FACE MASK DETECTION

Shania Dhani

Xuejin Gao

Michelle Lucero

Machine Learning CSCi 353, Fall 2020

Table of Contents

Problem Description

According to the World Health Organization, one of the most effective protective measures against the COVID-19 virus is to wear a face mask when in public areas. While wearing a face mask alone is not enough to protect you from catching the virus, its effectiveness has led many countries around the world to mandate the use of face coverings for civilians when in public (WHO, 2020). However, monitoring people for face mask coverage becomes difficult in large groups and gatherings. Thus, countries like France have turned to the use of AI for detecting incorrect wear/absence of face coverings on individuals in public settings to facilitate contact-tracing efforts to track the spread of COVID-19 and to predict areas where outbreaks are likely (BBC News, 2020).

Given the suddenness of the current COVID-19 pandemic, relatively few research papers address the face mask detection models. Nevertheless, the research presented in our baseline paper as well as a few other papers hold promising results for accurately detecting the wear of face masks on individuals. The focus for our project will be to design several models that determine whether or not an individual is wearing a mask, not wearing a mask, or wearing a mask incorrectly. We will use Loey, Manogaran, Taha & Khalifa's study as a baseline to evaluate our research. We will also use two external datasets, one composed of people of color (POC), and the other composed of white people (W) to test how our models perform for different groups of people (Loey, M et al., 2021). We will use this information to assess the overall bias in our models.

Approach/Goals

The goal of this paper is to create a model that accurately determines whether an individual is wearing a mask, not wearing a mask, or wearing a mask incorrectly. It should be noted however, that the machine learning models currently available, while minimal, yield promising results. Currently, the paper, “A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic,” has created a model for detecting face mask coverage that performs with 100% accuracy over one of their test datasets (Loey, M et al., 2021). While the results appear to be promising, this research only accounts for the binary classification of wearing a face mask v. not wearing a face mask. In terms of real world application and efficacy, it has been shown that wearing a face mask incorrectly does not protect you from catching the COVID-19 virus (Maragakis, 2020). We plan to address this third classification in our research by combining two separate face mask datasets to improve the real-world application of our model, creating a new dataset that is labelled for mask v. no mask v. incorrect wearing of a mask. We will also build in a few more classifiers whose performance metrics were not given in our baseline study, including the KNN classifier and different types of CNN neural networks. This will help us determine which models are the best predictors for our problem set and also allow us to assess the bias in our algorithms across a diverse range of classifiers.

Contribution

Shania will be in charge of determining how to fit the mask v. no mask v. incorrect wear of mask dataset using a KNN Classifier and a Decision Trees Classifier. The goal will be to find the most optimal parameters for these models based on accuracy and error analysis metrics (such as the best number of neighbors for the KNN model). Xuejin will be learning and reading more on the different types of CNN (CNN, R-CNN, Fast R-CNN) and applying CNN and Naive Bayes to the Face Mask Dataset. The objective is to find a model that has a low computational time with really good accuracy. Michelle will be collecting and cleaning the image dataset for our training dataset and for the bias evaluation datasets (which include a people of color, POC, dataset and a dataset of white individuals, W) adding facial landmark features for classifying. Michelle will also work on implementing and analyzing the SVM model.

Related Work

This paper offers us a relatively high baseline accuracy for detecting face masks across three different Mask/No Mask datasets, and contains a SVM solution that performs at best 100% testing accuracy for the Labeled Faces in the Wild database. It explores deep learning tactics to improve the feature extraction and classification of face mask coverage and offers an insightful range of *classical* machine learning classifiers like Decision Trees, SVM, and Ensemble and their respective performance metrics (including computation time)Its effectiveness has led many countries around the world to mandate face coverings when in public, however monitoring people for face masks becomes difficult in large groups and gatherings. This paper uses both deep and classical machine learning methods to develop an algorithm for face mask detection contributing SVM, decision trees, and ensemble models that do not overfit the training data. Extracting features from the image dataset in a way that improves overall classification and computation speed is the main challenge for this study and will impact the algorithm’s overall usefulness in real-world applications. A novel aspect of the paper is that the authors chose to use the residual neural network feature extraction method to reduce vanishing gradients from the dataset and improve their classification results. The authors concluded that the deep transfer learning classifier, currently, does not result in acceptable accuracies, and therefore omitted the results for that classifier. Amongst the classical machine learning classifiers used, it is ultimately concluded that the SVM algorithm is the best classifier because not only does it have relatively high testing accuracies across each dataset, but it also consumes the least amount of time. Interestingly, datasets that used simulated masks like the SFMD dataset resulted in lower training accuracies for the decision trees classifier. It was concluded that the SVM classifier was the best classifier since it had a relatively high testing accuracy but consumed the least amount of time to compute. We will refer to this paper as our baseline paper throughout the paper and use the research presented here to evaluate the effectiveness of our own findings in the project. It should be noted that while expansive in the amount of models the paper introduces to solve the problem, it does not include the code for replicating any of the models. Most of the research regarding hyperparameter tuning for our research was done independent of the paper.

Dataset

We will use a mask (class label 2) v. no mask (class label 0) image dataset provided by Kaggle, which are already separated into two folders, mask and no mask. We will combine this dataset with another one provided by Cornell (00000-05000) that provides correct (class label 2) v. incorrect wear of a mask (class label 1).

As for the dataset we used to test bias, we created a W (white) and POC (people of color) dataset to test biases in our models. Since there were no such dataset readily available, we obtained the images for W and POC by scouring the internet.

Dataset Preprocessing

In order to train our models with our dataset, we first had to normalize all the images before feeding them into our models. Normalizing our images meant that all images should be formatted in RBG or grayscale(not both) and all images should have the same shape. For our project, we decided to format our images in grayscale and resize the images 64X64. We felt that the extra information offered by an RBG image would not be necessary as color has no significance in detecting masks, which vary in color. Another reason for choosing grayscale is that processing a grayscale image is three-four times faster than processing an RBG image. As for the size of the image, we decided on a 64X64 because we felt the image was still readable and wouldn't complicate our runtime. The image below is an example of one of our preprocessed images.

We also normalized our W and POC dataset by using RBG images and resizing the images to 64X64.

Evaluation

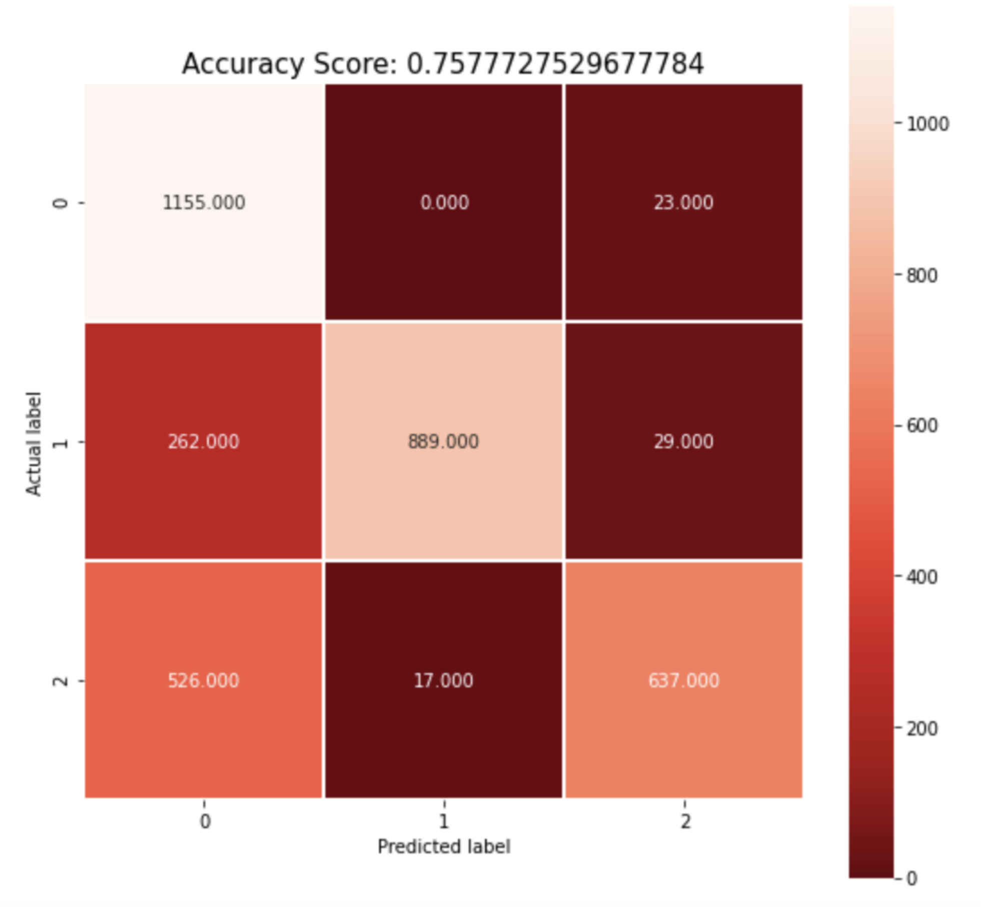
We will use the performance metrics gathered on the SVM, KNN, CNN, Naive Bayes, and Decision Trees Models and compare it to the research done in our baseline paper. It should be noted that while we will use our baseline paper to compare our results, we will be training our models on a different dataset that, unlike our baseline paper, includes the **third classification** of incorrectly wearing a face mask.

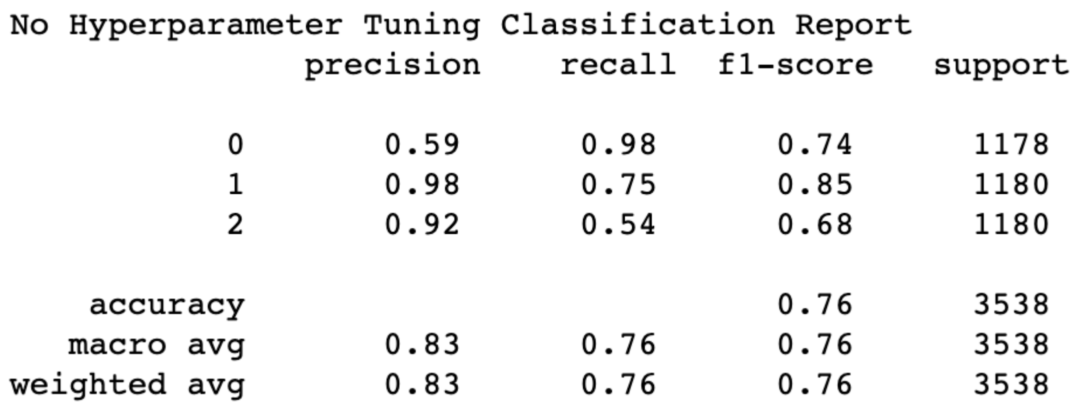
K-Nearest Neighbors

The K-Nearest Neighbors (k-NN) classifier was brought into our study as a standalone model for classifying our face mask dataset. While it was used as part of an Ensemble classifier in our baseline paper, the k-NN model's performance metrics were not specifically given. However, the overall Ensemble when combined with both Linear and Logistic Regression Models achieves up to 100% testing accuracy in a combined Real Face Mask and Simulated Face Mask dataset, referred to as DS3 (Loey, M et al., 2021). We expect there to be some variation between our models due to both differences in datasets used to train and examine our models as well as differences introduced to our data because of the **third classification** of incorrect wear of face masks.

Training k-NN

The k-NN model presented in our research underwent several hyperparameter tuning phases which we will break down here. Before the model was trained, it was normalized so that every pixel contained a value between 0-1, instead of a value between 0-255. The first iteration of the k-NN model developed used the default parameters for a single K-Neighbors Classifier provided by scikit-learn. These parameters are *n_neighbors*= 5, *weights*= "uniform", *p*= "2", *metric*= "minkowski". By default scikit-learn's k-NN classifier automatically decides which k-NN algorithm is the most appropriate to use for computing the nearest neighbors (ex. ball tree, kd tree, brute force k-NN) (Sklearn.neighbors.KNeighborsClassifier, 2020). The same training and testing sets are used to evaluate all of the models, and the splitting nature between the test and train sets is done with a random state of 45 and a test size of 20%. With no



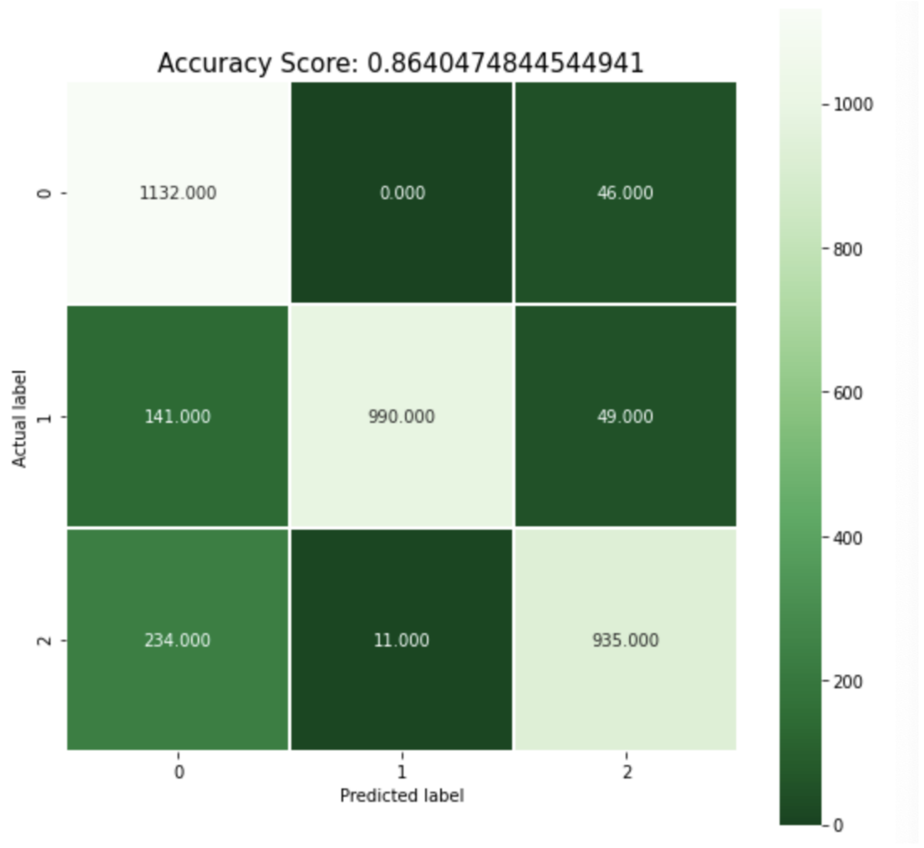

```
No Hyperparameter Tuning Classification Report
              precision    recall  f1-score   support

           0       0.59      0.98      0.74      1178
           1       0.98      0.75      0.85      1180
           2       0.92      0.54      0.68      1180

    accuracy                           0.76      3538
   macro avg       0.83      0.76      0.76      3538
weighted avg       0.83      0.76      0.76      3538
```

hyperparameter tuning applied, the model reaches an accuracy level of ~75.7%. Upon closer examination on the classification report for the model, it achieves 98% precision for classifying incorrect face mask wear samples and 92% precision for classifying correct face mask wear samples. However, the model underperforms significantly, for classifying no mask wear, with a precision of only 59%. This means that our classifier is returning a lot of false positives for classifying the no face mask label, which if applied to real world applications would make it hard to pinpoint high-risk civilians based on face mask coverage for virus transmission. On the other hand, class label 2 (correct wear of face mask) has a comparatively low recall score of 54%, meaning that there are a lot of false negatives. In order to improve our model, the errors caused by misclassification in

Due to the lazy-learning nature of the k-NN classifier, one major drawback of utilizing it came from the cost of the memory intensive and computationally expensive calls for training and classifying a new instance. Thus, the next iteration of the model focused on dimensionality reduction using the Principal Component Analysis (PCA) method to transform the dataset from a high dimensional space of 4,096 features into a low-dimensional space of just 153 features. The PCA model decomposition was set to preserve 90% of the variance in the dataset and prevent underfitting the model. The resulting model resulted in 83.5% testing accuracy and was 11% more precise in labeling the no face mask label.

While this performance was a step in the right direction, the model could be further tuned, and so for the third iteration a GridSearchCV was employed to test a combination of predetermined parameter values for n_neighbors, the distance metric, and p, the power parameter for the minkowski distance metric. The GridSearchCV was also computationally expensive, and took over 15 hours to compute all of the combinations chosen for the parameters. At this iteration of improving the model, the GridSearchCV was not used in conjunction with the Principal Component Analysis transformation. Nevertheless, the GridSearchCV ended up revealing a lot about the data as well as confirming some held expectations. While only performing at best with an accuracy of 82.7% (a bit lower than our results with just PCA), the best performing parameter combination was when n_neighbors was 5 (ironically the default in scikit-learn) and when the distance metric used was manhattan. This is perhaps unsurprising given that in high dimensional datasets, the manhattan distance metric performs better than the euclidean metric for all parameter combinations. This information is used in the fourth iteration of model improvement to limit the amount of parameter values used in testing combinations.

The fourth iteration of model improvement used RandomizedSearchCV instead of a GridSearchCV because it was expected to use less memory and perform faster



with a higher chance of finding an optimal combination of parameters for the model compared to GridSearchCV. The RandomizedSearchCV took around 3 hours to compute, and discovered an optimal parameter solution with 86.7% accuracy. The optimal parameters were *n_neighbors*=2, *weight* =distance, and *distance*=manhattan.

```
Classification Report
              precision    recall  f1-score   support

           0       0.75      0.96      0.84      1178
           1       0.99      0.84      0.91      1180
           2       0.91      0.79      0.85      1180

    accuracy                           0.86      3538
   macro avg       0.88      0.86      0.87      3538
weighted avg       0.88      0.86      0.87      3538
```

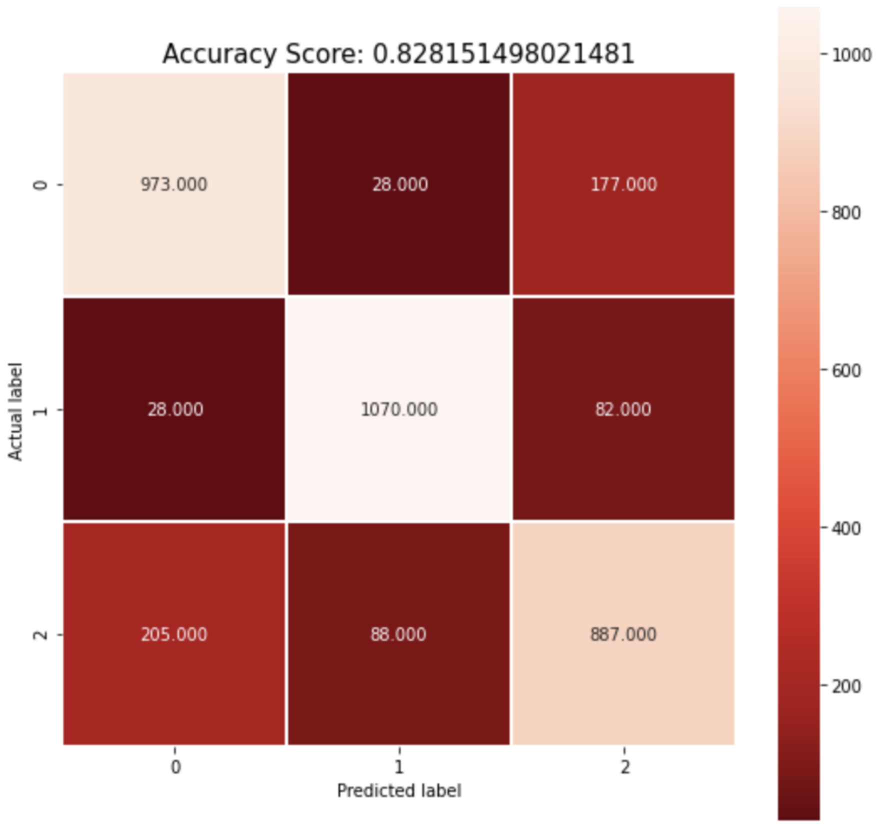
Notice how the class 0 label (no face mask label) has a precision of 75% in this model compared to our original model with no hyperparameter tuning of 59% and the class 2 label (correct wear of face mask label) has a recall of 79% compared to our original 54%. These parameters for the k-NN model resulted in better predictions for the test data, and minimized the error in precision and recall across two class groups. Applying PCA to the RandomizedSearchCV model was attempted in a fifth iteration of parameter tuning, however the model only achieved 83% accuracy, leaving the best parameters for the k-NN model founded in our study resulting from a standalone RandomizedSearchCV.

After some hyperparameter tuning, a k-NN model of 86.7% was achieved. While there is no direct comparison of this model used in our baseline study to effectively evaluate our model, it should be noted that k-NN does not always work well in high dimensional datasets. In this case, we believe that a higher accuracy could still be achieved by minimizing the error for labeling a no face mask and correct mask wear sample. In terms of k-NN, reducing the amount of features used in computational tasks for the model is important because it not only speeds up overall computation time, but it removes features that seem to be irrelevant and noisy. Therefore, attempting dimensionality reduction was useful, even though it didn't result in the most optimal results.

The optimal k-NN model constructed above performed poorly for both bias evaluations, for the People of Color (POC) dataset and the White (W) individuals dataset. The overall performance of the model against these new instances of data reveals that our model is likely either overfitting for the training dataset or that our datasets are malformed to some degree. The POC dataset had a testing accuracy of 50.2%, and the W dataset had a testing accuracy of 34%. In regards to the 16% discrepancy between the POC dataset's performance, and W dataset's performance, an important question arises from the integrity of our training dataset and its ability to fully represent the people it is likely to affect. Further examination is required to quantify the differences between each dataset, but the performance metrics alone suggests that white individuals were not as properly represented in our dataset as people of color were. It is also possible that differences in image perspective played a key role in the discrepancies between the accuracies of the two datasets, where our training images were more “zoomed-in”.

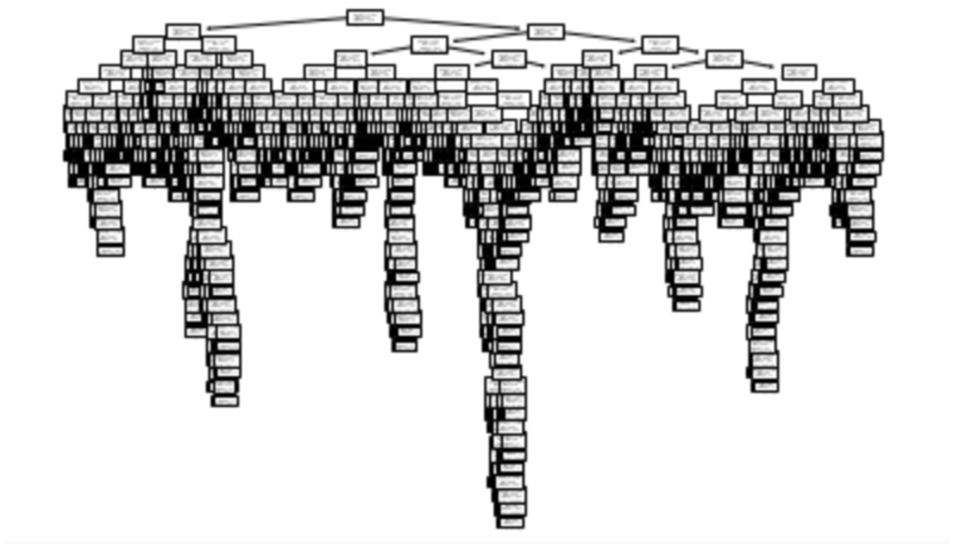
Decision Trees

The Decision Trees classifier is introduced to our study as another non-parametric model and can be directly evaluated with the performance metrics for the Decision Trees model presented in our baseline paper. While both models can be directly compared and evaluated, it should be noted that discrepancies in predictions are expected due to differences in training datasets. Similar to the k-NN model, the Decision Trees model went through a few iterations of improvement in which we will break down here. Before the model was trained, the pixels were normalized to a range of 0-1 from a range of 0-255, and split with a test size of 20% and a random state of 45. The first iteration of the decision trees model developed used the default parameters for a single DecisionTreeClassifier() provided by scikit-learn. A few highlights from that parameter set includes the *criterion*= gini (for Gini impurity), *splitter*= “best”, *p*= “2”, *max_depth*= “None”, *min_samples_split*=2, *min_samples_leaf*=1, *max_features= None*. Note that the *criterion* parameter was not changed during hyperparameter tuning from Gini Impurity, which determines the probability of misclassifying an observation,

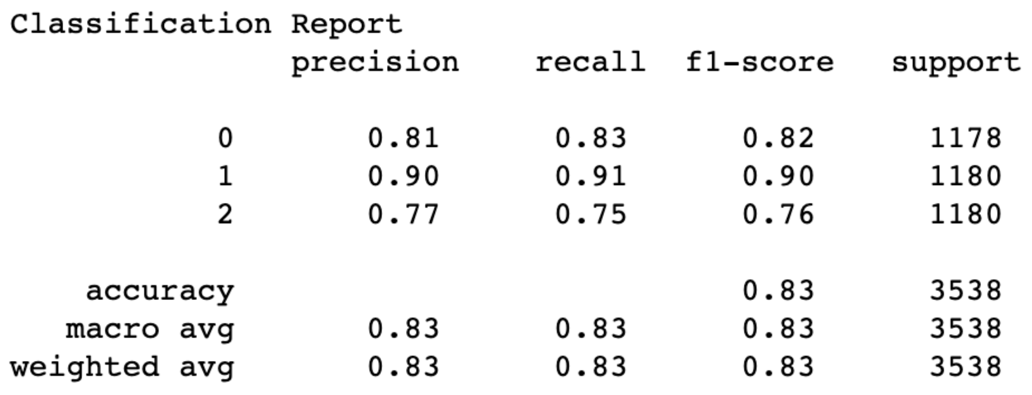


because computing the logarithmic functions for the alternative, Entropy, would be computationally expensive.

Without any hyperparameter tuning, the decision trees model had an accuracy score of 82.8% and better overall precisions for each of the classification labels in comparison to the k-NN model. Moreover, the decision trees classifier had a faster computation time of around 3-7 minutes for fitting and predicting a single model. Nevertheless, the second iteration of the decision trees model explored dimensionality reduction in the hopes of discarding unimportant features that may negatively affect the



model.

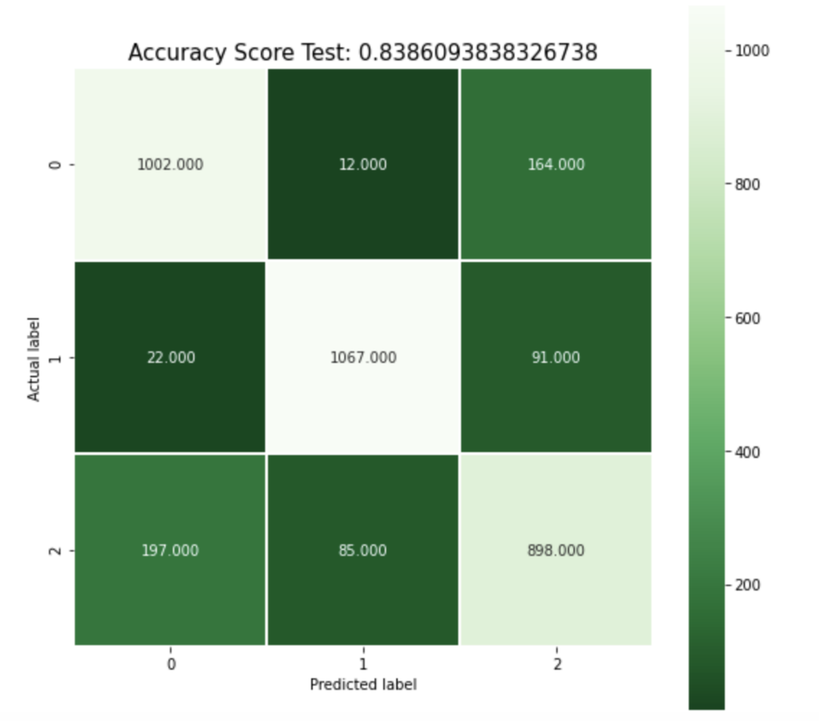


```
Classification Report
              precision    recall  f1-score   support

           0       0.81      0.83      0.82      1178
           1       0.90      0.91      0.90      1180
           2       0.77      0.75      0.76      1180

    accuracy                           0.83      3538
   macro avg       0.83      0.83      0.83      3538
weighted avg       0.83      0.83      0.83      3538
```

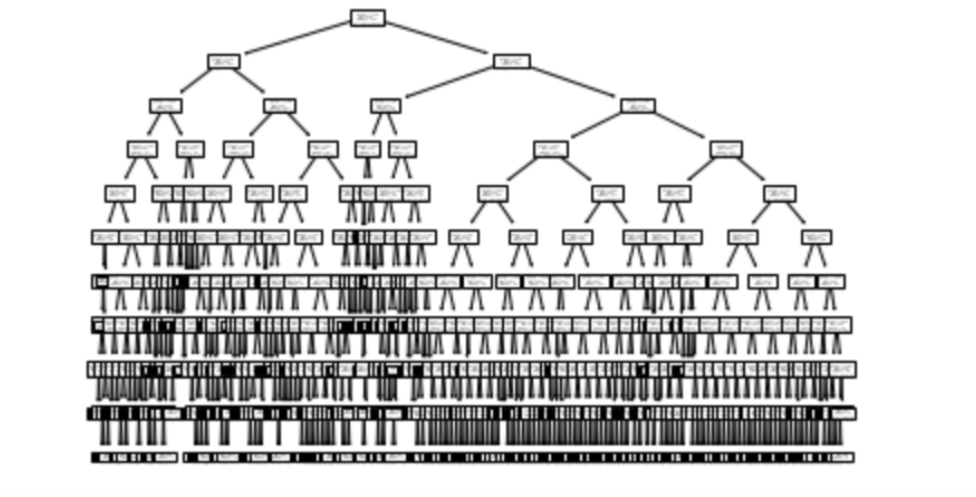
Principal Component Analysis was applied with preserving 90% of the variance in our training set. Again, the feature size was reduced to 153 features compared to its previous value of 4,096 features. Interestingly, however, with PCA applied, the model performed with 75.6% accuracy, a drop of ~8% from our decision tree model without any dimensionality reduction applied. Changing the PCA variance preservation from 90% to 99% and to 80% also did not improve the model's accuracy. Since reducing features through PCA seemed to have a negative effect, more focus in the next iterations was placed on finding optimal parameters for the model.

The parameters chosen for focus during the hyperparameter tuning phase included the *max-depth* attribute, the *minimum samples splits* attribute, the *minimum samples leaf* attribute, and the *max features* attribute. The max-depth attribute determines the depth of a tree. The deeper the tree, the more splits/information is garnered about the data, and consequently, the likelier the chances of the model overfitting. The minimum samples splits refers to the minimum number of samples required to split a node. If increased, a tree considers more samples at each decision node and is likely to underfit the model. The minimum samples leaf parameter refers to the minimum number of samples required for each leaf node, which will also underfit if increased. Finally, the max features parameter refers to the number of features to consider when searching for a best fit, and will overfit the model with more features.

Each parameter underwent testing to see which values performed best when fitted with the training dataset. The training accuracy was used to measure how much the model was overfitting in comparison to the testing accuracy. It was concluded that the best parameters for the model was when *max_features*= 1,138 and a *max_depth*=10. In the end, the *min_samples_split* and the *min_samples_leaf* parameters were not used in the model because all of the value variations for these



```
Hyperparameter Tuning Classification Report
              precision    recall  f1-score   support

           0       0.82      0.85      0.84      1178
           1       0.92      0.90      0.91      1180
           2       0.78      0.76      0.77      1180

    accuracy                           0.84      3538
   macro avg       0.84      0.84      0.84      3538
weighted avg       0.84      0.84      0.84      3538
```



parameters resulted in poor model performances. Since both of these parameters have high bias and are likely to underfit, it may be at no surprise that they underperformed. The resulting model accuracy with the combined parameters chosen was ~84%, slightly higher than the model without hyperparameter tuning. Notice that similar to the k-NN model, the class label 2 (correct wear of face mask) performs lower than the other categories. Again this may arise from inconsistencies introduced to our dataset with simulated and real face mask samples. The above method for finding the ideal parameter values was used in conjunction with PCA in an attempt to improve it, however the resulting PCA model with the best parameters achieved only 77.4% accuracy.

In comparison to our baseline study which achieved up to 99.89% testing accuracy for the LFW data, our model falls short by around 15% for meeting comparable standards in accuracy. It should be noted, however that the baseline paper uses four different datasets to train and test their models on. The lowest performance achieved for their decision trees model came from its performance over the SMFD (Simulated Face Masks Dataset), achieving at highest, 95.64% testing accuracy (Loey, M et al., 2021). This dataset is significant because like our dataset for detecting the incorrect face mask wear category, it uses simulated or fake photoshopped images of face masks on individuals to satisfy the mask wearing condition. It is probable that issues arising in both results stem from the same pitfalls in training data that uses this simulated pixel methodology. That being said, the baseline decision trees classifier still outperforms our model on a similar dataset by 11% in terms of accuracy.

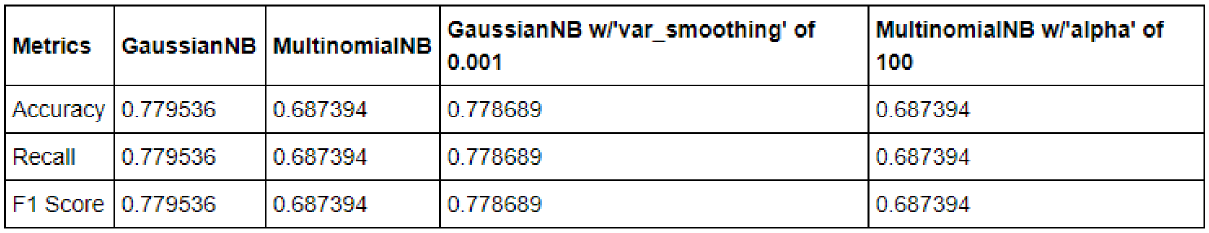
The optimal Decision Trees model constructed above performed poorly for both bias evaluations on the People of Color dataset (with 55.6% testing accuracy) and the White individuals dataset (with ~31.4% testing accuracy), but slightly better than the *optimal* k-NN model. In general terms, the overall performance of the model against these new instances of data reveals that our model is likely overfitting for the training dataset. It also might be likely that the images in which we trained our data on are distinctively different from the images from our bias datasets collection, and we might have to reconsider our approaches for overall feature selection, feature relevancy and data preprocessing measures. In regards to the extreme ~24% discrepancy between the POC dataset's performance, and the W dataset's performance, an important question arises from the integrity of our training dataset. Further examination is required to measure the differences between each dataset in comparison to our training dataset, but the performance metrics alone suggests that white individuals were not as properly represented in our dataset as people of color were.

Naive Bayes

Naive Bayes is another algorithm that we implemented into our project. The reason why we decided to include this algorithm was because it was a model that was not used in our baseline reference paper and it's a powerful algorithm that is still used to detect spam in our emails. Before creating this model, we made a prediction that this type of algorithm would not perform well because the naive bayes has the assumption that all features are conditionally independent. When it comes to image classification of any image, the majority of the pixels in the image are correlated to each other; however because the naive bayes algorithm works surprisingly well in natural language processing even when it’s assumption doesn’t hold, we decide to test it out. We used sklearn’s Gaussian and Multinomial naive bayes, with and without hypertuning.

After normalizing the dataset by dividing all the pixels by 255, we split 20% of the data to testing and the rest to training with a random state of 10. We create a GaussianNB model without hyperparameter tuning. Next we fit the training data along with its corresponding labels and test our testing data. Repeat the steps for MultinomialNB. Next we move on to hyperparameter tuning for both models. For GaussianNB it’s hyperparameter is “var_smoothing” and Multinomial is “alpha.” Create a GridSearchCV for GaussianNB with params set to “ 'var_smoothing' : [1e-09, 1e-07, 1e-05, 1e-03, 1, 100]” and for MultinomialNB’s params to “ 'alpha' : [0.1, 0.5, 1, 2, 5, 100].” Set the models with the optimal params and rerun the evaluation test on them.

Results from these models are:

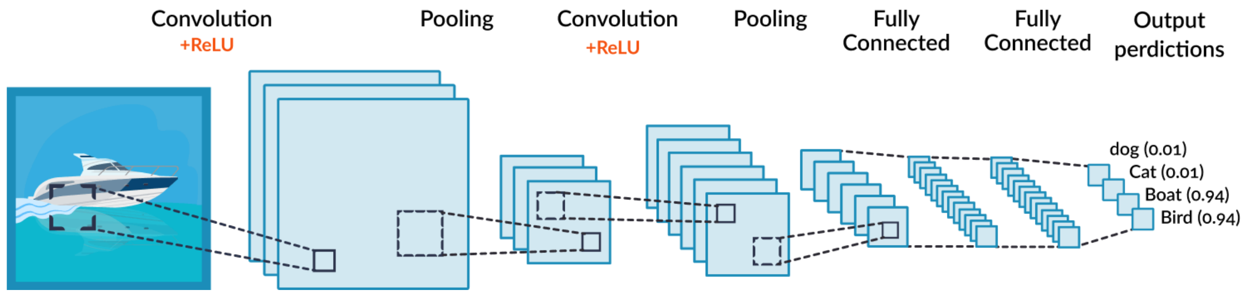


Metrics	**GaussianNB**	**MultinomialNB**	**GaussianNB w/'var_smoothing' of 0.001**	**MultinomialNB w/'alpha' of 100**
Accuracy	0.779536	0.687394	0.778689	0.687394
Recall	0.779536	0.687394	0.778689	0.687394
F1 Score	0.779536	0.687394	0.778689	0.687394

The numbers for the hyperparameter tuned models actually perform the same or slightly worse. Even without hyperparameter tuning, the GaussianNB reached only ~78% accuracy, recall and f1 score; and the MultinomialNB reached ~68% in all three metrics.

In conclusion, naive bayes doesn’t perform well with images because of its structure as an algorithm. The naive assumption doesn’t hold in this case and hurts the classification. With these accuracies, we decided not to use this algorithm to test for any bias in our model or data because its performance was already relatively low. Also from our results, we can understand why the baseline researchers chose to exempt this algorithm in their studies.

Convolutional Neural Network (CNN)



Convolutional neural networks are very popular in computer vision tasks such as identifying if there is a mask in the image. A brief summary of CNN: it is a neural net with additional layers in the front to extract certain features in the image. Those additional layers are the convolution and pooling layers that distinguish unique features that an image should have.

Unlike our baseline research paper uses ResNet to extract features and uses that to train their models, CNN learns and extracts its own features that are important in our classification. To build the model, we will be using Tensorflow's Keras and Google Colab's GPU because it speeds up training. When on Google Colab, make sure to set the runtime to GPU and we can start.

First applying all exploratory data analysis, normalize all the pixels in the dataset to be in the range of 0 and 1 so our model has a faster time converging. A problem that occurs if you have a big dataset when you divide by 255 is there will be a huge usage of RAM because of floating points; make sure you use batch training to limit the space it takes up. Another important thing to note about Keras is the input (i.e. the image) has to be formed in a dimension of (width x length x dimension); where dimension is the number of channels, (i.e. 1 if the image is in grayscale and 3 if it is in color or rgb.)

Since we are training on 64 x 64 grayscale, make sure to resize the data to 64 x 64 using Cv2 or Pillow and then use numpy to reshape the dimension (X = X.reshape(-1, 64, 64, 1) .)

We will be using 2 convolutional layers and 2 max-pooling layers. First creating the Sequential model and we will be adding layers into it. A CNN has convolution and max-pooling layers in the front to extract features; so we will be adding a total of 6 layers: 3 conv2d layers and 3 max-pool.

Sequence of layers:

1. Conv2d layer has the parameters: filters=8, kernel_size=3, activation='relu'
2. Max-pooling
3. Conv2d parmas: filters=16, kernel_size=3, activation='relu'
4. Max-pooling
5. Conv2d params: filters=32, kernel_size=3, activation='relu'
6. Max-pooling

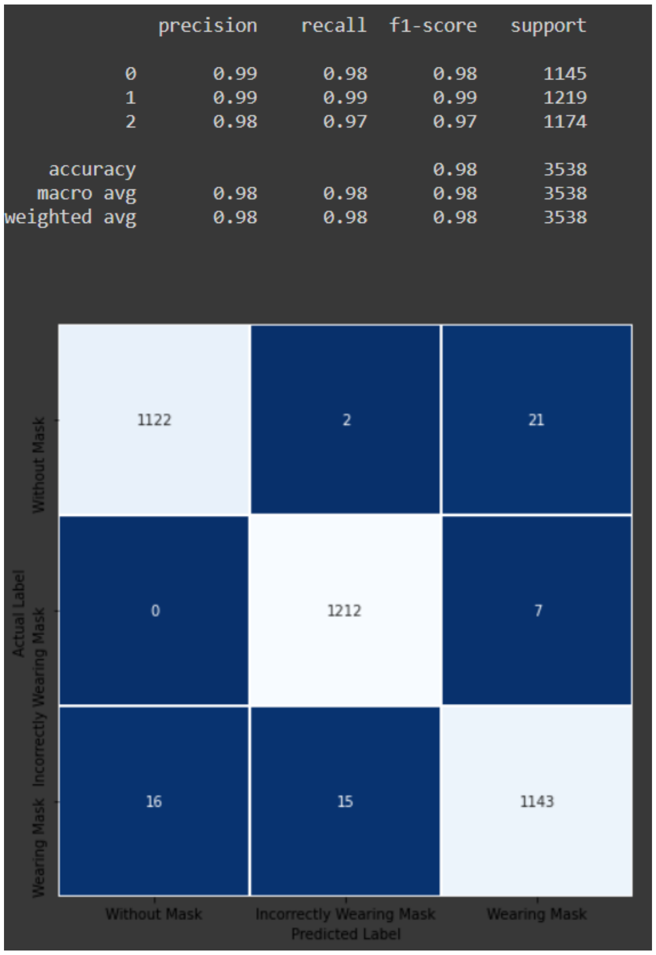
We are finished with the convolutional part and now we will start the neural network

7. We flatten the filters into one dimension using Flatten
8. Followed by Dense layers which are the hidden layers, params: 512, activation='relu'
9. Another Dense: 512, activation='relu'
10. Dropout layer: 0.3
11. Dense: 256, activation='relu'
12. Dropout layer: 0.3
13. Dense: 128, activation='relu'

14. Dense: 3, activation='softmax'

The reasons for lower filter numbers in the convolution layers and the dropout layers are for the model to be less prone to overfitting therefore work better for general classification. We don't want our model to overfit our data because then we can't use it to make predictions of other images the model has never seen before.

Finally we compile the model together by using compile, and we do the training/fitting method similar to other models.

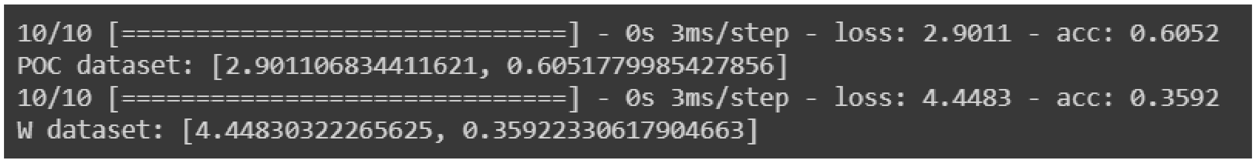


```
              precision    recall  f1-score   support

           0       0.99      0.98      0.98      1145
           1       0.99      0.99      0.99      1219
           2       0.98      0.97      0.97      1174

    accuracy                           0.98      3538
   macro avg       0.98      0.98      0.98      3538
weighted avg       0.98      0.98      0.98      3538
```

As we evaluate the model, we see the model performs extremely well on our dataset. We reach around ~98% accuracy on our testing data which is awesome. Next we will proceed to evaluating our model and dataset if there are any biases.

We have two datasets prepared, one is only of people of color and another is people who are white. We still have to do the same EDA on these datasets to run it against our model. After we finish, we evaluate these new test samples that the model has never seen before.

From these new results on the people of color and white datasets, we obtain very interesting observations.



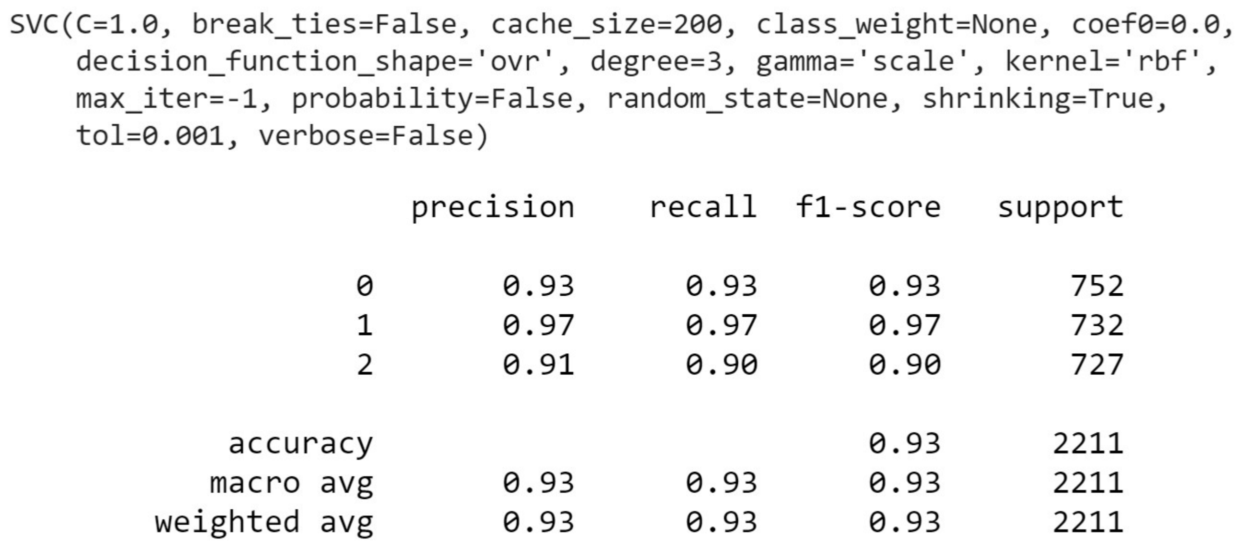
```
10/10 [==============================] - 0s 3ms/step - loss: 2.9011 - acc: 0.6052
POC dataset: [2.901106834411621, 0.6051779985427856]
10/10 [==============================] - 0s 3ms/step - loss: 4.4483 - acc: 0.3592
W dataset: [4.44830322265625, 0.35922330617904663]
```

The people of color dataset has around 60% accuracy while the white dataset has around ~36% accuracy. Although just based off of this information is not enough to clarify, we still need more evidence but so far it seems like our dataset that we used to train our model is very diverse. Having a diverse dataset implies that we are not discriminating against a certain group, however we still need to run more analysis to be over 100% confident that no one is getting harmed by artificial intelligence that uses our models.

Some precautions to be aware of: why is our model working better for the people of color dataset better than the white ones, is it because we have some copies of the original dataset in the people or color? Is our white consistent with our training data where people are wearing similar masks or are they wearing masks but it's made out of a different material and our model is not picking that up? These are just some questions that can help clarify and make our model, research, and arguments stronger by having these answers.

SVM Results

SVM models are great for image classification and great for small to medium sized datasets, which is why we are exploring SVMs. Since computation time was a concern, we took advantage of the fact that SVMs are great for classifying small to medium sized datasets. Instead of using all seventeen thousand samples, we used fifty



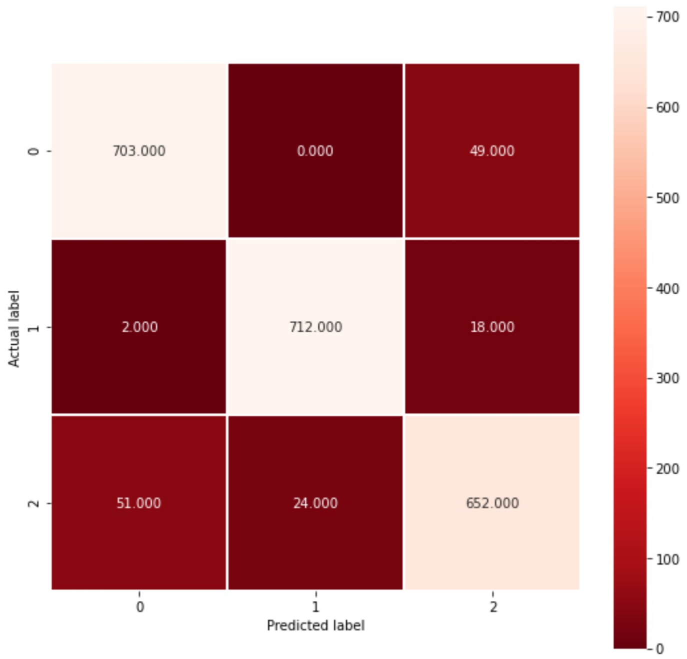
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)

              precision    recall  f1-score   support

           0       0.93      0.93      0.93       752
           1       0.97      0.97      0.97       732
           2       0.91      0.90      0.90       727

    accuracy                           0.93      2211
   macro avg       0.93      0.93      0.93      2211
weighted avg       0.93      0.93      0.93      2211
```

percent of the samples to train our SVM model. At first the SVM model was trained with its default parameters, which are shown below.

The default parameters resulted in favorable metric values. The default SVM was chosen over the hyperparameter tuning SVM(‘C’: 100,’gamma’:0.000001) as it received poorer results when testing against other datasets.

Choices/Experiments

Overall, class 1(Incorrectly Worn Mask) has the best metrics. This could be explained by the fact that all the images in that class are photoshopped with the same type of mask over someone’s face. Perhaps the consistency in displaying the same photoshopped mask in that class led to class 1’s favorable metrics.



When our SVM model was tested against the W and POC dataset the results suggested an overfit. The POC metrics were higher than the W metrics. The POC resulted in a 58% accuracy, while the W resulted in a 36% accuracy.

Compared to the baseline, which had over ninety-nine percent accuracy, our SVM model falls short by around six percent. It's worth noting that we are not using the same dataset as the baseline paper. Therefore, we are unable to train our model with the same dataset as the research paper.

Results

To sum up, our CNN model performed the best with a testing accuracy of ~98%. Following is our SVM model with ~93% accuracy, the KNN model with ~86.7% accuracy, the Decision Trees model with ~ 84% accuracy and Gaussian Naive Bayes with 77.8% accuracy. The best classification overall for a single class category goes to the incorrectly worn face mask category that consistently performed with the highest accuracies compared to the other two classes. While our results for the SVM, KNN and Decision Tree models underperform when compared to our baseline paper, it should be noted that discrepancies in our dataset may be the cause. Not only did we realize that the zoomed in perspective of the images in which our models were trained on negatively affected the way our models handled new unseen instances, but the simulated/photoshopped images of face masks on individuals for the incorrectly worn face mask category likely affected our feature set across instances. This could help explain the extremely low performances of our models when introduced to our People of Color, POC dataset, with ~50%-~60% testing accuracy, and our White individuals, W dataset with ~30% testing accuracy across models. Possible overfitting trends across our models can also be to blame for the poor performances. It should be especially noted that while our algorithms performed poorly on both groups of people, they all collectively performed worse for the W dataset compared to the POC dataset. This suggests some unfair bias against White individuals in our training data compared to people of Color. That being said, this discrepancy can also arise from a number of reasons including watermarks present in our training images themselves compared to the W and POC datasets, image perspective differences from our training set compared to the new instances and discrepancies mentioned above with simulated face masks. Further examination of our datasets is needed to make a comprehensive evaluation on why this performance is so low, and, more importantly, how we can take steps to counteract it.

Conclusion

Through experimenting with the non-parametric models for k-NN and decision trees, the importance of dimensionality reduction and computational expense was highlighted. The images had 4,096 features and when applied to memory intensive classifiers like k-NN, the model ends up eating a lot of computer resources to complete the computation. By reducing unnecessary features we can not only improve the runtime, but we can also potentially increase the accuracy of our model by removing irrelevant features. We also learned the importance of data preprocessing and dataset selection. It is likely that discrepancies in our datasets and image quality such as zoomed in images and simulated/photoshopped face masks on faces led to errors in training our models and hindered their ability to predict unseen instances that varied in these aspects. We also learned that SVM is prone to overfitting and similar to k-NN it is difficult to find the optimal parameters when computation time is so expensive. Finding a method to reduce computation time was important. We also struggled in tuning the SVM RBF kernel, which since our classes overlapped and visualizing the decision boundary was difficult, would have been ideal to apply for this application. Other problems involved getting the input of the image to work with Keras, and saving a Keras model for the CNN implementation. Goals that were not achieved was using RCNN to help classify parts of the image. Overall we learned how important features were in making a good classifier, the ease of overfitting and underfitting models and how to avoid doing so, and the importance of data and how that affects biases in models.

In conclusion, while we had high performing models like CNN and SVM, all of our models performed poorly for classifying new instances that were introduced in our bias examination phase of our model. Further examination on the drawbacks of our training datasets as well as the configuration of our models is needed to improve our accuracies across groups of people as well as new training instances. It is also important to explore possible overfitting across our models to the training dataset. While both the POC dataset and the W dataset performed poorly, it should be noted that the W dataset performed around ~10%-25% worse than the POC dataset across all models. This discrepancy alone suggests an unfair bias against White individuals in comparison to People of Color. However, further examination of image perspective and image malformations should be further examined in both datasets before making this conclusion. What it means to correctly wear a face mask is a category that continues to change and evolve as more research is done. While our models cater for this third category, it does not necessarily reflect the evolving standards of mask wearing today, and will likely misclassify instances of incorrect mask wearing. It is something to keep in mind when regarding this study as well as when choosing an ideal dataset to reflect the nuances presented in correct mask wear.

References

BBC News. (2020, May 04). Coronavirus France: Cameras to monitor masks and social distancing. Retrieved December 13, 2020, from https://www.bbc.com/news/world-europe-52529981

Cabani, Adnane and Hammoudi, Karim and Benhabiles, Halim and Melkemi, Mahmoud, (2020 Nov). MaskedFace-Net - A dataset of correctly/incorrectly masked face images in the context of COVID-19,
Retrieved October 20, 2020 from
https://arxiv.org/abs/2008.08016

Jangra, Ashish, (2020 May). Face Mask ~12K Images Dataset, Version 1.
Retrieved October 20, 2020 from
https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset/version/1

Loey, M., Manogaran, G., Taha, M., & Khalifa, N. (2021). A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic. *Measurement : journal of the International Measurement Confederation*, *167*, 108288. https://doi.org/10.1016/j.measurement.2020.108288

Maragakis, L. L., M.D., M.P.H. (2020, September 11). How to Properly Wear a Face Mask: Infographic. Retrieved December 13, 2020, from https://www.hopkinsmedicine.org/health/conditions-and-diseases/coronavirus/proper-mask-wearing-coronavirus-prevention-infographic

Sklearn.neighbors.KNeighborsClassifier¶. (n.d.). Retrieved December 15, 2020, from

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

World Health Organization. (2020, December 01). When and how to use masks. Retrieved December 13, 2020, from 2020, from https://www.who.int/emergencies/diseases/novel-coronavirus-2019/advice-for-public/when-and-how-to-use-masks