Facial Expression Detection with Limited Features

CPE-695 Team 5: Final Report

1st Allison Aprile
Stevens Institute of Technology
Jersey City, NJ, United States
aaprile@stevens.edu

2nd Juan Cristobal Amezquita Martinez de Alva
Stevens Institute of Technology
Miami, FL, United States
jamezqui@stevens.edu

3rd Zachary Schaber
Stevens Institute of Technology
Hoboken, NJ, United States
zschaber@stevens.edu

Abstract—**Facial expression detection is a prominent computer vision task. Unlike prior research, we propose a model that can detect expression based on a reduced subset of features; namely, the top half of the face. By eliminating the mouth from the input, we are not only challenging the original problem, but also working towards a topical model. With the increased use of facial coverings, the COVID-19 pandemic has instituted this angle of facial expression detection into reality. The intention of this project is to investigate the effectiveness of three emotion classification algorithms on inputs missing the mouth facial feature - that is, subjects wearing a face mask. Specifically, we work with i) Convolutional Neural Networks; ii) Bayesian Networks; and iii) Support Vector Machines.**

Index Terms—**Computer Vision, Facial Expression Detection, Convolutional Neural Networks, Bayesian Methods, Support Vector Machines, Emotion Detection, Face Masks**

I. Introduction

Communication is a complex system of processes that connects every member of societies all around the globe. Too often, however, communication is simplified to the words we exchange and the actions that we take. In reality, there are numerous subtleties that comprise the communicative capacity of a population. In the wake of COVID-19, in a society rightfully obsessed with hygiene and, specifically, masks, one specific subtlety of communication that is restricted is full facial expression. When faces are obscured from the eyes down, a large portion of the expressive communication is restricted, especially in pictures where other body language and verbal communication is not present.

II. Problem Description

This project aims to solve this problem of inhibited communication via classification of facial expression with limited features. With protective masks covering the portion of the face below the nose, this must be done using only the features above this point on the face. By developing a multi-class facial expression classifier to detect emotions through facial expressions even when a mask is present, this project aims to classify still facial expressions by emotion. This will be done by simulating the presence of a mask in a data-set of faces through cropping.

III. Related Work

A. Facial Emotion Recognition - Machine Learning

[RautRaut2018] explored the methodology necessary for emotion recognition by machine learning. It explores the general emotion detection steps: dataset preprocessing, face detection, feature extraction, and classification based on features. This Thesis employs Facial Action Coding System to assign each facial movement a number referred to as an action unit. Combinations of these action units result in facial expressions. This paper also defines 68 facial landmarks via the DLib library to detect expressions. Extracted facial features, in conjunction with SVM, were ultimately used to predict the emotions using “One-Vs-Rest” algorithms. This paper ultimately found a range of 66-100 percent accuracy across 7 different emotions and 82 different samples.

This paper is related to this project due to the nature of its final goal. It, like this project, seeks to classify facial expressions within emotional categories. However, a key difference is that this thesis had additional facial features to extract in the form of mouth movement.

B. Masked Facial Expression Detection - Human Standard

[CarbonCarbon2020] discusses a psychology experiment where people were asked to assign emotional labels to people they observed. The sample population was comprised of 41 participants of mean age 26.7 years. Emotional labels were ‘angry’, ‘disgusted’, ‘fearful’, ‘happy’, ‘neutral’, and ‘sad’. The key relation to our project is that half of the observed individuals were wearing masks, while the other half were not. The key notable result from this paper was that, compared to identification of facial emotion without masks, correct identification of facial emotions with masks was, on average, much worse. Without masks, performance saw a mean correct identification percentage of 89.5 percent, with no participant performing below an overall rate of 76.4 percent. With masks on however, the mean correct identification percentage was significantly lower when identifying emotions ‘angry’, ‘disgusted’, ‘happy’, and ‘sad’. These results serve to validate this project’s problem assessment and seem to indicate that any meaningful results generated will be of use in identification of facial emotion.

C. New approach using Bayesian Network to improve content based image classification systems

[Jayech and MahjoubJayech and Mahjoub2010] examined how Bayesian methods can be applied for facial recognition. They combined recent work to build a multi-part algorithm that culminated in using variations of a Bayesian network to make the final classification decision. Images where divided into a grid of blocks, each block had a vector of metrics associated with its color and roughness calculated, those vectors were clustered according to k-means, and the final predictions were then made according to different Bayesian networks which were: Naive Bayes, Tree Augmented Naive Bayes, and Forest Augmented Naive Bayes. The best performing algorithms out of sample were the Naive Bayes and a variation of the Tree-Augmented Naive Bayes with mean accuracies of 0.62 and 0.64, respectively.

IV. Problem Solutions

A. Challenges Faced

While this project was an enriching and stimulating challenge in and of itself, there were a number of roadblocks that further delayed the groups progress throughout this semester. Obtaining sufficient data proved difficult, as the team faced delays and rejection from the source of the original dataset they had intended on using. There were also issues in facial feature recognition and proper cropping of the images during data preprocessing. Additionally, due to the circumstances of this semester, the team experienced difficulty sharing data and collaborating due to distance and remote operation. Finally, high computational cost and long training times led to stalls in the team's productivity. Nonetheless, all of our models performed better than our baseline of 0.1429 (equivalent to random guessing).

B. Description of Dataset

To train and test the models, we required labelled images of faces displaying several common emotions. The images needed to span only the face from the approximate median of the nose to the top of the head, thereby excluding any background noise or additional body parts, especially the mouth. These requirements would simulate the subject wearing a face mask.

The FER-2013 dataset was the best choice for our problem, only requiring cropping. The data includes grayscale face images of shape (48, 48, 3). The faces are approximately centered and occupy a nearly equal amount pixels in the available space. Additionally, the images are organized by folders named with their respective emotion category, encoded as: (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). The dataset includes train.csv and test.csv, containing two columns: "emotion" (the numeric label) and "pixels" (the string representation of space-separated pixel values in row-major order). However, we did not require these files and, instead, created our own NumPy array files for easier data transfer (see C. Data Preprocessing). The data is also pre-split into training and testing sets. The training set has 28,709 images and the testing set has 7,178 images, summing to a total of 35,887 examples.

The datasets can be considered unbalanced, due to unequal quantities of each class available. As seen in Figure 1, Class 1 ('Disgust') has a significantly lower number examples in all of the sets, while Class 3 ('Happy') has significantly more. Due to time constraints, we were not able to address this (besides with standard data augmentation), but we discuss potential solutions later in this paper (see VI. Future Improvements). Due to this imbalance, we hypothesize that the resulting models will be biased and overly predict Class 3, and infrequently predict Class 1.

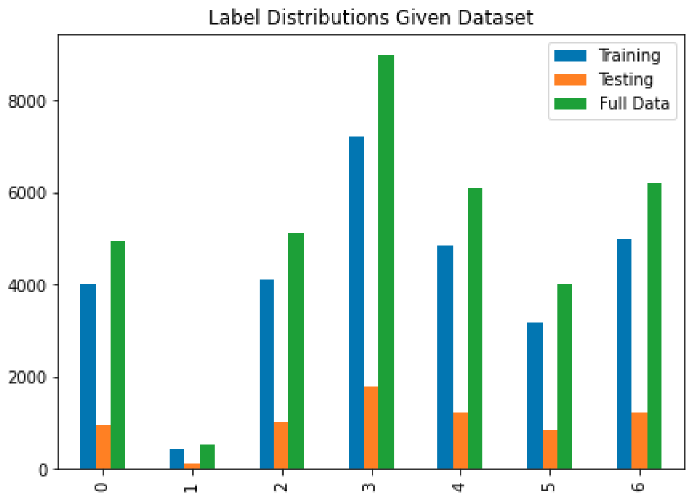


Fig. 1. Label Distribution by Dataset

C. Data Preprocessing

Our solutions are to be implemented only considering facial features from the median of the nose to the top of the head. Therefore, it was necessary to crop the portion of the images from the median of the nose to the chin, as well as any other noise. We considered overlaying a pixel mask (i.e. replacing the bottom section of the images with black pixels), as well as a face mask clear background image, but this would be too computationally expensive for the size of our dataset and given the time limit. Also, the latter mask approach could possibly introduce more noise into the image, especially if the model learns it as a significant feature. From our experiences, we analyze the eyes, eyebrows, and forehead (as well as contextual clues, which were not included) of people wearing face masks to decide their emotions. To translate that into the data, we decided to simply crop the images from the median of the nose and reshape it having the average height.

To accomplish this, we used the Computer Vision and Dlib libraries, which provide machine learning toolkits. Computer Vision (cv2) was helpful for reading and converting the color types of each image, while Dlib provided useful face and facial feature detectors. For each image in the FER-2013 dataset, we set the bounding box to match the image dimensions. Then,

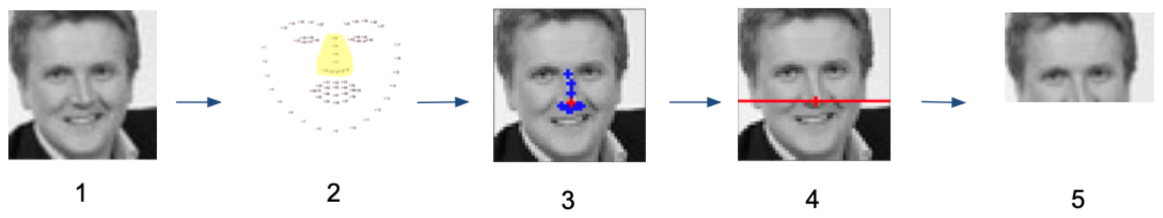


Fig. 2. Preprocessing Steps: 1) Input grayscale image of shape (48, 48); 2) Call Dlib facial landmark detector; 3) Detect nose, identifying Point 31 for the lower part of the nose; 4) Save y-coordinate of Point 31; 5) Crop the image at Point 31 and reshape to (29, 48).

using the bounded contents as a focus point, we called the facial feature landmark function. Based on Figure 2, we are interested in Points 28-36. After some discussion and testing images, we concluded that Point 31 was the optimal cropping height; it best simulated where the top of a face mask would lie, given that the person is wearing the mask as recommended by guidelines (i.e. covering the mouth and nostrils). It also would allow for leeway in case of any rotation or shadowing in the images. To crop the images, we saved the y-coordinate of Point 31 and cropped it. Before finalizing the dataset, we took the average of these y-coordinates and resized each image to have that as their height. The final images have a 29 pixel height and a 48 pixel width. This process is visualized in Figures 2-4.

For easier data transfer, we created NumPy array representations of the datasets. We did this by reading in each image with the second argument as 0 (a grayscale flag), and then vertically stacking the arrays. We also vertically stacked the labels, creating two versions - one being one-dimensional, and the other being encoded as seven-dimensional. This concluded our preprocessing steps.

D. Additional Data

We will train and test the models on this resulting dataset. It has already been presplit into training and testing, the cropped images following the exact same structure and sorting as the original dataset. Later, we create and test a pipeline that can handle an image with one or multiple people, with or without face masks. To make the predictions, a copy of the original image will be cropped using the same preprocessing techniques. The testing data was sourced with manual Google Image searches, consisting of eight images. These images differ in the number of people and the existence of the face mask to evaluate the model on images more complicated and realistic than those in the original FER-2013 dataset. All of these images and their results can be viewed in the Appendix.

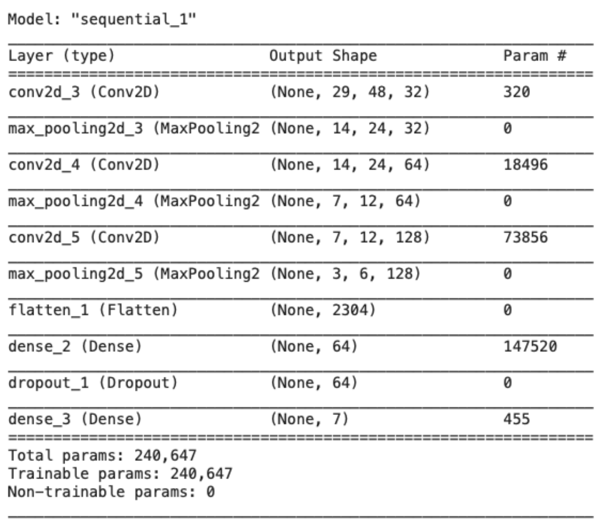
Also, the finalized Convolutional Neural Network model (following) was trained on augmented training data. Data augmentation involves creating a new, more diverse dataset from the input data, through adjustments to rotation, focus, and more. It helps with dataset size as well as class imbalance, a problem present in our dataset. We augmented our training data using a Image Data Generator, specifying the rotation range to 8, and the width shift range, shear range, height shift range and zoom range to 0.08.

E. Convolutional Neural Network

1) Overview: Convolutional Neural Networks, or CNNs, are at the forefront of computer vision applications. CNNs are feedfoward neural networks consist of two main parts: convolutional blocks for feature extraction, and a fully-connected multi-layer perceptron for the decision process. The convolutional blocks consist of alternating convolutional and pooling layers, the first of which creating feature maps, and the second reducing the dimensionality of said maps. The multi-layer perceptron is constructed with a number of hidden dense layers, the final output layer having a number of neurons equal to the number of classes. The predictions are decided by an activation function, typically sigmoid or softmax. Dropout layers and regularizers can also be included to prevent overfitting. We believed a CNN model would be useful due to its memorization capabilities, compatibility with media inputs, and its easy, customizable implementation [SharmaSharma2017].

2) Implementation: We designed three neural networks from scratch, running each for 100 epochs to see the initial results without fine-tuning. These models differed in their depth; i.e. number of layers. The first was a shallow network, consisting of one convolutional block and two dense hidden layers in the multi-layer perceptron, having no dropout. This trained quickly but only performed slightly greater than the .1429 baseline, determined by the random guess probability $\frac{1}{7}$. The second was a deep network, constructed with five convolutional blocks and six dense layers in the MLP, having 0.4 and 0.2 dropout rates after the second and fourth layers. The dense layers had 512, 256, 128, 64, 32 and 7 neurons per layer, respectively. Despite being promising, it took 26 hours to complete training and barely surpassed the performance of the moderately deep network, which was the third and selected model. It is comprised of three convolutional blocks and two dense layers in the MLP, having a 0.2 dropout rate after the first dense layer.

As mentioned above, the selected model architecture consists of three convolutional blocks, followed by a multi-layer perceptron. Each convolutional block is composed of two layers: a convolution followed by a pooling/subsampling layer. Their purpose is feature extraction. The features are collected by the convolutional filter into the format of feature maps. It is common practice that for each convolution layer, the number of learned filters a power of two (such as 32), and then is



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 29, 48, 32)        320
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 24, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 24, 64)        18496
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 12, 64)         0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 7, 12, 128)        73856
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 3, 6, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                147520
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 455
=================================================================
Total params: 240,647
Trainable params: 240,647
Non-trainable params: 0
_________________________________________________________________
```

Fig. 3. CNN Model Architecture

multiplied by two for each successive layer. This increases the filters as the output spacial volume decreases. That being said, our model has 32, 64, and 128 learn filters, respectively. Further, the kernel size was selected as 3x3 for each layer because of both common practice and input size. Increasing the filter dimensions is only recommended when the input image is greater than 128x128 pixels. Because the samples are approximately a third of that size, this would be unnecessary and possibly cause information loss. In earlier experiments, we worked with microfilters (1x1) to try to emulate the microarchitectures of the ResNet/Inception models, but this proved to be difficult, especially considering the time constraint. It also did not appear to make too large of a difference. Next, we chose to default the step size at 1x1, given that the input images are already relatively small. We also chose to change the padding to the 'same' value so that we could specify the pooling layers as two-dimensional. However, we believe using the 'valid' value (thereby eliminating the pooling layers) would have a similar result. Finally, we decided on the ReLU activation function for each layer, as ReLU trains much faster, has less expensive operations, and is more expressive for fine-tuned classification. ReLU is also the choice for must current deep learning implementations.

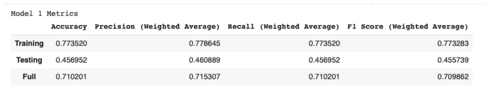
The pooling layers reduce the dimensionality of the feature maps produced by the convolution layers. For our model, we chose MaxPooling, which is a form of nonlinear down-sampling. MaxPooling works by partitioning the input image into a set of non-overlapping rectangles, then outputting the maximum value of each set. We defined these layers with the default pool size of 2x2, which specifies the dimensions of pooling window for the resulting feature map. As with some of our convolutional choices, this was decided because of common practice.

Following the convolutional blocks is the flattening layer, which has the purpose of preparing the feature maps to be input into the multi-layer perceptron.

The architecture of the multi-layer perceptron came about through experimentation. The output layer was not considered in these experimentations; the following discussion exclusively refers to 'internal' hidden layers. Initially, we tried the model with three hidden dense layers, each successive layer having the number of neurons equal to the power of two, in the order opposite to the filter quantities in the convolution layers. However, this increased the complexity and had little to no performance payoff. Then, we tried a single dense layer with 32 perceptrons, which did not meet even the baseline performance. Next, we analyzed the results of a single and double layer architecture with 128 and 64 neurons. We ultimately decided on a single layer with 64 neurons, due to the approximately equal performance with lower complexity. Finally, the output dense layer has seven neurons, each one representing a class. The network outputs seven values, which are normalized and converted from weighted sums into probabilities by the softmax activation function. That being said, the output for each image will be an array of seven values, representing the probability of membership into each respective class. To get a single value, we simply output the index of the maximum value and used a dictionary to map it to the corresponding emotion.

After the model architecture was defined, we compiled the model using categorical cross entropy loss and the Adam optimizer, which is the standard for image classification. Then, we trained and analyzed the results of two models, one using the original training data, and the other using augmented data. Each was trained for 20 epochs, which was determined after examining charts of training versus testing accuracy and loss. The results of each model are discussed next.

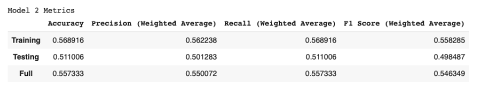
3) Performance: The following tables (Figures 4-5) summarize the performance of Model 1 (fit to original training data) and Model 2 (fit to augmented training data):



Model 1 Metrics

	Accuracy	Precision (Weighted Average)	Recall (Weighted Average)	F1 Score (Weighted Average)
Training	0.773520	0.778645	0.773520	0.773283
Testing	0.456952	0.460889	0.456952	0.455739
Full	0.710201	0.715307	0.710201	0.709862

Fig. 4. Model 1 (Original Data) Performance



Model 2 Metrics

	Accuracy	Precision (Weighted Average)	Recall (Weighted Average)	F1 Score (Weighted Average)
Training	0.568916	0.562238	0.568916	0.558285
Testing	0.511006	0.501283	0.511006	0.498487
Full	0.557333	0.550072	0.557333	0.546349

Fig. 5. Model 2 (Augmented Data) Performance

There are a few notable points about the model performances:

- For Model 1, there are large disparities in the training/full and testing metrics. This indicates that Model 1 is possibly overfit, despite stopping it at the apparent overfit point (20 epochs). This is not an issue in Model 2, where all of the values are approximately .5.
- For Model 2, the training/full and testing accuracies are approximately equal, especially across all metrics. This

indicates a considerably balanced performance across all classes, although we must also take into consideration the imbalance in the training data. Because the metrics are calculated with a weighted average, it is plausible that the less/more prominent classes could be affecting this average.

- Between the models, there was a decrease in training/full data performance, but a significant increase in testing data performance - approximately up six percentage points. Therefore, we can conclude that augmenting the data had a significant impact on performance.

Based on the above analysis, we decided to use Model 2 (fit on augmented data). Despite having lower training/full data performance than Model 1, the confusion matrices revealed that Model 2 is far more generalized (see Appendix).

Of all the models that will be discussed in this paper, the Convolutional Neural Network performed the best by a wide margin. Achieving over 50 percent accuracy, it is comparable to the highest performing full-face emotion detector (83.4 percent full data accuracy). This is further discussed in Section V, Comparing Algorithms.

*F. **Bayesian Network***

1) Overview: Bayesian Networks have been mentioned in recent research about image classification tasks, particularly facial expression detection. This model will have the advantage of probabilistically predicting the emotion contained within a facial image instead of through a direct read by looking at the texture. Because of the subtlety inherent in detecting emotions without a mouth, this model may offer a way of providing satisfactory accuracy.

2) Implementation: The algorithm works by subdividing the image into an k by k grid and then mapping each block of the image to a vector, assigning a grouping to it, computing the probability of an emotion given the group with each block. The probability is calculated using the Naive Bayes algorithm. Naive Bayes is the simplest Bayesian Network. It works by assuming each attribute is independent and progressively multiplying them together through Bayes Theorem. The specifics of the algorithm are outlined as follows:

- Divide the image into an k by k grid. This is done in place of feature extraction and an essential component of this algorithm is to find the appropriate partitions to make for this grid so that each block that is created is large enough to be distinct from those around it but small enough to capture detail. Each block is stationary through this algorithm, preventing the granularity of a sliding window but also reducing the correlation between blocks making the use of naive bayes easier and more effective.

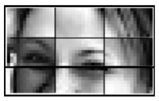


Fig. 6. Face partitioned into 3 by 3 grid

- For each block, a series of five statistics are calculated and mapped to a five dimensional vector. The first statistic to be calculated is the mean grey scale color of the entire block, the following four are used to measure the roughness of the block and are all calculated from the co-occurrence matrix of the block. The co-occurrence matrix measures the smoothness of an image, it works by having a row and column for every grey scale and filling in that cell the number of times a pair of pixels with the grey scale corresponding to the row and column indices occurs. From this matrix the Haralick features are derived. These include the matrix's energy, entropy, contrast, and homogeneity. Energy is most affected by the level of greyness in the image. Entropy measures the lack of uniformity in a sample, a higher entropy would, in this case, mean that the block has a greater variety of colors which could be interpreted as being rougher. Contrast weighs more heavily pixels that have similar colors to the ones around them while the calculation for homogeneity is the inverse of the contrast with the same weights. Their equations are outlined below:

$$E = \sum_{i,j} (p(i,j))^2$$

$$\mathrm{ENT} = -\sum_{\mathrm{i}} \sum_{\mathrm{j}} \mathrm{p(i,j)} \log \mathrm{p(i,j)}$$

$$\mathrm{CONT} = \sum_{\mathrm{i}} \sum_{\mathrm{j}} (\mathrm{i}-\mathrm{j})^2 \mathrm{p(i,j)}$$

$$\mathrm{HOM} = \sum_{\mathrm{i,j}} \frac{1}{1+(\mathrm{i}-\mathrm{j})^2} \mathrm{p(i,j)}$$

- For each vector kmeans clustering is applied. The cluster group then becomes the attribute associated with each block feature of each image.
- Multinomial Naive Bayes is applied to each image with the cluster of its block being each attribute.

3) Performance: Multiple different pairing of grid partitions against kmeans clusters were tested.

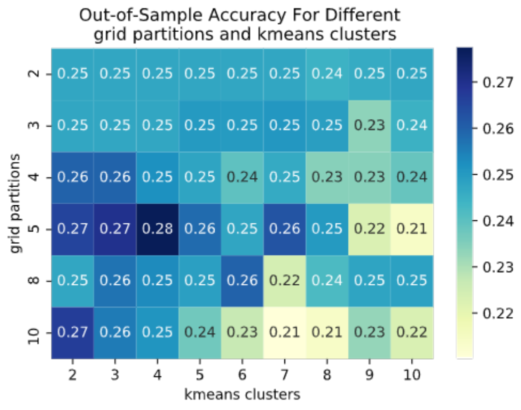


Fig. 7. Grid Partition to Kmeans Clusters Performance

From this the best number of grid partitions is found to be five and the best number of kmeans clusters to be four. For this best performing algorithm, the out-of-sample accuracy was 0.2772. What is perhaps more surprising is that the in-sample performance was 0.2761. It appears that the use of probabilistic methods were very effective in this problem for a limiting the bias-variance trade-off.

When examining the confusion matrix of the bayesian network based model one finds that the probabilistic nature of the algorithm lead to biases in preferring certain classifications. This was likely made worst by the imbalanced nature of the dataset.

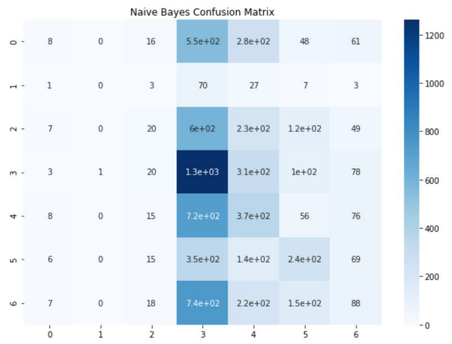
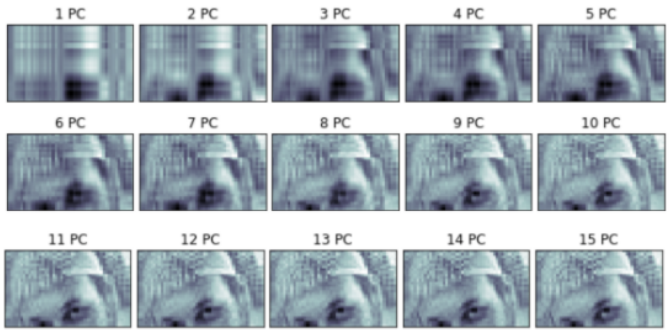


Fig. 8. Bayesian Network Confusion Matrix

G. Support Vector Machine

1) Overview: Support Vector Machines are also a frequent solution cited in recent research. SVMs are extraordinarily flexible, and offer significant utility in this application for categorizing subtle facial structures associated with each emotion. They are characterized by their functionality, which involves mapping arrays via kernel functions to hyperplanes in higher dimensions. With many parameters to tune, including kernel parameters that define the hyperplane and the regularization parameter, represented as 'C', that controls the trade-offs between training point classification, SVMs are highly customizable and offer the tunable flexibility necessary for this project.

2) Implementation: Exploring SVMs, three different models were trained and tested using scikit-learn. For two of the models, Support Vector Classification was the implementation chosen from the SVM wrapper due to the aforementioned flexibility. The kernel assumed for these models was a Radial Basis Function. The other model employed LinearSVC, which assumes a Linear kernel and is implemented in terms of liblinear instead of libsvm. In pursuing a high-accuracy final result, the initial goal was to aim for 95 percent explained variance. This led to 1392 features, however, which resulted in significant overfitting in regular SVC. To combat this overfitting, the data was scaled and Principal Component Analysis was performed. Through PCA, it was determined that reconstruction of images from the dataset using Principal Components yielded diminishing returns in explained variance very quickly. Visibly, this is apparent before using even 10 PCs, and the full image can be reconstructed using only the first 15.



```
PC 1 Explained variance: 0.39869102439820026
PC 2 Explained variance: 0.5412700335194147
PC 3 Explained variance: 0.6432693611037161
PC 4 Explained variance: 0.7214895170270448
PC 5 Explained variance: 0.7753611522361763
PC 6 Explained variance: 0.812693556717163
PC 7 Explained variance: 0.8421244369508674
PC 8 Explained variance: 0.8667397411990909
PC 9 Explained variance: 0.8880694358642881
PC 10 Explained variance: 0.9075002040829063
PC 11 Explained variance: 0.9206023530063825
PC 12 Explained variance: 0.9330749928301334
PC 13 Explained variance: 0.9445244821757762
PC 14 Explained variance: 0.9551441506189958
PC 15 Explained variance: 0.9631660893050251
```

Fig. 9. Reconstruction based on principal components

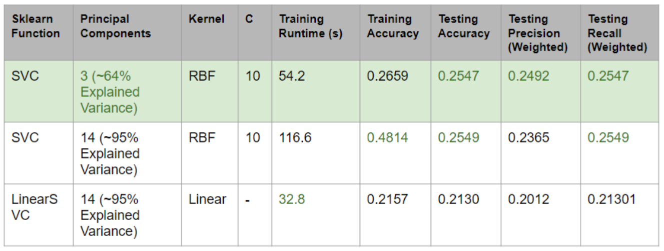
As a result of PCA, experimentation was done with 1-14 PCs and different regularization parameters. Basic SVC would not complete without a very low number of PCs. In the best model, the number of features decreased from 1392 to only 3, as selected based on explained variance and several basic tests. While this number corresponded with an explained variance of only 64.3 percent, the model yielded testing accuracy similar to that of a model using 10 principal components. This indicates that the vast majority of the explained variance past 64.3 percent was noise, and therefore provided little for the model [BhattacharyyaBhattacharyya2019].

3) Performance: The 3 models trained produced using SVM are as follows:

- Model 1: sklearn function SVC assuming RBF kernel, C=10, using 3 PCs for 64 percent explained variance
- Model 2: sklearn function SVC assuming RBF kernel, C=10, using 14 PCs for 95 percent explained variance
- Model 3: sklearn function LinearSVC assuming Linear kernel, using 14 PCs for 95 percent explained variance

The above chart lists performance results for each trained model. Most notably, it lists:

- Model 1: Testing Accuracy = 25 percent, Testing Precision (Weighted) = 25 percent, Testing Recall (Weighted) = 25 percent
- Model 2: Testing Accuracy = 25 percent, Testing Precision (Weighted) = 24 percent, Testing Recall (Weighted) = 25 percent



Sklearn Function	Principal Components	Kernel	C	Training Runtime (s)	Training Accuracy	Testing Accuracy	Testing Precision (Weighted)	Testing Recall (Weighted)
SVC	3 (~64% Explained Variance)	RBF	10	54.2	0.2659	0.2547	0.2492	0.2547
SVC	14 (~95% Explained Variance)	RBF	10	116.6	0.4814	0.2549	0.2365	0.2549
LinearSVC	14 (~95% Explained Variance)	Linear	-	32.8	0.2157	0.2130	0.2012	0.21301

Fig. 10. Performance results for all SVM models

- Model 3: Testing Accuracy = 21 percent, Testing Precision (Weighted) = 20 percent, Testing Recall (Weighted) = 21 percent

From these results, it is clear that SVC is generally a better option than LinearSVC. LinearSVC was chosen as an alternative for its faser computation time and increased flexibility with loss function, which is reflected in its very low training runtime, but it ultimately yields testing accuracy that is significantly lower than either of the SVC models. Model 1, when compared to the similar Model 2, displays a significantly shorter training runtime, worse training accuracy, similar testing accuracy, better testing precision, and similar testing recall. Therefore, Model 1 is the best model trained. Additionally, due to the testing accuracy remaining the same for models 1 and 2, we can conclude that the explained variance past 64 percent is largely noise.

H. Pipeline

We thought it would be interesting to test both our preprocessing techniques and best model (in this case, the CNN trained on augmented data) on images 1) containing subjects wearing face masks; and 2) containing more environmental noise. After collecting the images (see Section D, Additional Data), we built a pipeline consisting of preprocessing, prediction, and annotation steps.

The preprocessing is similar to that of the training/testing dataset, including only a few additional steps. First, the input image is copied and converted to grayscale. Then, the Dlib face detector is called to retrieve the bounding box coordinates of each face. Using these coordinates, we create separate image arrays for each face detected in the image. These image arrays are transformed to match the model input, being cropped using Point 31 and then reshaped to (48, 48). Once these arrays are prepared, we load the CNN weights and architecture, then compiling the model with categorical cross entropy loss and Adam optimization, as in the training stage. Then images are then fed into the model, which returns a vector with seven class probabilities. We take the index of the maximum probability as the key to a dictionary, which maps the index to the emotion string. Finally, we draw the bounding box and the emotion label onto the copy of the original image, and then export it to JPEG file.

The results of one image are displayed in Figure 11, and the rest can be viewed in the Appendix. In the example, there are two concerns: 1) one of the faces is not detected; 2) one

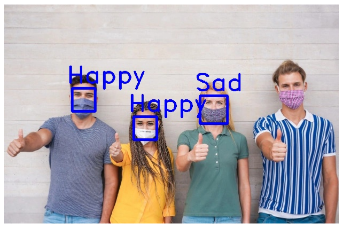
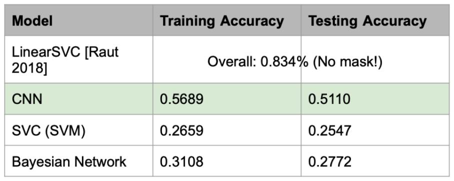


Fig. 11. Resulting annotation on a multi-subject, face mask present image.

of the faces is 'incorrectly' labelled, as determined by context clues. These issues are consistent among all of the test images, and can possibly be attributed to the noise contributed by the face mask (especially if the cropping is poor). A larger issue we face is that majority of the labels are Class 3, 'Happy', which made up the highest proportion of the training set. This highlights the negative effects of class imbalance and further motivates us to address this issue in future improvements.

V. Comparing Algorithms

Because our models were trained using limited facial features, we did not have high expectations for performance. The [RautRaut2018] model, trained on images with full facial features, achieved .834 full data accuracy. Unlike our images, their data included the mouth, which is unarguably the most important facial feature for emotion inference. That being said, we set our accuracy baseline at 0.1429, which is equivalent to random guessing $\frac{1}{7}$. Overall, all three of our models exceeded this baseline, with the SVM and Bayesian Network falling short of the CNN. The training and testing accuracies of each finalized model are summarized in Figure 12.



Model	Training Accuracy	Testing Accuracy
LinearSVC [Raut 2018]	Overall: 0.834% (No mask!)	
CNN	0.5689	0.5110
SVC (SVM)	0.2659	0.2547
Bayesian Network	0.3108	0.2772

Fig. 12. Accuracy Comparison of Algorithms

Because the SVM and Bayesian Network performed in the same range, they can be compared first. In the preprocessing stage, the Bayesian Network uses filters and histogram analysis to create color and texture feature vector. Especially when considering the flexibility of the filter size, the Bayesian Network implements a rather robust feature extraction process, one similar to the CNN. On the other hand, the SVM performs dimensionality reduction using Principal Component Analysis, which, while useful in terms of complexity, does not do much for prediction performance. However, the SVM model does have the flexibility for histogram analysis like the Bayesian Network, as discussed in Section VI, Future Improvements. Therefore, we can hypothesize that these different preprocessing techniques play a role in the Bayesian's slight performance lead. Strictly looking at the algorithms, their main difference is with their feature handling. Because Naive Bayes makes an assumption of independence, the Bayesian Network treats the features independently. On the other hand, in an SVM (with a nonlinear kernel), feature dependency is considered - to which extent is questionable. However, these algorithms are not truly comparable, being that Naive Bayes is more probabilistic and SVM is more geometric. Therefore, we attribute their similar performances to coincidence in this application.

The Convolutional Neural Network performed far better than its two competitor models. This was expected being that CNNs are more compatible and essentially designed for computer vision tasks. Therefore, we will compare it to the Raut model [RautRaut2018], which was a SVM with a linear kernel. While CNN and SVM are both supervised learning techniques, they differ in their handling of multiclass classification problems. Both algorithms are centric to the 'one-versus-all' idea, the CNN can implement this with one classifier, whereas SVM requires a 'one-versus-all' classifiers for each class, making SVM less computationally favorable. Additionally, although CNNs are more designed to the image classification task, SVMs cannot overfit, which is a major advantage. On another note, it is interesting that Raut achieved the highest accuracy with a linear kernel, especially considering our success with and the general flexibility of nonlinear kernels (namely Radial Basis Function) for image classification tasks.

In terms of performance, it is not surprising that the Raut model achieved significantly higher full data accuracy (0.834) than the CNN (0.557). As mentioned briefly earlier, the Raut model was trained on full face images, including the mouth. The mouth is undoubtedly the most important feature for emotion detection, and with us eliminating it from our images, our models were at a tremendous disadvantage. Nonetheless, the CNN performance is fair given the limited features, although the opportunity exists for improvement.

VI. Future Improvements

Despite us far exceeding our baseline on all models, all of the solutions could see improvement. This include, but are not limited to:

- Balancing the dataset - This could be achieved with additional data augmentation, as we saw improvement between the CNN model performance with that change. However, we could also try to incorporate more training examples for the classes making up a lower proportion of the data (namely, Class 1). Another option would be to remove samples from Class 3 to make it more proportionate to the others, but common practice in machine learning would be to increase, rather than decrease, the total number of samples.
- Data Augmentation - As mentioned above, data augmentation is good for addressing class imbalance. Additional data augmentation could help us to make the dataset more diverse, especially for the lower performing classes.
- Developing a deeper/finer/regularized neural network - Further experimenting with smaller filters, hidden layers and their densities, as well as regularizers could possibly improve the CNN performance.
- Performing a more intensive hyperparameter search - For SVM and the Bayesian networks, there are many combinations of parameters to test. Given more time, we could implement more robust grid search, exploring parameters beyond just the kernel and margin sizes.
- Establishing ensembling classifiers - All of our references mention ensembling classifiers. This process could be intensive, which is why we did not attempt it. Especially considering our class imbalance, ensembling our classifiers or multiple of the same algorithm could significantly help our model performance, particularly in precision and recall.
- Incorporating more transformative techniques into SVM, including Histogram Oriented Gradient.
- Training models to process image context such as background, body language, etc. - When looking at the pipeline results, we used environmental features (such as hand gestures and setting) to infer the emotions of each image. However, our models ignore this 'noise' and classify the faces independent of their setting. Using techniques such as object detection and image segmentation, we could associate setting objects or body language features with certain emotions, then use such to supplement the predictions.

The team believes that all of these improvements are viable options to pursue for an improved final result. However, each of these routes would come at a higher computational cost. Thus, we would proceed with these improvements only after careful analysis of their effects on performance and complexity.

VII. Conclusion

While the complexities of human socialization could never be fully mapped by a model, as each of us is a model more complex than any system that could be created by code, this project has taken the most preliminary of steps to begin mapping emotions to facial expressions in an ever-changing world more complex than one could possibly imagine. Comparing the applications of Convolutional Neural Networks, Bayesian Networks, and Support Vector Machines for this project, it is almost difficult to make a comparison. Convolutional Neural Networks are so fundamentally different from Bayesian Networks and Support Vector Machines that it is an unfair comparison. It is no surprise, in hindsight, that the difference it shows lends itself to a vastly different result than the other methods. Ultimately, the Convolutional Neural Network yielded the project's best result at over 50 percent accuracy, approaching the accuracy of top full-face emotion detection models. Never-the-less, the project showed the potential and effectiveness that can be achieved with algorithms that would normally rely on the face, and while there is much room for improvement it is still striking to see that better than random predictions are achieved, even with little innovation from those algorithms prior. This is a promising sign that the team feels indicates meaningful functionality and a potential for future use.

References

[BhattacharyyaBhattacharyya2019] Bhattacharyya, S. (2019). Understanding pca (principal component analysis) with python. *Towards Data Science 1*(1), 1.

[CarbonCarbon2020] Carbon, C.-C. (2020). Wearing face masks strongly confuses counterparts in reading emotions. *Frontiers in Psychology 11*, 2526.

[Jayech and MahjoubJayech and Mahjoub2010] Jayech, K. and M. A. Mahjoub (2010). New approach using bayesian network to improve content based image classification systems. *International Journal of Computer Science Issues (IJCSI) 7*(6), 53.

[RautRaut2018] Raut, N. (2018). Facial emotion recognition using machine learning.

[SharmaSharma2017] Sharma, A. (2017). Convolutional neural networks in python. *DataCamp Community 1*(1), 1.

Appendix

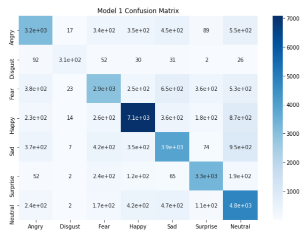


Fig. 13. CNN - Full Data Confusion Matrix for Model 1 (trained using original data)

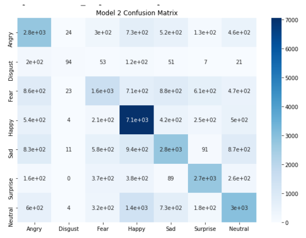


Fig. 14. CNN - Full Data Confusion Matrix for Model 2 (trained using augmented data)

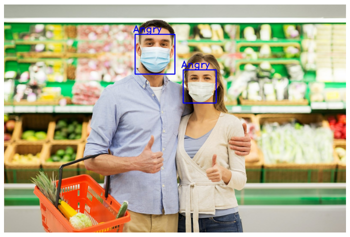


Fig. 15. Pipeline - Test Image 1

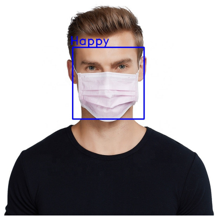


Fig. 16. Pipeline - Test Image 2

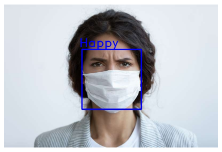


Fig. 17. Pipeline - Test Image 3

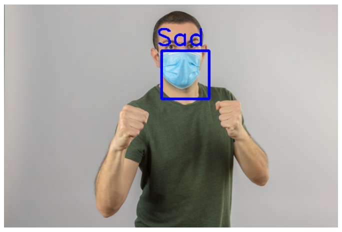


Fig. 18. Pipeline - Test Image 4

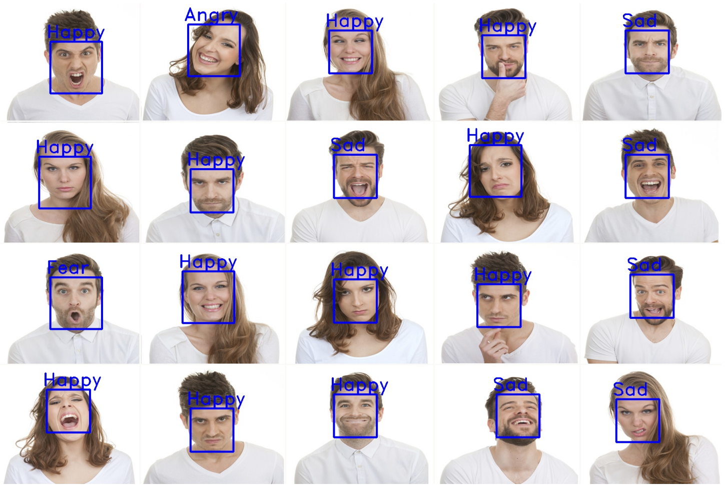


Fig. 19. Pipeline - Test Image 5

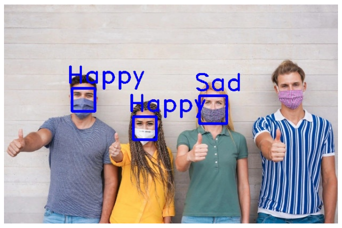


Fig. 20. Pipeline - Test Image 6