barbara, CA, 1998, pp. 38-44, doi: 10.1109/CVPR.1998.698585. https://ieeexplore.ieee.org/abstract/document/698585

[3] S. Lawrence, C. L. Giles, Ah Chung Tsoi and A. D. Back, "Face recognition: a convolutional neural-network approach," in IEEE Transactions on Neural Networks, vol. 8, no. 1, pp. 98-113, Jan. 1997, doi: 10.1109/72.554195. https://ieeexplore.ieee.org/abstract/document/554195.

[4] Shang-Hung Lin, Sun-Yuan Kung and Long-Ji Lin, "Face recognition/detection by probabilistic decision-based neural network," in IEEE Transactions on Neural Networks, vol. 8, no. 1, pp. 114-132, Jan. 1997, doi: 10.1109/72.554196. https://ieeexplore.ieee.org/abstract/document/554196

[5] Y. Sun, D. Liang, X. Wang, and X. Tang, "Deepid3: Face recognition with very deep neural networks," arXiv preprint arXiv:1502.00873,2015.https://arxiv.org/abs/1502.00873

VII. Project contribution

We worked as a team to research and formulate the project idea,identify the dataset, framework and methodologies to implement. We then decided to implement one technique each and then consolidate the findings. Sean worked on implementing the vanilla neural network. He examined the data augmentation that helped us obtain a more robust dataset. Also, the application of a simple neural network to our problem helped us understand why we needed more specific approaches. Aayushi implemented the Convolutional Neural Network. She implemented convolutional layers and max pooling to identify the features in the images. From her work we understood how CNNs can help us reduce the number of parameters without losing on the quality of models. We then moved to some newer models which were used to combat the problem of vanishing gradient and the degradation issue which are residual networks. We moved forward with two of its implementation.Yathukulan worked on ResNet34 which has 34 convolutional layers, He trained them on the Google Colab hosted GPUs to reduce training time. He used the fastai hook implementation to grab activations and gradients to plot heatmaps of the gradients over specific images. Rajeev worked on the ReNet50 architecture. He made use of a different dataset with a real mask being used to see if it contributed in giving better performance over real world images.He also used a bounding box regressor to mark parts of the images which the classifier found important in making its decision

VIII. Supplementary material

Github for project code:
https://github.com/Sean-Wu-TW/257-Group-Project

Datasets:

1) https://www.kaggle.com/niharika41298/withwithout-mask
2) https://www.kaggle.com/andrewmvd/face-mask-detection/download
3) https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset