Eigen Emotions: Detecting Emotions From Facial Expression

Authors

Tal Kozakov	315720250
Omer Hazan	206513616
Yaniv Rosenberg	318433562
Ofir Bar Lev	318203403

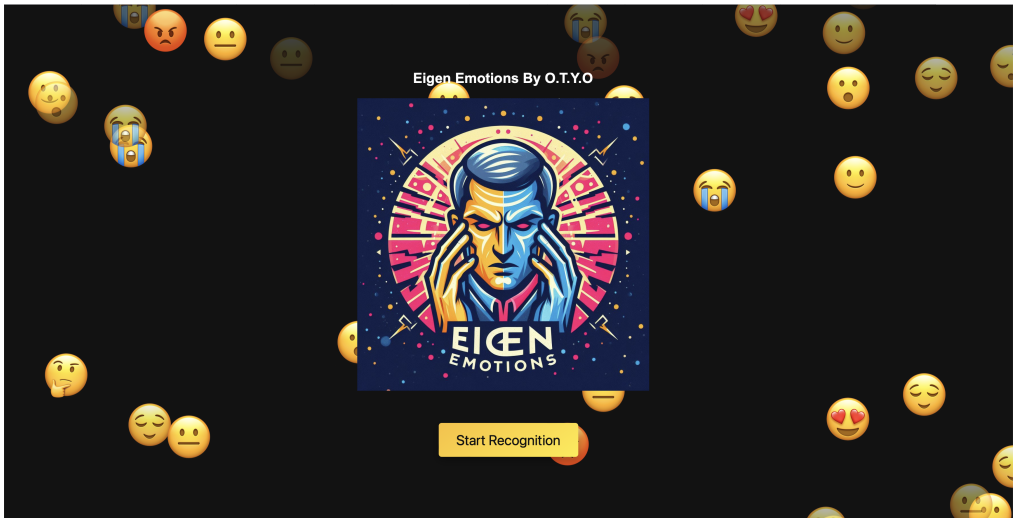


Figure 1: Landing Page of our GUI

1 Abstract

Eigen Emotions is a computer vision project aimed at detecting emotions from facial expressions. The project utilizes image processing techniques to analyze facial features and infer the underlying emotions. Our project is composed of two main parts: face detection and emotion recognition. This paper provides an overview of the key components and methodologies employed in Eigen Emotions.

2 Introduction

This paper presents a comprehensive analysis of a project composed of two main aspects: face detection and emotion classification.

Face detection serves as a fundamental step in many computer vision applications. In our project, we employed the Viola-Jones algorithm, a well-established method known for its efficiency and accuracy in detecting faces in images. This algorithm utilizes a cascade of simple classifiers trained with Haar-like features to efficiently scan images and identify potential regions of interest.

Emotion recognition is another vital area within image processing, particularly in fields like affective computing and human-computer interaction. In our project, we attempt to classify four main emotions that humans display distinctively: Happy, Angry, Sad, and Surprised. Our approach to emotion classification was based on the principles underlying eigenfaces. Eigenfaces represent a popular technique for facial recognition, leveraging the concept of principal component analysis (PCA) to extract essential features from face images. By applying similar principles, we developed a framework to classify emotions based on facial expressions.

In addition to face detection and emotion classification, our project involved preprocessing steps to enhance the robustness and accuracy of our algorithms. One crucial preprocessing step involved eliminating lighting effects that could vary between images. Lighting variations pose a significant challenge in face detection and emotion recognition tasks, as they can obscure facial features and affect the performance of algorithms. By standardizing the lighting conditions across images, we aimed to improve the reliability of our results and ensure consistency in our analyses.

Our project faces several challenges and constraints, including a poor quality dataset set, our algorithm fails to detect emotions when the user is not facing the camera (for example when the user turns his profile to the camera). We also encountered difficulties classifying emotions when the user is talking. The algorithm works best when the user is expressive above average. Our main assumptions in the project, was that it was possible to classify our four chosen emotions by creating mathematical representation of each emotion space and classifying the correct emotion by projecting the image on each space.

Overall, this paper provides insights into the methodologies, techniques, and results obtained through the implementation of face detection and emotion classification algorithms in the context of image processing.

3 Methodology

3.1 Face tracking

3.1.1 Preprocessing

To mitigate the effects of varying lighting conditions across images, we applied a preprocessing step to normalize the mean intensity of each image. This involved calculating the mean pixel intensity of the entire image and adjusting it to a standardized value of 0.5. Normalizing the mean helps to reduce the impact of lighting variations, ensuring that the algorithms for face detection and emotion classification are more robust and accurate across different lighting conditions.

3.1.2 Viola–Jones

To solve the problem of identifying a face within an image, we employ a technique known as the Viola-Jones algorithm. This method, widely used for face detection, operates by utilizing a cascade of classifiers trained on Haar-like features. These features, resembling rectangular filters, compute the difference between the sum of pixel intensities in adjacent regions of an image. By efficiently scanning the image using this cascade of classifiers, the Viola-Jones algorithm can accurately and rapidly identify potential regions of interest containing faces. Imagine we want to find faces of a certain size. To accomplish this, we utilize a window that is roughly the size of the face we are seeking, and we traverse this window across the image, possibly through discrete convolution operations. At each pixel, we extract features into a vector f and then classify these features as either a face or a non-face (binary '0' or '1'). The algorithm effectively detects faces by analyzing these features within the image, such as the red window detecting faces within the green window as shown in figure 2.

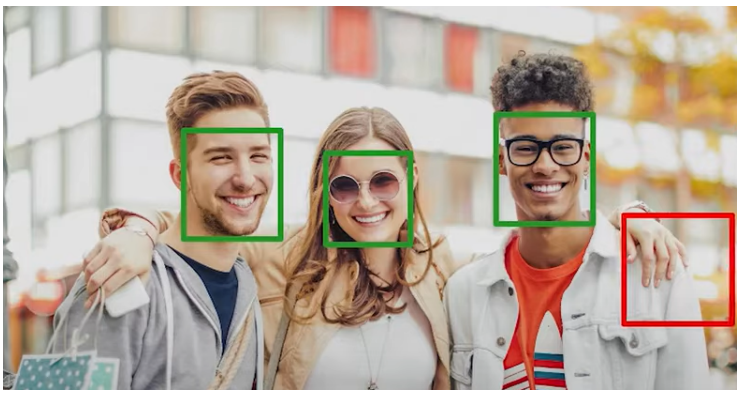


Figure 2: Viola-Jones Algorithm

Overall, for each window we would go over the next procedure:

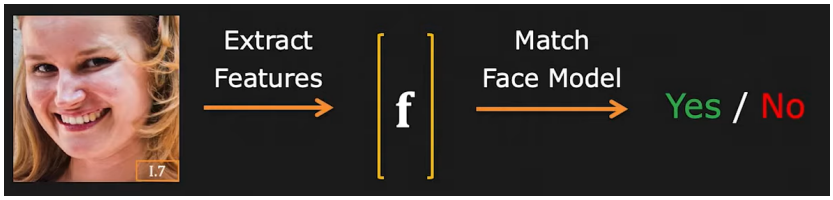


Figure 3: Viola-Jones Algorithm

That brings us to the next questions:

1. Which features represent faces well?
2. How to construct a face model and efficiently classify features as face or not?

3.1.3 Ha'ar features

As we develop features for face detection, we want them to be able to discriminate between faces and non-faces within as accurate as it can get. A second attribute that we are looking for, we want these features to be efficient to compute. An image can have thousands of pixels and we are going to apply these features at each one of these pixels, so it must be extremely efficient to compute. Essentially, we would like to detect a face in a live video so computing processes have to be very fast.

The features we are going to use are called Haar features and computed by Haar filters. These kinds of filters are based on the square function. For example, here are some Haar filters. The white templates represent the number '1' and the black templates represent the number '-1':

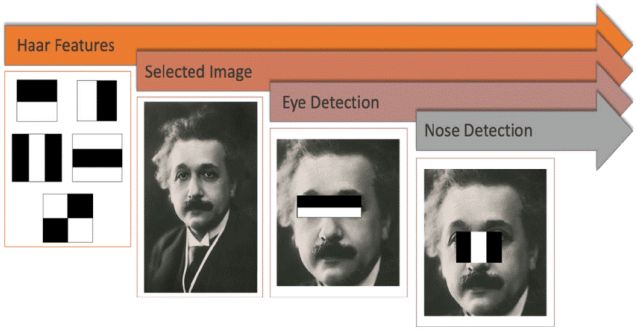


Figure 4: Haar Features

The Haar filters tend to be two-valued filters, and this is computationally very advantageous. We will apply each one of these filters as a correlation of the image, namely convolution operation, which will produce for each pixel (i, j). Finally, we end up with the features vector $f[i, j]$ as described in the following example:

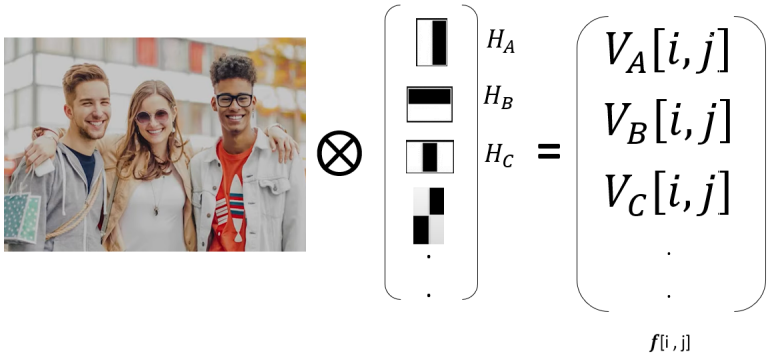


Figure 5: Haar Features Extraction

Each product $V_k[i, j]$ is calculated by the next formula:

$$V_k[i,j] = \sum_m \sum_n I[m-i, n-j] H_k[m,n]$$

Since it is a Haar filter, which is only two-valued of 1 and minus 1, it is more efficient to apply this correlation without doing any multiplication. Essentially, we are applying this method:

$V_k[i,j] = \sum$(pixel intensities in white area)$-\sum$(pixel intensities in black area)

For filters with a size of $M \times N$, the computational cost will be $(M \times N - 1)$ per pixel, per filter, and per scale. Hence, we should ask the question, can we do better? The next method, so-called the 'Integral Image', can answer this question easily. We can calculate the 'Integral Image' and use it to compute the Haar vector. For example, we will look at the next image (left side) and its 'Integral Image' (right side), which is a table that holds the sum of all pixel values to the left and to the top of a given pixel, inclusive:

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Figure 6: Caption for the figure

Basically, with any given Haar filter, we can calculate the product $V_k[i, j]$ by adding and subtracting pixels from the integral image. This method is much more efficient since we only calculate the integral image once and use its values for computing the Haar vector.

3.1.4 Classification

The next question was how we can efficiently classify features as face or not. The Viola-Jones algorithm requires a training phase where positive and negative (whether the image contains a face) image samples are used. Features (Haar-like features) are evaluated on these samples, and 'Adaptive Boosting' is used to select a subset of features that are most effective in discriminating between positive and negative samples. After the feature selection and 'Adaptive Boosting' training, the resulting strong classifier is converted into a cascade of weak classifiers organized into stages. During detection, the integral image is used to rapidly compute the Haar-like features in the image. Each stage of the cascade processes a region of interest, and if a region passes all stages without being rejected, it is classified as containing a face.

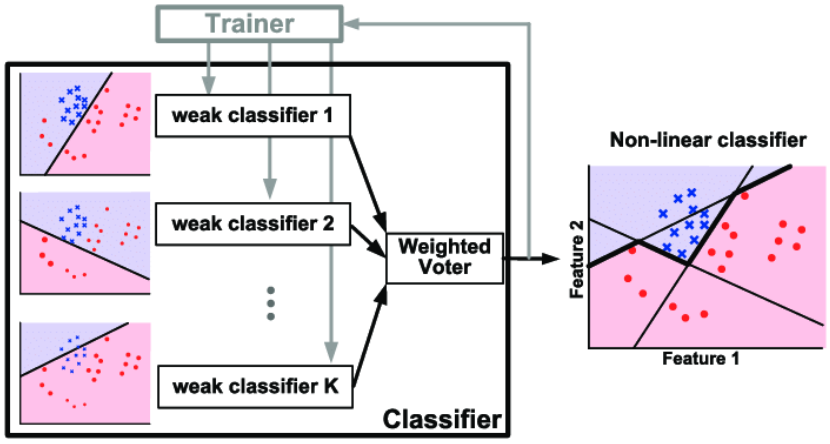


Figure 7: Classification

3.2 Post-processing

Following the face detection process using the Viola-Jones algorithm, we implemented post-processing techniques to enhance the accuracy of the results and to optimize the face images in preparation for emotion detection. Initially, we resized the detected face image to dimensions of 48×48 pixels. This step ensured uniformity in the size of all detected face images for use with the emotion detection algorithm. Additionally, we applied histogram equalization to mitigate extreme variations in image lighting. Finally, we used a Gaussian filter to smooth the image and then subtracted it to create a high-pass filter. This process helped to enhance the details of the detected faces while reducing noise and improving the overall quality of the images.

3.3 Emotion classification

3.3.1 Preprocessing

To eliminate the influence of lighting variations, a common challenge in image processing, we adopted a preprocessing approach. Firstly, we converted the images to grayscale to better suite the mathematical process in our algorithm. Next, we adjusted the mean intensity of each grayscale image to a standardized value of 0.5 by adding a constant value. This step helps in reducing the impact of varying lighting conditions across different images. Additionally, we applied histogram equalization to the normalized grayscale images. Histogram equalization redistributes the pixel intensities in an image to achieve a more uniform distribution, enhancing the overall contrast and improving the visibility of facial features. By performing these preprocessing steps, we aimed to improve the robustness and accuracy of our emotion classification algorithm, particularly in the presence of lighting variations.

The constant value to be add to each image is calculated by:

$$\text{constant value} = 0.5 - \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \text{image}(i, j)$$

3.3.2 Building Emotion Matrices

To construct the emotion matrices, we collected approximately 1000 images of people displaying an emotion ,for each emotion that we aim to classify (Happy, Angry, Sad or Surprised), each image is of size 48×48. Each image was then flattened into a column vector of size 2304.For each emotion, we calculated the mean of the flattened images.

$$\text{Mean Vector}_{\text{emotion}} = \frac{1}{N_{\text{emotion}}} \sum_{i=1}^{N_{\text{emotion}}} \text{Image}_i$$

Where:

- Mean $\text{Vector}_{\text{emotion}}$ is the resulting mean vector specific to the emotion.
- N_{emotion} is the total number of flattened images for the particular emotion.
- Image_i represents the i-th flattened image vector.

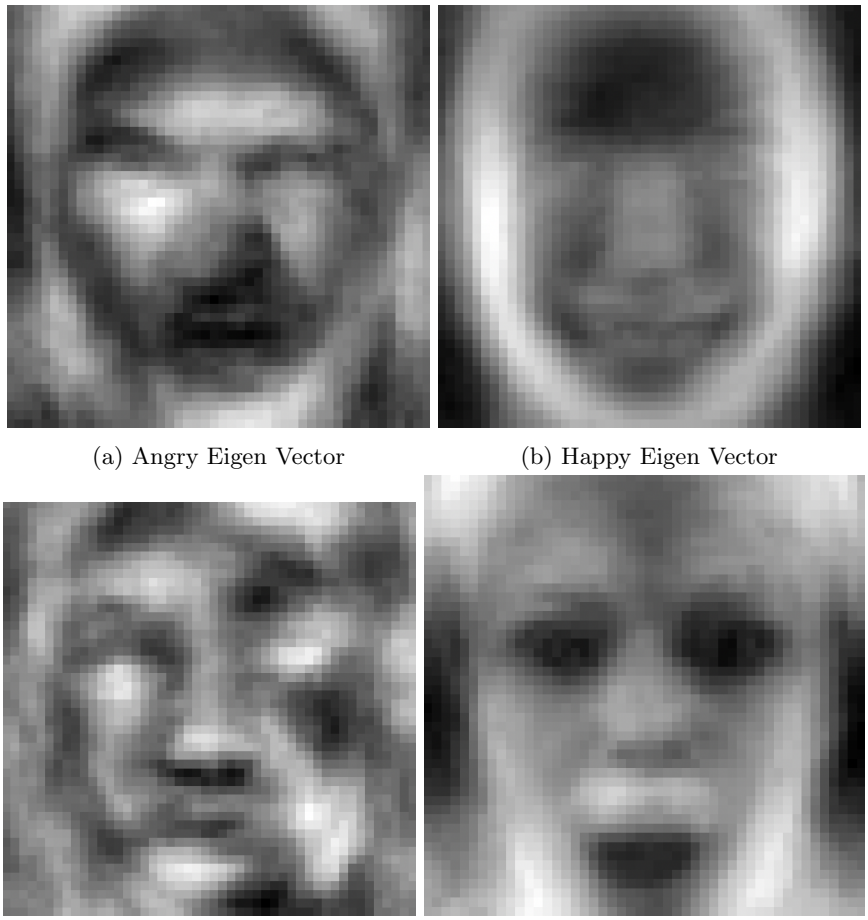
From each flattened image we subtract the Mean Vector. For each emotion category, we formed a matrix by concatenating these flattened images (after mean subtraction) as columns of the matrix.

For instance, let N represent the number of images collected for a specific emotion category. Then, the matrix representing that emotion category would have dimensions $2304 \times N$, where each column represents a flattened image.

Next, we applied Principal Component Analysis (PCA) to each emotion matrix. PCA is a dimensionality reduction technique that identifies the principal components, or eigenfaces in this context, that capture the most significant variations in the data. By applying PCA to the emotion matrices, we obtained a set of eigenfaces corresponding to each emotion category.

To select the most relevant eigenfaces, we implemented a scoring system. This system allowed us to identify the eigenfaces with the highest scores, indicating their significance in representing the variability within the emotion category.

It is noteworthy to mention the scalability and efficiency of our approach. Despite collecting a large dataset of approximately 1000 images for each emotion, we observed that comparable results, or nearly as good, could be achieved with significantly fewer images. For instance, by reducing the dataset to only 200 images per emotion category, our algorithm still produced satisfactory results. This scalability highlights the effectiveness and adaptability of our methodology, even with limited training data.



(a) Angry Eigen Vector (b) Happy Eigen Vector

(c) Sad Eigen Vector (d) Surprised Eigen Vector

Figure 8: Eigen vectors

3.3.3 Scoring System

Our scoring system is designed to identify eigen vectors that exhibit strong correlation with a specific emotion while displaying weak correlation with other emotions. This ensures that the selected eigen vectors effectively capture the unique characteristics of each emotion category.

To achieve this, we compute the correlation of each eigen vector with the images corresponding to the target emotion as well as with images from other emotions in the dataset. Specifically, for a given eigen vector associated with an emotion category (e.g., "happy"), we calculate its correlation with all images belonging to that emotion category.

Let V_{target} denote the eigen vector under consideration and I_{happy} represent the set of "happy" flattened images in the dataset. The correlation score for V_{target} with respect to the "happy" emotion is computed as the sum of the dot products between V_{target} and each flattened image in I_{happy}.

Similarly, we compute the correlation scores for V_{target} with flattened images from other emotion categories (e.g., "angry", "sad", "surprised"). Let I_{angry}, I_{sad}, $I_{surprised}$ denote the sets of flattened images corresponding to these emotions, respectively.

The deviation between the sum of correlations with the target emotion and the sum of correlations with other emotions is then calculated. This deviation serves as a measure of the discriminative power of the eigen vector, indicating its ability to distinguish the target emotion from others.
For example,Let $\mathcal{S}$ Denote the score and $\mathcal{S}_{happy}$ represents the score for an eigen vector associated with the "happy" emotion, it is computed as:

$$\mathcal{S}_{\text{happy}} = \frac{\sum_{\mathbf{v} \in I_{\text{happy}}} \mathbf{v} \cdot V_{\text{target}}}{\sum_{\mathbf{v} \in I_{\text{angry}} \cup I_{\text{sad}} \cup I_{\text{surprised}}} \mathbf{v} \cdot V_{\text{target}}}$$

Finally, to construct the final matrices, we select the 30 eigen vectors with the highest scores for each emotion category. These selected eigen vectors effectively represent the essential facial features associated with each emotion, allowing for accurate classification and recognition.

3.3.4 SVM Classification

To train our Support Vector Machine (SVM) classifier, we utilized a feature extraction approach based on the chosen eigenfaces for each emotion category.

For each image in the dataset of each emotion category, we followed these steps:

1. Flatten the image to obtain a vector representation.
2. Subtract the mean of the corresponding emotion dataset from the flattened image vector.
3. Compute the dot product of the resulting vector with the matrix of 30 chosen eigen vectors specific to the emotion category.
4. Concatenate this dot product result with the dot products obtained for the other emotion categories.

For example, to process an image in the "happy" emotion category, we flattened the image, subtracted the mean of the "happy" images, computed the dot product with the matrix of 30 chosen eigen vectors for "happy", and concatenated

the result with the dot products for other emotions.

Subsequently, we trained the SVM model using the default parameters provided by the scikit-learn library (SKlearn). These default parameters include the radial basis function (RBF) kernel, among others.

Furthermore, we employed the one-vs-rest (OvR) strategy for multiclass classification. In this approach, a separate SVM classifier is trained for each emotion category, treating it as the positive class while the remaining emotion categories are treated as the negative class.

During the prediction phase for a new image, we repeated the same process of feature extraction. Once the features were extracted, we fed them into the trained SVM models to make predictions about the emotion category of the input image.

4 Analysis: The Process of Choosing Eigen Vectors

For the analysis section, we chose to delve into the three different approaches for selecting eigen vectors we tried and examine their effectiveness in emotion classification within our project.

Principal Component Analysis (PCA) plays a pivotal role in our project, as it forms the basis for constructing emotion matrices. These matrices serve as the fundamental representation of emotional features extracted from facial expressions. By decomposing the high-dimensional space of facial images into a lower-dimensional subspace spanned by eigen vectors, PCA enables us to capture the essential variations in facial expressions. Each eigen vector corresponds to a principal component or a unique facial feature that contributes to the overall variability within the dataset. Therefore, the process of choosing eigen vectors directly influences the quality and representativeness of our emotion matrices.

The significance of selecting appropriate eigen vectors becomes evident in the subsequent classification process. After computing the dot product between an input image and the emotion matrix corresponding to each emotion category, the resulting projection highlights the extent to which the facial expression aligns with the characteristic features of that emotion. Consequently, the accuracy and efficacy of emotion classification heavily rely on the discriminative power of the chosen eigen vectors. A well-curated set of eigen vectors ensures that the emotion matrices effectively capture the nuances of each emotion, facilitating more accurate and reliable classification outcomes.

In the following sections, we meticulously analyze three distinct strategies for

selecting eigen vectors and evaluate their impact on emotion classification performance. Through rigorous experimentation and comparative analysis, we aim to elucidate the strengths and limitations of each approach, ultimately guiding the selection of the most effective method for our project.

The first approach involves selecting eigen vectors based on their contribution to capturing variance within the dataset. Specifically, we opt for a threshold-based method, retaining eigen vectors that collectively represent 95% of the total variance. By prioritizing eigen vectors with larger eigenvalues, we aim to capture the most significant variations within the data while reducing dimensionality. After sorting the eigenvalues in decreasing order, we have $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_p$, and the corresponding eigenvectors $v_1, v_2, ..., v_p$. The total variance in the data is equal to the sum of the eigenvalues, which we denote as $\Lambda = \lambda_1 + \lambda_2 + ... + \lambda_p$. Now, if we want to retain 95% of the variance, we need to find the smallest number k such that:

$$\frac{\lambda_1 + \lambda_2 + ... + \lambda_k}{\Lambda} \geq 0.95$$

In other words, we keep adding eigenvalues from largest to smallest until the sum is at least 95% of the total variance. The corresponding eigenvectors $v_1, v_2, ..., v_k$ are the principal components that we keep. This process is known as dimensionality reduction, as we reduce the number of variables from p to k, while retaining 95% of the variance in the data. The reduced dataset consists of the scores on the first k principal components. This method, uses approximately 250 eigen vectors per emotion (with slight changes for each emotion).

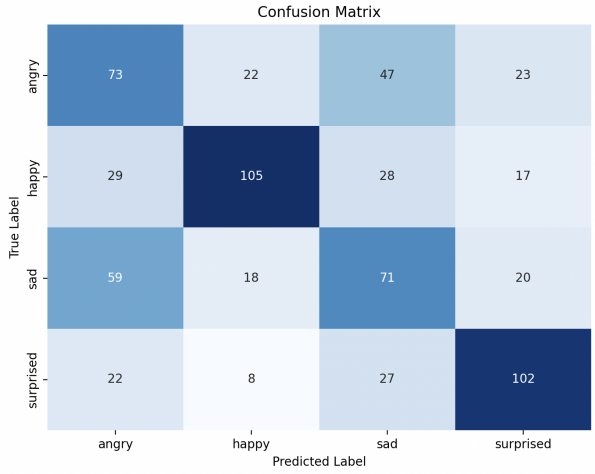


Figure 9: Confusion matrix for 95% variance in data

The second approach entails a more manual selection process. Here, we visually inspect the first 100 eigen vectors (meaning the first 100 eigen vectors corresponding with the 100 largest eigen values) rearranged as images, returning them to their original 48×48 dimensions. We inspected only the 100 first eigen vectors due to two main reasons, the first reason is because of the difficulty and time consumption of going through 1000 eigen vectors for each emotion and the second reason is the fact that after about 80 vectors the images started to look noisier and not interpretable for emotion classification. This method allows us to directly observe the facial features represented by each eigen vector and select those that best correspond to the distinct emotional expressions we aim to classify. By handpicking eigen vectors tailored to each emotion, we tried to achieve a more targeted and nuanced representation of facial features relevant to emotion recognition. This method, uses 20-40 eigen vectors per emotion. For more distinctive emotion like happy there were more high quality eigen vectors (in our opinion), for those the number will be closer to 40 and vise versa for emotions with low quality eigen vectors like sad.

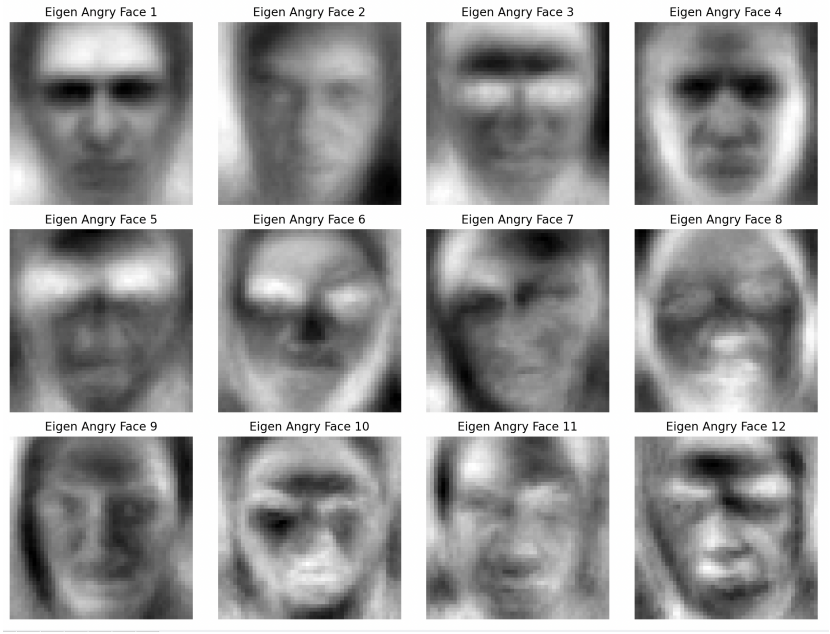


Figure 10: 12 handpicked angry eigen vectors

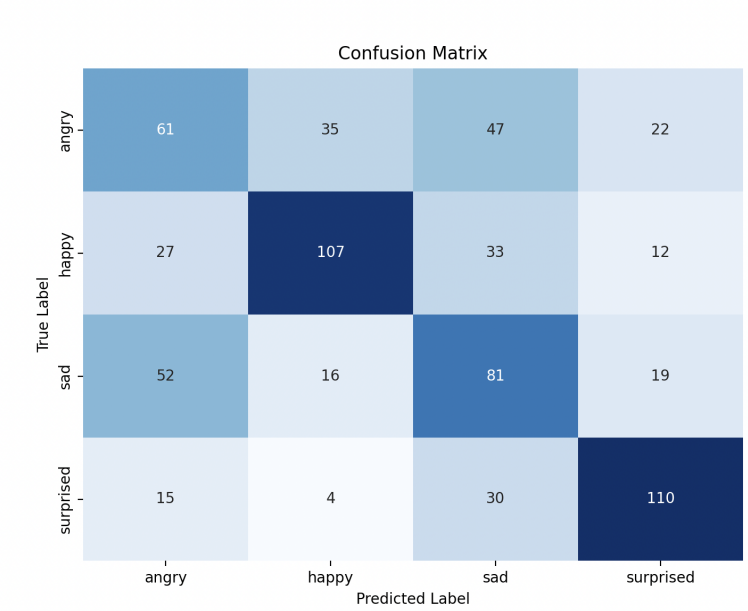


Figure 11: Confusion matrix for handpicking method

(a) High quality happy eigen vector

(b) Low quality sad eigen vector

Figure 12: Comparison of eigen vectors

The third approach, our scoring system (which we derived aka Omer's score), introduces a quantitative measure to guide the selection of eigen vectors. This system evaluates the correlation of each eigen vector with images of the target emotion compared to images from other emotions. By prioritizing eigen vectors with higher correlation to the target emotion and lower correlation to other emotions, we aim to identify discriminative features specific to each emotion category. The complete scoring system is specified in section 3.3.3. We also wanted to state that in each method we have a hyper parameter that we could change, each of the hyper parameters can be an analysis section by itself and that it is why we chose to focus on the three methods with fixed hyper parameters.

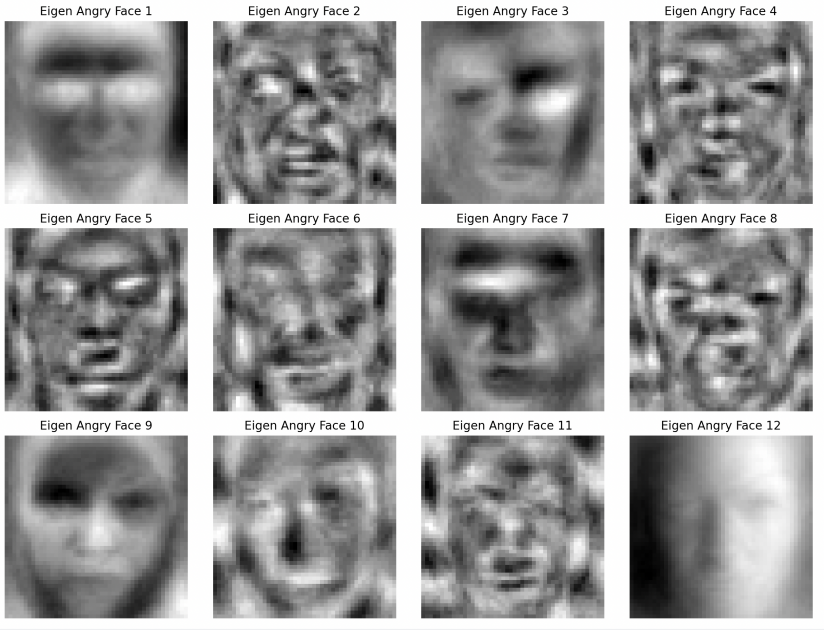


Figure 13: 12 Eigen faces chosen by the scoring system

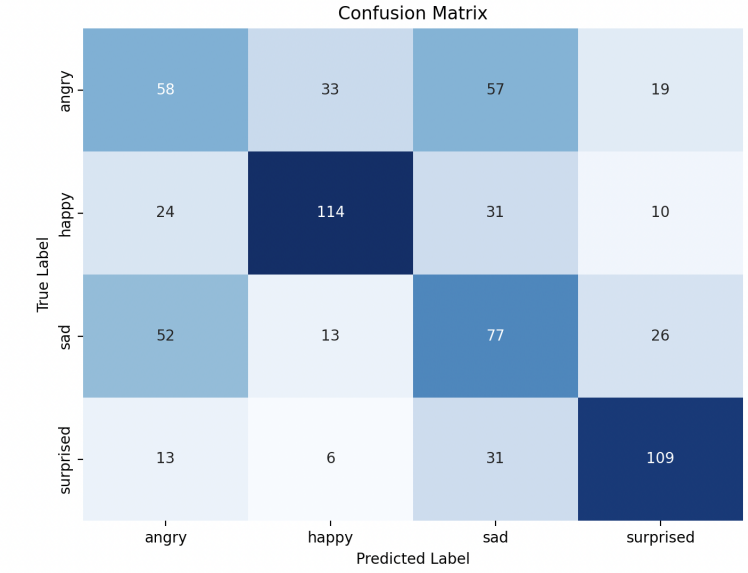


Figure 14: Confusion matrix for scoring method

4.1 Conclusions

In comparing these approaches, several considerations arise. The threshold-based method offers a straightforward and automatic way to select eigen vectors, leveraging mathematical principles to determine relevance. However, it may overlook subtle yet crucial facial expressions that contribute to emotion recognition.

On the other hand, the manual selection process provides a more intuitive approach, allowing for direct consideration of facial features. By visually inspecting eigen vectors, we can identify subtle nuances in expression that may be missed by automated methods. However, this approach is subjective and may introduce bias based on individual interpretation.

The scoring system offers a balance between automation and intuition, providing a quantitative measure of relevance while still considering the discriminative power of facial features. By prioritizing eigen vectors based on their correlation with target emotions, this approach aims to capture both the statistical significance and the perceptual relevance of facial expressions.

From the confusion matrices, we can observe that it is challenging for the model to distinguish between sad and angry emotions with all three methods. In the scoring method, we achieve the best distinguishing of happy emotions and almost the best for surprised emotions. In the "95%" method, we have the highest accuracy of classification for the angry emotion. In the handpicked method, we find a middle ground between the other two methods, meaning better results in classifying happy and surprised emotions, better than the "95%" method, but the angry emotion is worse. Additionally, the angry and sad emotions in the handpicked method are better than the scored method, but the happy emotion is worse.

In evaluating these approaches, we consider factors such as classification accuracy, computational efficiency, and real-time performance when tested on people in the library where we presented our project. Real-time testing in the library setting was instrumental in determining the effectiveness of each method in practical scenarios, providing valuable insights into their real-world applicability.

Moreover, the scored and handpicked methods exhibit significantly higher memory efficiency compared to the 95% method, requiring approximately ten times less memory while maintaining comparable or even superior performance. This improved memory efficiency further enhances the practicality and scalability of these methods for real-world applications.

In conclusion, while examining the confusion matrix of each option does not decisively indicate the best method, real-time testing on individuals in the li-

brary setting where our project was displayed revealed that the scoring system yielded the best results. This practical validation underscores the effectiveness of our scoring system in real-world scenarios and emphasizes its superiority in achieving accurate emotion classification.

5 Results

Eigen Emotions achieved promising results when tested in a real-world scenario at the library. The system worked effectively for the majority of users, particularly when expressing happy and surprised emotions. Occasionally, users were prompted to show teeth for precise classification of the happy emotion and to open their mouths for the surprised emotion, enhancing the accuracy of detection.

For users expressing sad or angry emotions, approximately half of them obtained good results, while around 30% were prompted to express their emotions more extravagantly for improved detection. However, for about 20% of users, the system encountered difficulty in accurately classifying one of the emotions (angry or sad).

These results showcasing the effectiveness of classical image processing algorithms and high-level linear algebra in achieving satisfying results, even in tasks typically associated with deep learning techniques.

6 References

References

[1] Matthew A Turk and Alex P Pentland. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition*, pages 586–587. IEEE Computer Society, 1991.

[2] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001.
[2]
[1]

Figure 15: First try of face tracking algorithm