**EECE 5626** **Image Processing and Pattern Recognition** **Fall 2021**

# Classification of Human Emotion using DCT Coefficients

Carolina Binns, *B.S. in Computer Engineering (2021)*
Tyler McKean, *B.S. in Electrical Engineering (2021)*
December 10, 2021

**Department of Computer and Electrical Engineering**
**Northeastern University**

## ABSTRACT

**As a critical method of human communication, facial expression of emotions is a perfect candidate for image classification research. Using the MATLAB Classification Learner, we will explore how Discrete Cosine Transform (DCT) Coefficients can be used as features to identify the emotion being expressed from a facial image. Two datasets, the Japanese Female Facial Expression Dataset (JAFFE) and the Extended Cohn-Kanade Dataset (CK+), are used to train and test this method. The images will be preprocessed with a variety of methods including grayscale conversion, feature identification, and image cropping. Once the images are preprocessed, the Discrete Cosine Transform is performed and its coefficients are extracted and used to train an optimized KNN classifier. The trained model is used to test images from the dataset that were left out of the training set. The DCT method was successful in identifying emotions, with testing accuracy levels ranging from 58%-100% depending on the number of DCT coefficients and regions of interests used.**

# Contents

# I. Introduction

Before humans developed the very first spoken languages, we relied on nonverbal cues to guide our communication - from facial expression to posture to bodily gestures. Despite the development of language around 150,000 years ago, 55% of modern human communication remains nonverbal [1]. In this project, we will focus specifically on facial expressions. As a prominent part of emotion recognition, facial expression has inspired much research in the field of image classification.

One way of identifying emotions in images is by calculating Discrete Cosine Transform (DCT) Coefficients for the face region or eye and mouth regions of the image. DCT Coefficients are useful for identifying unique points within an image, as they convey information from the spatial representation of the image into its corresponding frequency domain representation. [2] In this project we will analyze both methods by using these coefficients to train an optimized KNN classifier in the MATLAB Classification Learner Application as part of the Deep Learning Toolbox. The first steps in image classification involve preprocessing, which ensures that the images are as compatible with the classification method as possible. First, the images are converted to grayscale, as color does not hold any significance and should be removed as a factor. Then the MATLAB Cascade Object Detector is used to identify the region of interest from the image that contains the face. [3] If only the eyes and mouth are being used for classification, then the next step would be to crop the eyes and mouth into two separate regions. We created our own scripts that are located in Appendix A that utilize these methods to do this, as the Cascade Object Detector was not accurate enough to use on its own. In combination with the object detection, we use a few conditional statements specifically suited to the data set to correctly identify the eye and mouth regions. Throughout the project, we experimented with a variety of other preprocessing methods, but ultimately did not include them in our test results, as the effects were negligible.

After preprocessing is complete, the matrix of coefficients for all training images is loaded into the Classification Learner to train a KNN (Kth-Nearest Neighbors) classifier. Once trained, this classifier will result in a model that can be used to classify test images. Ultimately, the goal of this project is to evaluate the feasibility of DCT coefficients as a method for emotion classification, to compare the results using different datasets and different facial regions. Our

method, in the best case, resulted in 84.4% training accuracy and 100% test accuracy when testing with images from the same dataset. This was using the JAFFE dataset, which performed much better than the CK+ dataset, which was composed of older, lower quality images. A flowchart overviewing our process can be seen in the figure below.

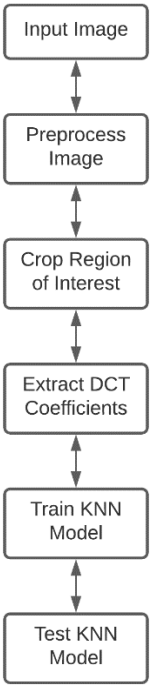


*Figure 1 - Flowchart diagram of proposed Emotion Classification method*

# II. Preprocessing Images

## A. Selection of Datasets

To train and test our method, we identified several datasets that were good candidates for our project. The features of an ideal dataset were:

1. Large number of images (100+)
2. Images were taken in a constant setting (i.e. background is solid, subjects are in front of the same background)
3. High quality images (no geometric or illumination bias)

Though we found many suitable datasets, the limiting factor was gaining access to these datasets. Because they included pictures of people, many of the datasets required forms to be submitted or for the person accessing the dataset to be employed by a research institution. Unfortunately, this meant choosing datasets that were lower quality or smaller in size. Due to the time frame of this

project, labeling training images by hand was not feasible, so we also needed to select pre-labeled datasets. We selected the Japanese Female Facial Expression Dataset (JAFFE) and the Extended Cohn-Kanade Dataset (CK+) [4][5]. JAFFE contains 212 labeled images: 30 anger, 29 disgust, 32 fear, 31 happiness, 30 neutral, 30 sadness, and 30 surprise. The subjects of these images are 10 Japanese women. The CK+ dataset contains 270 labeled images: 45 anger, 59 disgust, 25 fear, 69 happiness, 28 sadness, 44 surprise. The subjects of these images are 123 men and women, ranging in age from 18 to 50. The main drawback of the CK+ dataset is the quality of the images. Many of the images have poor lighting that obscures the facial features responsible for showing emotion. For example, the following below image is so dark that the eyes are obscured. Because the quality is different for each image, preprocessing to address these issues would need to be done on a case-by-case basis − which is not realistic for a dataset of this size. This is one reason why the JAFFE dataset, which has more stable illumination intensities, is better suited for the DCT Coefficient method.

*Figure 2- Example Image from the CK+ Dataset*

However, the JAFFE dataset was not perfect either. In Figure 3, the image on the left shows an example of a high-quality training image. The subject is centered in the frame, in front of a plain background. The lighting is not so bright or so dark that the facial features cannot be seen. In the center, we see an image that may require additional preprocessing because of an illumination bias — the right side of the expressor's face is obscured by a shadow. The image on the right displays a geometric bias in which the face is tilted to the right. These biases need to be accounted for in the preprocessing step in order to achieve maximum performance from a dataset.

*Figure 3 - Example Images from the JAFFE Dataset*

## B. Preprocessing Methods

The most common types of images used for automatic expression recognition are 2-D monochrome (grayscale) facial images [2]. Thus, the first step in preparing a dataset for training is converting all images to grayscale. We accomplished this with the built-in MATLAB function rgb2gray, which can see in Appendix B incorporated into our MATLAB functions. For the JAFFE dataset, which is in grayscale by default, this step is not necessary. For CK+, the dataset contains a mix of images that are in both RGB and grayscale, so we apply this function to ensure no RGB images are used in training. Figure 4 shows an example of this step being applied to an image from the CK+ dataset.

*Figure 4 - MATLAB RGB to Grayscale Conversion Example Image*

Once converted to grayscale, the MATLAB Cascade Object Detector was used to identify regions of interest − either the facial region or eye and mouth regions. The Object Detector comes pre-trained as part of MATLAB's Computer Vision Toolbox and uses the Viola-Jones algorithm to detect areas of interest like eyes, noses, faces, and mouths [6]. Viola-Jones uses

AdaBoost to train a classifier based on the sum of values in subwindows of an image. This algorithm builds on the concept of Haar Features, shown in Figure 5, which note that facial features share common properties [7][8]. We can think of these rectangular images of patterns that show up in images.

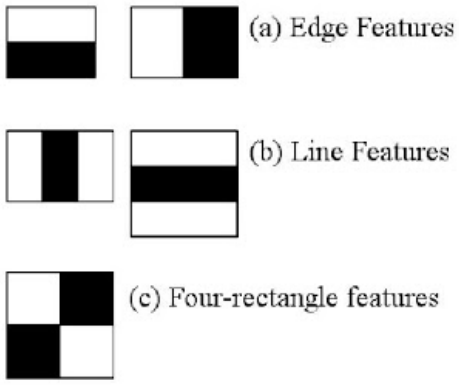


*Figure 5 - Haar Features*

For example, we can see in Figure 6 that the bridge of a nose is similar to an edge feature, in that the bridge of the nose is lighter than the sides of the nose. This is the concept that is applied in the Viola-Jones algorithm.

*Figure 6 - Example of Identifying Haar Features within a Face*

While the intensities in real images are not 0 and 1, like the Haar features, we can still see the similarity. In order to represent these features numerically, so that they can be used to train a classifier, Viola-Jones computes the integral image for each feature. The integral image is calculated by iteratively computing each pixel in the image as the sum of all pixels to the left/above the given pixel [6]. The result of the Viola-Jones algorithm is the trained classification

model, *vision.CascadeObjectDetector*, that MATLAB provides. To identify eye and mouth regions within an image, the code at the end of Appendix A results in the images shown below.

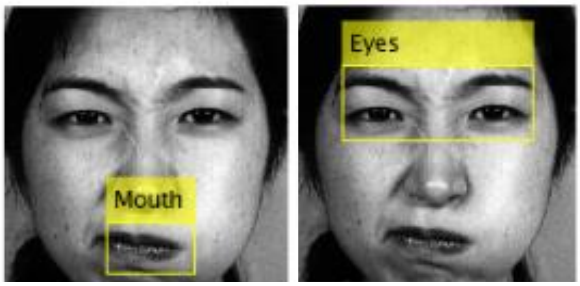


*Figure 7 - Example Face Annotation from Eye and Mouth Detection Using the CascadeObjectDetector*

Once the regions of interest are identified, the image can be cropped to include only these regions.

One challenge in the preprocessing step is the possibility of multiple bounding boxes being returned, i.e. more than one face, pair of eyes, or mouth is identified in the image. In our dataset, we know that this is not the case and that only one of the bounding boxes is valid. In order to resolve this issue, we used knowledge of each specific database to create conditional statements to identify the correct bounding box.

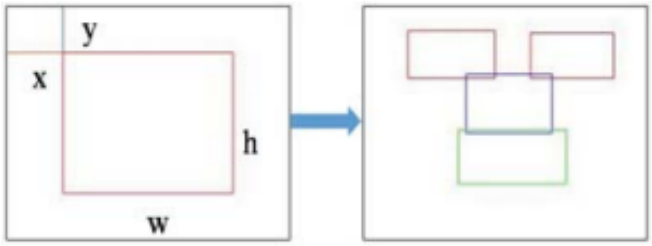


*Figure 8 – Selection of Bounding Boxes based on Pixel Coordinates of Image[13]*

For example, we know that in our datasets, the face takes up the majority of the image, so we choose the largest bounding box, as in the left image of Figure 8 above. Similarly, we know that the nose is often misidentified as the mouth, so choosing the bounding box lowest in the image coordinates such as the green bounding box in the right image of Figure 8 will prevent this from happening. Of course, these methods are not guaranteed to work for every case, so we follow this step up by visually confirming that the results are accurate and manually fixing any errors.

# III. Feature Extraction

## A. Discrete Cosine Transform

To generate an accurate model for predicting different classifications, we must select the appropriate features that will represent and distinguish each emotion from the datasets. As the title of the paper suggests, we chose to use the 2-D Discrete Cosine Transform function because of its properties, which are similar to the Fourier and Hartley Transforms, except its spectrum contains twice the frequency resolution across the same frequency range as these other transforms [9]. The Discrete Cosine Transform expresses an array of finitely many data points into terms of a summation of cosine functions with different frequencies of which it oscillates. Eq. (1) defines the 2-D DCT of an $M \times N$ matrix **A** as:

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} cos\left(\frac{(2m+1)u\pi}{2M}\right) cos\left(\frac{(2n+1)v\pi}{2N}\right) \tag{1}$$

with the spectrum components $p, q$ in the range: $0 \leq p \leq M-1$, $0 \leq q \leq N-1$ that accounts for the number pixels in each row and column of the 2-D matrix **A**. The constant components outside of the summation are defined as:

$$\alpha_p = \begin{cases} \frac{1}{\sqrt{M}}, & p = 0 \\ \sqrt{\frac{2}{M}}, & 1 \leq p \leq M-1 \end{cases} \tag{2}$$

$$\alpha_q = \begin{cases} \frac{1}{\sqrt{N}}, & q = 0 \\ \sqrt{\frac{2}{N}}, & 1 \leq q \leq N-1 \end{cases} \tag{3}$$

Where the values of $B_{pq}$ generated by this function are called the DCT Coefficients. A DCT basis function of maximum frequency can be generated with *p* and *q* equal to 7 and can be used in 1-D or 2-D computations [9]. A 2-D DCT basis image is shown in Figure 9 with $N = 8$.

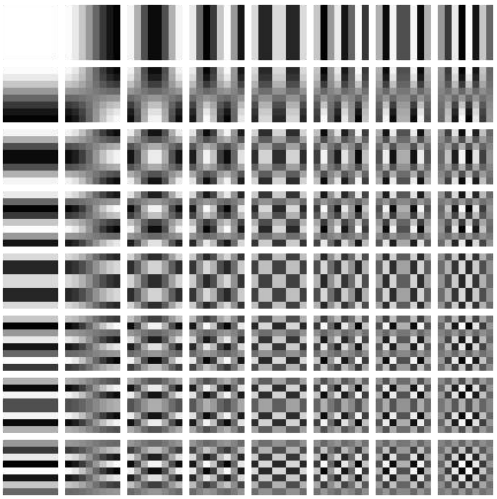

*Figure 9 - 2D DCT Basis Function*

The 2-D DCT is considered to be the real and orthogonal relative to the Discrete Fourier Transform but features a $2N$ - periodicity and even symmetry that minimizes discontinuities and "artificial" high-frequency components [9]. These features make the Discrete Cosine Transform the ideal method when working with image compression, such as in the JPEG compression format.

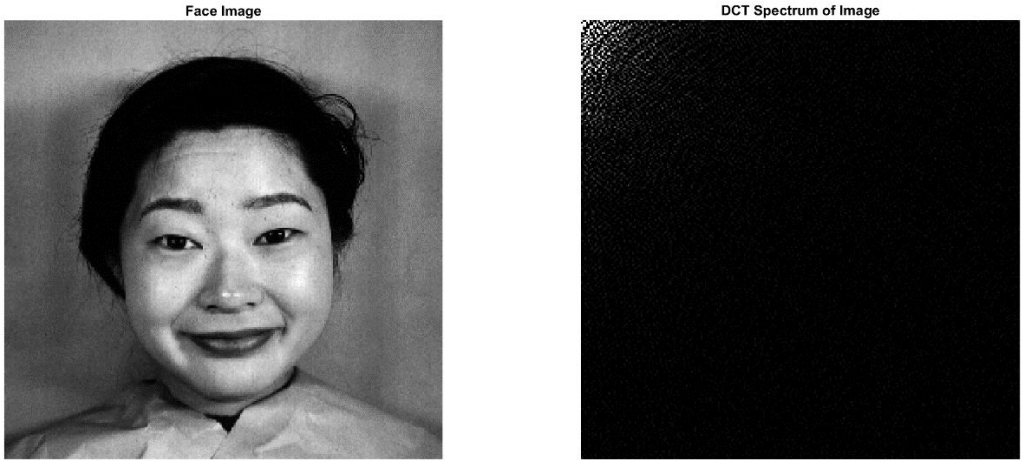


*Figure 10 - JAFFE Database Image and Spectrum of Discrete Cosine Transform*

An image from the JAFFE dataset and its spectrum are displayed in Figure 10. The spectrum image contains all the frequency component magnitudes for which a summation of cosines can describe the image on the left. The top left corner of the spectrum constitutes the DC component or the zero-frequency component, which contains the average amount of energy of the image. The upper left corner of the spectrum represents the lower frequencies in the frequency domain.

The higher frequencies, or AC components, are linearly spaced and move in a zig-zag sequence from the top left to the bottom right of the spectrum image. The unique feature of the 2-D DCT is that the lower frequency magnitudes capture the most relevant information about an image [10]. These lower frequency magnitudes can also be used in reconstructing an image after eliminating redundancy in the high frequency components.


*Figure 11 - Original Image and Reconstructed Image using only 1000 DCT Coefficients*

The images shown in Figure 11 above display the lossy compression feature of the Discrete Cosine Transform. The reconstructed image was created in MATLAB by using a mask of size $N \times N$ to extract a subsection of the DCT coefficients in the upper left corner of the spectrum. In the case of the reconstructed image in Figure 11, the value of $N = 100$ was used to extract 1000 coefficients from the original spectrum and then was reconstructed using the Inverse Discrete Cosine Transform in Eq (4).

$$A_{mn} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \alpha_p \alpha_q B_{pq} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N}, \quad \begin{matrix} 0 \le m \le M-1 \\ 0 \le n \le N-1 \end{matrix}, \tag{4}$$

where again,

$$\alpha_p = \begin{cases} \frac{1}{\sqrt{M}}, & p = 0 \\ \sqrt{\frac{2}{M}}, & 1 \le p \le M-1 \end{cases} \tag{5}$$

$$\alpha_q = \begin{cases} \frac{1}{\sqrt{N}}, & q = 0 \\ \sqrt{\frac{2}{N}}, & 1 \le q \le N-1 \end{cases} \tag{6}$$

You can see even with removing more than half of the frequency information of the DCT, the original image is still intact and contains a slight blur to it, which equates to low pass filtering the image. By utilizing these properties of the DCT, the image compression of JPEG is achievable by reducing the redundancy of the images and lowers the file size of images while keeping the visual discrepancies at a negligible margin. It is with these aspects of the DCT that compact the energy of an image into the lower end of the spectrum for which we found suitable for use in classifying different emotion images.

## B. Feature Selection

After the preprocessing methods described in the previous sections were completed, the face and eye/mouth regions would have the 2-D DCT performed on all three regions in order to extract the lower frequency features used as inputs for image classification. The following figures below show an example of the preprocessed images with their respective DCT spectrums.

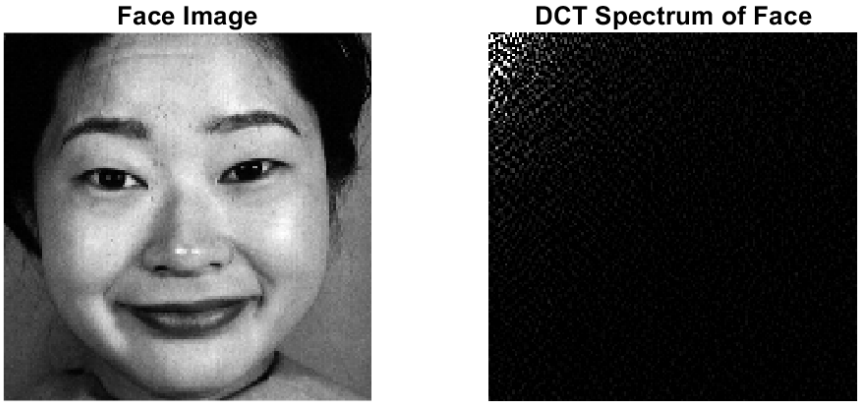


*Figure 12 - Cropped Face ROI and DCT Spectrum*


*Figure 13 - Cropped Eye ROI and DCT Spectrum*

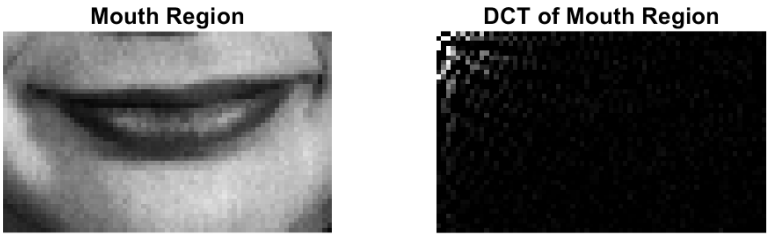


*Figure 14 - Cropped Mouth ROI and DCT Spectrum*

Once these regions of interest were cropped from the preprocessing methods, only the lower frequency components needed to be extracted from the DCT spectrum to serve as features into our classification model. To determine how many DCT coefficients were appropriate to achieve an accurate classification model, we decided to test various amounts of extracted DCT coefficients. Thus, we devised MATLAB scripts that would extract $n$ number of DCT coefficients from the spectrum, which can be seen in Appendix A. Once the DCT was computed on a region, a subsection of size $n \times n$ starting with the DC component in the top left corner was saved to a matrix variable. To keep the frequency components linear in the extracted feature vectors, we used a zigzag function [11] in MATLAB that unravels the $n \times n$ coefficient matrix in the same manner as the figure below to create a single row vector of DCT coefficients for each facial image.

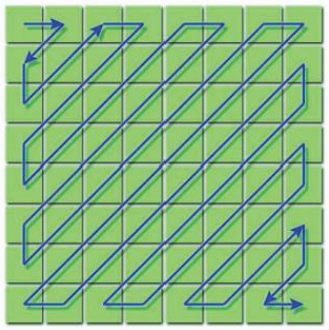

*Figure 15 - Zigzag sequence for DCT Coefficient Extraction [12]*

With this method we could extract $n^2$ coefficients from each region and experiment with how many lower frequency components were needed for an acceptable accuracy level for our classification models. In addition to testing with varying numbers of coefficients, we also experimented with training classification models using the DCT coefficients of just the face region and comparing them to the coefficients of the eye/mouth regions as a separate classification model. For example, the feature vectors for the face region of the JAFFE dataset

would comprise of *M* images used to train the dataset with *n* extracted features, resulting in a feature table of size $M \times n$ used as input to the MATLAB Classification Learner App. When using the eye/mouth regions of the datasets, the feature vectors containing *n* coefficients for each region would be appended together to create a single feature vector that was $2 * n^2$ long and described one face emotion image.

# IV. Image Classification

For Image classification, many models were seen as viable options, but ultimately, we decided to use a Kth-Nearest Neighbor classification for our method. We initially thought to use a Neural Network for image classification, but these classification models require large training sample sizes and require computationally long execution times to acquire accurate results [13]. A KNN model, would be computationally less complex, and faster for us to quickly train and test several different classification models.

After the DCT Coefficients have been compiled into a 2-D feature matrix, the data is loaded directly into the MATLAB Classification Learner from the workspace. The loaded data is displayed in a scatter plot, as shown in Figure 7, with each point representing one image in the training set.

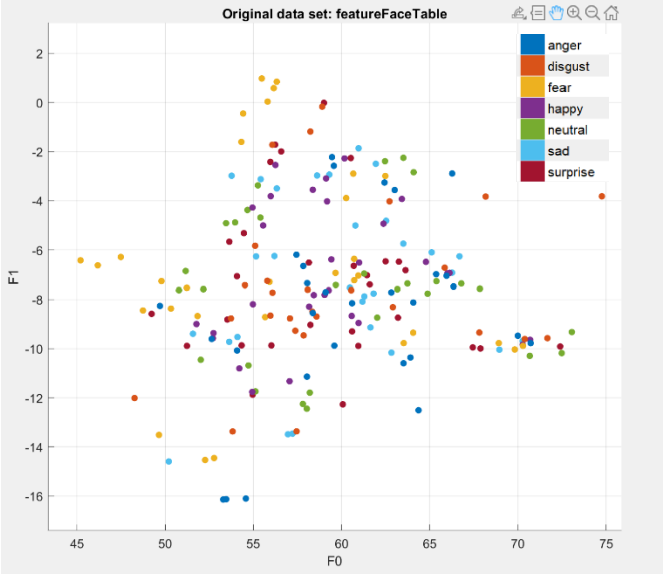


*Figure 16 – Scatter Plot Image from MATLAB Classification Learner App with different colored dots representing the different emotions*

From there, the Optimizable KNN Classifier was selected and trained with the default optimization parameters: Bayesian optimization with 30 iterations. The K-Nearest Neighbor algorithm is a supervised classification method that is based on the nearest-neighbor classifier, also known as the minimum distance classifier [6], which matches an unknown object with the class of the datapoint that it is closest to. The closest point can be calculated by several different methods, including Euclidean, Correlation, Hamming, Spearman, and more. The Euclidean distance, for example, is described by the following equation from Gonzalez and Woods [9]:

$$D_j(x) = ((x - m_j) * (x - m_j)^T)^{1/2}, \ j = 1, 2, 3, \dots, N_c \tag{7}$$

In this equation, $m_j$ is the mean vector of the $jth$ class. For KNN, the same process is used, but the input is compared to one or more nearest neighbors. The output class is simply the class with the highest frequency among the k-nearest neighbors to the input. The benefit to using the optimizable KNN classifier is that many different distance metrics will be used, and the trained model will use whichever results in the highest training accuracy, so we did not need to worry about selecting this method manually.

Once the classifier was trained, we exported the model back into our workspace where we used it to test images. In the training process, we left out some images from the training set so that we could use them for testing. To identify the emotion expressed in a given image, the same preprocessing methods used in testing were applied and the DCT coefficients were calculated. The MATLAB code located in Appendix A shows the process for testing multiple emotion images, where the *testTable* variable refers to the testing matrix of DCT coefficients with the variable *testResults* containing a column vector of the predicted emotions for each test image.

# V. Results and Comparisons

JAFFE Dataset

| Model (Face) | # of DCT Coefficients | Training Accuracy | Testing Accuracy | Model (Eye/Mouth) | # of DCT Coefficients | Training Accuracy | Testing Accuracy |
|---|---|---|---|---|---|---|---|
| KNN | 16 | 73% | 71.43% | KNN | 16 | 75.57% | 78.57% |
| KNN | 25 | 83.9% | 71.43% | KNN | 25 | 79.80% | 100% |
| KNN | 36 | 81.5% | 100% | KNN | 36 | 79.90% | 91.67% |
| KNN | 49 | 84.4% | 85.71% | KNN | 49 | 77.90% | 91.67% |
| KNN | 64 | 80.5% | 100% | KNN | 64 | 78.30% | 100% |

*Table 1 – Testing Results for the JAFFE Dataset*

CK+ Dataset

| Model (Face) | # of DCT Coefficients | Training Accuracy | Testing Accuracy | Model (Eye/Mouth) | # of DCT Coefficients | Training Accuracy | Testing Accuracy |
|---|---|---|---|---|---|---|---|
| KNN | 16 | 45.2% | 33.3% | KNN | 16 | 64.8% | 58.0% |
| KNN | 25 | 50.4% | 58.3% | KNN | 25 | 70.3% | 83.3% |
| KNN | 36 | 50% | 25% | KNN | 36 | 67.0% | 58.0% |
| KNN | 49 | 50.8% | 50% | KNN | 49 | 73.0% | 58.33% |
| KNN | 64 | 56% | 25% | KNN | 64 | 72.2% | 66.67% |

*Table 2 – Testing Results for the CK+ Dataset*

Results for both datasets using both the Face regions and Eye/Mouth regions can be seen above in Table 1 and Table 2. Each of the Training Accuracy measurements for all the models were derived as a result of training the KNN model in the MATLAB Classification Learner App using the input feature table that contained the feature vectors for the emotion images. Rather than using completely different images for testing, each of the trained KNN classification models were tested with randomly selected images that were left out from their respective training dataset. As for the case of the JAFFE dataset, there were only 212 images total. We wanted to be able to train the models with as much training data as possible, so we only left out 2 images for each emotion, leaving 14 emotion images to test and 198 images for training. When deriving a classification model using only the Face region, the accuracy varied between 73 – 84% when training the model in the MATLAB Classification Learner App. Once the models were exported to the workspace in MATLAB, each of the 14 emotion images were fed into the prediction function inside the struct datatype. The testing accuracies ranged from 71.43% to 100% in classifying the 14 emotion images. Across all of these tests, the emotion images for Happiness, Sadness, and Neutral were classified perfectly, with Fear being the emotion that was

misclassified the most. A Confusion Matrix is shown in Figure 17 to displays which classes were correctly predicted. When using the Eye/Mouth regions of the JAFFE dataset, the accuracy ranged from 75.8 – 79.8% in accuracy when varying the number of DCT coefficients. The highest accuracy when varying the number of DCT coefficients for the Face region was $N = 7$ or 49 coefficients and $N = 6$ or 36 coefficients for the Eye/Mouth regions.

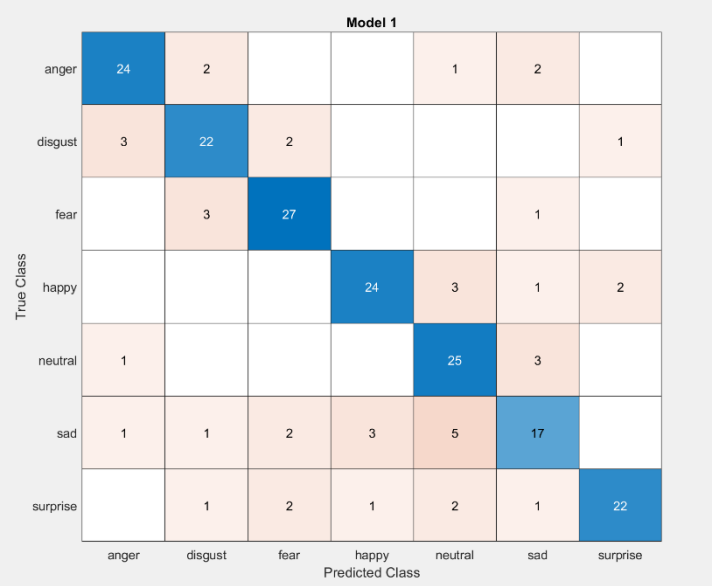


*Figure 17 - Confusion Matrix for extracted Face Regions of JAFFE Dataset with N = 8*

For the CK+ dataset, the extracted Eye/Mouth regions performed better with a training accuracy range of 64.8% to 73% when increasing the number of DCT coefficients from $N = 4$ to $N = 8$, with $N = 7$ resulting in the highest accuracy. Although these are not the best accuracy results, the reason these regions provided more accurate predictions than just the Face region was because of the variety of faces between the expressor of this dataset. When taking the DCT coefficients of the whole face, the accuracies ranged from 45.2% to 56% when training the KNN models. Since the isolated regions of just the eyes and mouth were used to extract features, the differences between faces of the expressors was not as much of an influence on the accuracy.

After experimental results were drawn, we assessed some of the reasons that lead to our misclassifications and what steps could have been made to improve our accuracy results. For the JAFFE dataset, the advantage of only having 10 different female expressors with very little illumination or geometrical bias, gave us some decent results for accuracy using our DCT feature extraction method with a KNN classification model. However, accurate image classification

methods insist on having large datasets to both train and test with [12]. Having only about 30 images per emotion gave us results in the 70-84% accuracy range, and given more images could have led to better accuracy scores. Another method for increasing the accuracy would involve the process of subtracting a neutral image from all of the emotion images [15]. Using this difference image would highlight expression from the eyebrows and mouth regions that could have led to more accurate features for classification. For the CK+ dataset, we did not anticipate the differences in faces being as much of an issue on our accuracy. One solution we discovered late into this project was to create an average of the faces using an approach called "eigenfaces" [16] that could subtract the differences between all the faces and ensure we only extract DCT coefficients from the expression differences from the faces, rather than the difference amongst each expressor. Factors like difference in gender, skin tone, age, and even hairstyle add complexity to the classification process [14]. These are factors we believe lead to the satisfactory results in our accuracy for this dataset.

# VI. Conclusions

In this report we've demonstrated a facial emotion classification technique using the Discrete Cosine Transform coefficients for the face, eyes, and mouth regions of an image. Two datasets were used in our experiments that consisted of about 200 images for the JAFFE dataset, and about 300 images for the CK+ dataset that were used. We successfully preprocessed these images by first converting RGB images to grayscale, extracting the Haar features such as the face, eyes, and mouth regions of an image, and cropped these regions of interests. Once all three regions were cropped, the face, eyes, and mouth images for both datasets were converted from the spatial domain representation to their respective frequency domain spectra by performing the 2-D Discrete Cosine Transform. Varying numbers of DCT coefficients were extracted and used to compare KNN classification models derived from just the face images and just the eye and mouth images. Tests were conducted by using two random images from each dataset that were left out intentionally in an attempt to train the models with as much images as possible. A comparison was drawn from the results of the classification results based on whether the face region or eye and mouth regions were used, along with the accuracy from the varying number of DCT coefficients. Factors like size of the image dataset, the differences between expressors, and the low-quality of the images were considered as the primary reasons that lead to

misclassification and poor accuracy results. Though this method resulted in same adequate classification accuracy, steps such as deriving an average face or subtracting a neutral expression from the emotion images would lead to a more accurate classification model.

# References


[1] Mehrabian, Albert; Ferris, Susan R. (1967). "Inference of Attitudes from Nonverbal Communication in Two Channels". Journal of Consulting Psychology. 31 (3): 248–252.

[2] N. Thomas and M. Mathew, "Facial expression recognition system using neural network and MATLAB," *2012 International Conference on Computing, Communication and Applications*, 2012, pp. 1-5, doi: 10.1109/ICCCA.2012.6179169.

[3] MathWorks, “vision.CascadeObjectDetector,” 2012. [Online]. Available: https://www.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-system-object.html

[4] Michael J. Lyons, Miyuki Kamachi, Jiro Gyoba. Coding Facial Expressions with Gabor Wavelets (IVC Special Issue) arXiv:2009.05938 (2020) https://arxiv.org/pdf/2009.05938.pdf

[5] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar and I. Matthews, "The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression," 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, 2010, pp. 94-101, doi: 10.1109/CVPRW.2010.5543262.

[6] MathWorks, “Train Nearest Neighbor Classifiers Using Classification Learner App.” [Online]. Available: https://www.mathworks.com/help/stats/train-nearest-neighbor-classifiers-in-classification-learner-app.html

[7] Viola, Paul and Michael J. Jones, " Rapid Object Detection using a Boosted Cascade of Simple Features" , Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001. Volume: 1, pp.511–518.

[8] OpenCV, “Cascade Classifier.” [Online]. Available: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

[9] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th ed. New York, NY: Pearson, 2018

[10] [5] L. Ma, Y. Xiao, K. Khorasani and R. K. Ward, "A new facial expression recognition technique using 2D DCT and k-means algorithm," *2004 International Conference on Image Processing, 2004. ICIP '04.*, 2004, pp. 1269-1272 Vol.2, doi: 10.1109/ICIP.2004.1419729.

[11] Damilola Ogunbiyi (2021). Zig-Zag scan (https://www.mathworks.com/matlabcentral/fileexchange/27078-zig-zag-scan), MATLAB Central File Exchange. Retrieved December 10, 2021.

[12] S. K. Gupta, S. Agrwal, Y. K. Meena and N. Nain, "A Hybrid Method of Feature Extraction for Facial Expression Recognition," *2011 Seventh International Conference on Signal Image Technology & Internet-Based Systems*, 2011, pp. 422-425, doi: 10.1109/SITIS.2011.64.

[13] F. Mahmud, B. Islam, A. Hossain and P. B. Goala, "Facial Region Segmentation Based Emotion Recognition Using K-Nearest Neighbors," *2018 International Conference on Innovation in Engineering and Technology (ICIET)*, 2018, pp. 1-5, doi: 10.1109/CIET.2018.8660900.

[14] R. Nannapaneni and S. Chatterjee, “Human Emotion Recognition Through Facial Expressions,” Algorithms for Intelligent Systems, pp. 513–525, 2021.

[15] L. Ma, Y. Xiao, K. Khorasani and R. K. Ward, "A new facial expression recognition technique using 2D DCT and k-means algorithm," *2004 International Conference on Image Processing, 2004. ICIP '04.*, 2004, pp. 1269-1272 Vol.2, doi: 10.1109/ICIP.2004.1419729.

[16] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1991, pp. 586-591, doi: 10.1109/CVPR.1991.139758.

# Vita

Carolina Binns completed her BS in Computer Engineering at Northeastern University in 2021, through the NSF CyberCorps Scholarship for Service program. She is currently an Associate Wireless Communications Security Engineer at the MITRE Corporation. Alongside her work at MITRE she is pursuing a Master’s degree in Electrical and Computer Engineering with a concentration in Communications, Control, and Signal Processing at Northeastern University.

Tyler McKean completed his BS in Electrical Engineering at the University of Massachusetts - Boston in the Spring of 2021. He is currently working for Panasonic North America as an Electrical Engineering Co-op while pursuing his Master’s of Science in Electrical and Computer Engineering with a concentration in Communications, Control, and Signal Processing at Northeastern University. He hopes to apply his knowledge from his MS program towards a career in the Audio and Health/Fitness Consumer Electronics Industries after completing his degree.

# Appendix

## Appendix A. MATLAB Scripts

### 1. Face Region Script

### Read Emotion Images into Workspace and Extract n DCT Coefficients from only the Face Regions

```
clc; clear all; close all;
```

Declare how many DCT coefficients (n) to extract

```
n = 8;
```

Load Anger Images

```
M = 30; % 30 Anger Images in JAFFE dataset

angerFaceTable = zeros(M,n^2);

for i = 1:M

    f = imread(['C:\Users\pc\Documents\MATLAB\NEU\Image Processing\Final Project Files\JAFFE\anger\AN' num2str(i) '.tiff']);

    angerFaceTable(i,:) = extractDCT_Face(f,n);

end
```

Load Disgust Images

```
M = 29; % 29 Disgust Images in JAFFE dataset

disgustFaceTable = zeros(M,n^2);

for i = 1:M

    f = imread(['C:\Users\pc\Documents\MATLAB\NEU\Image Processing\Final Project Files\JAFFE\disgust\DI' num2str(i) '.tiff']);

    disgustFaceTable(i,:) = extractDCT_Face(f,n);

end
```

Load Fear Images

```
M = 32; % 32 Fear Images in JAFFE dataset

fearFaceTable = zeros(M,n^2);

for i = 1:M

    f = imread(['C:\Users\pc\Documents\MATLAB\NEU\Image Processing\Final Project Files\JAFFE\fear\FE' num2str(i) '.tiff']);
```

```
    fearFaceTable(i,:) = extractDCT_Face(f,n);
end
```

### Load Happiness Images

```
M = 31; % 31 Happiness Images in JAFFE dataset
happyFaceTable = zeros(M,n^2);
for i = 1:M
    f = imread(['C:\Users\pc\Documents\MATLAB\NEU\Image Processing\Final Project Files\JAFFE\happiness\HA' num2str(i) '.tiff']);
    happyFaceTable(i,:) = extractDCT_Face(f,n);
end
```

### Load Neutral Images

```
M = 30; % 30 Neutral Images in JAFFE dataset
neutralFaceTable = zeros(M,n^2);
for i = 1:M
    f = imread(['C:\Users\pc\Documents\MATLAB\NEU\Image Processing\Final Project Files\JAFFE\neutral\NE' num2str(i) '.tiff']);
    neutralFaceTable(i,:) = extractDCT_Face(f,n);
end
```

### Load Sadness Images

```
M = 30; % 30 Sadness Images in JAFFE dataset
sadFaceTable = zeros(M,n^2);
for i = 1:M
    f = imread(['C:\Users\pc\Documents\MATLAB\NEU\Image Processing\Final Project Files\JAFFE\sadness\SA' num2str(i) '.tiff']);
    sadFaceTable(i,:) = extractDCT_Face(f,n);
end
```

### Load Surprise Images

```
M = 30; % 30 Surprise Images in JAFFE dataset
surpriseFaceTable = zeros(M,n^2);
for i = 1:M
    f = imread(['C:\Users\pc\Documents\MATLAB\NEU\Image Processing\Final Project Files\JAFFE\surprise\SU' num2str(i) '.tiff']);
```

```
    surpriseFaceTable(i,:) = extractDCT_Face(f,n);
end
```

## Randomly Extract test image from each image set

```
% Extract random Anger Image and delete it from Test Images
r1 = randi(size(angerFaceTable,1));
TestImageAnger = angerFaceTable(r1,:);
angerFaceTable(r1,:) = [];
% Extract random Disgust Image and delete it from Test Images
r2 = randi(size(disgustFaceTable,1));
TestImageDisgust = disgustFaceTable(r2,:);
disgustFaceTable(r2,:) = []
% Extract random Fear Image and delete it from Test Images
r3 = randi(size(fearFaceTable,1));
TestImageFear = fearFaceTable(r3,:);
fearFaceTable(r3,:) = [];
% Extract random Happy Image and delete it from Test Images
r4 = randi(size(happyFaceTable,1));
TestImageHappy = happyFaceTable(r4,:);
happyFaceTable(r4,:) = [];
% Extract random Neutral Image and delete it from Test Images
r5 = randi(size(neutralFaceTable,1));
TestImageNeutral = neutralFaceTable(r5,:);
neutralFaceTable(r5,:) = [];
% Extract random Sad Image and delete it from Test Images
r6 = randi(size(sadFaceTable,1));
TestImageSad = sadFaceTable(r6,:);
sadFaceTable(r6,:) = [];
% Extract random Surprise Image and delete it from Test Images
r7 = randi(size(surpriseFaceTable,1));
TestImageSurprise = surpriseFaceTable(r7,:);
surpriseFaceTable(r7,:) = [];
```

All the Test Image Matrices are now 1 image less, which was extracted for the "Leave-One-Out" method for testing the trained classifier model

## Combine Extracted DCT Coefficients into FeatureTable with Emotion Labels

```
featureFaceTable = vertcat(angerFaceTable, disgustFaceTable, fearFaceTable,
happyFaceTable, neutralFaceTable, sadFaceTable, surpriseFaceTable);

featureFaceTable = array2table(featureFaceTable);

for i = 1:n^2

    fname = ['F',num2str(i - 1)];

    featureFaceTable.Properties.VariableNames(i) = {fname};

end

clear('fname');

% Add labels to the Emotion Class Cell Array

featureFaceTable.EmotionClass = cell(size(featureFaceTable,1),1);

featureFaceTable.EmotionClass(1:size(angerFaceTable,1)) = {'anger'};

featureFaceTable.EmotionClass(size(angerFaceTable,1)+1:size(angerFaceTable,1)+siz
e(disgustFaceTable,1)) = {'disgust'};

featureFaceTable.EmotionClass(size(angerFaceTable,1)+size(disgustFaceTable,1)+1:s
ize(angerFaceTable,1)+size(disgustFaceTable,1)+size(fearFaceTable,1)) = {'fear'};

featureFaceTable.EmotionClass(size(angerFaceTable,1)+size(disgustFaceTable,1)+siz
e(fearFaceTable,1)+1:size(angerFaceTable,1)+size(disgustFaceTable,1)+size(fearFac
eTable,1)+size(happyFaceTable,1)) = {'happy'};

featureFaceTable.EmotionClass(size(angerFaceTable,1)+size(disgustFaceTable,1)+siz
e(fearFaceTable,1)+size(happyFaceTable,1)+1:size(angerFaceTable,1)+size(disgustFa
ceTable,1)+size(fearFaceTable,1)+size(happyFaceTable,1)+size(neutralFaceTable,1))
= {'neutral'};

featureFaceTable.EmotionClass(size(angerFaceTable,1)+size(disgustFaceTable,1)+siz
e(fearFaceTable,1)+size(happyFaceTable,1)+size(neutralFaceTable,1)+1:size(angerFa
ceTable,1)+size(disgustFaceTable,1)+size(fearFaceTable,1)+size(happyFaceTable,1)+
size(neutralFaceTable,1)+size(sadFaceTable,1)) = {'sad'};

featureFaceTable.EmotionClass(size(angerFaceTable,1)+size(disgustFaceTable,1)+siz
e(fearFaceTable,1)+size(happyFaceTable,1)+size(neutralFaceTable,1)+size(sadFaceTa
ble,1)+1:size(angerFaceTable,1)+size(disgustFaceTable,1)+size(fearFaceTable,1)+si
ze(happyFaceTable,1)+size(neutralFaceTable,1)+size(sadFaceTable,1)+size(surpriseF
aceTable,1)) = {'surprise'};

% Convert Emotion Class to a Categorical type for Classifier Variable in

% Classification Learner App

featureFaceTable.EmotionClass = categorical(featureFaceTable.EmotionClass);
```

## Output Trained Classification Model to the Workspace

```
testTable =
vertcat(TestImageAnger,TestImageDisgust,TestImageFear,TestImageHappy,TestImageNeu
tral,TestImageSad,TestImageSurprise);

testTable = array2table(testTable);

for i = 1:m

    fname = ['F',num2str(i - 1)];

    testTable.Properties.VariableNames(i) = {fname};

end

clear('fname');

% Add labels to the Emotion Class Cell Array

testTable.EmotionClass = cell(size(testTable,1),1);

testTable.EmotionClass =
{'anger';'disgust';'fear';'happy';'neutral';'sad';'surprise'};

testTable.EmotionClass = categorical(testTable.EmotionClass);
```

### Test Model

```
testResults = trainedFaceModel8.predictFcn(testTable)
```

## 2. Eye/Mouth Region Script

## Read Emotion Images into Workspace and Extract n DCT Coefficients from Eye and Mouth Regions

### Declare how many DCT coefficients (n) to extract

```
n = 4;
```

Load Anger Images

```
M = 30; % 30 Anger Images in JAFFE dataset
angerEyesMouthTable = zeros(M,2*(n^2));
for i = 1:M
       f = imread(['C:\Users\cbinns\Documents\MATLAB\NEU\Image Processing\Final Project
Files\JAFFE\anger\AN' num2str(i) '.tiff']);
       FaceDetect = vision.CascadeObjectDetector;
       face_bboxes = step(FaceDetect, f);
       onebox = face_bboxes(1,:);
       for i=1:size(face_bboxes,1)
       if face_bboxes(i,4) > onebox(1,4)
              onebox = face_bboxes(i,:);
       end
       end
```

```
        % Crop face from original image
        f = (imcrop(f,onebox));
    angerEyesMouthTable(i,:) = extractDCT_Mouth(f,n);
end
```

Load Disgust Images

```
M = 29; % 29 Disgust Images in JAFFE dataset
disgustEyesMouthTable = zeros(M,2*(n^2));
for i = 1:M
        f = imread(['C:\Users\cbinns\Documents\MATLAB\NEU\Image Processing\Final Project
Files\JAFFE\disgust\DI' num2str(i) '.tiff']);
        FaceDetect = vision.CascadeObjectDetector;
        face_bboxes = step(FaceDetect, f);
        onebox = face_bboxes(1,:);
        for i=1:size(face_bboxes,1)
        if face_bboxes(i,4) > onebox(1,4)
                onebox = face_bboxes(i,:);
        end
        end
        % Crop face from original image
        f = (imcrop(f,onebox));
    disgustEyesMouthTable(i,:) = extractDCT_Mouth(f,n);
end
```

Load Fear Images

```
M = 32; % 32 Fear Images in JAFFE dataset
fearEyesMouthTable = zeros(M,2*(n^2));
for i = 1:M
        f = imread(['C:\Users\cbinns\Documents\MATLAB\NEU\Image Processing\Final Project
Files\JAFFE\fear\FE' num2str(i) '.tiff']);
        FaceDetect = vision.CascadeObjectDetector;
        face_bboxes = step(FaceDetect, f);
        onebox = face_bboxes(1,:);
        for i=1:size(face_bboxes,1)
        if face_bboxes(i,4) > onebox(1,4)
                onebox = face_bboxes(i,:);
        end
        end
        % Crop face from original image
        f = (imcrop(f,onebox));
    fearEyesMouthTable(i,:) = extractDCT_Mouth(f,n);
end
```

Load Happiness Images

```
M = 31; % 31 Happiness Images in JAFFE dataset
happyEyesMouthTable = zeros(M,2*(n^2));
for i = 1:M
        f = imread(['C:\Users\cbinns\Documents\MATLAB\NEU\Image Processing\Final Project
Files\JAFFE\happiness\HA' num2str(i) '.tiff']);
        FaceDetect = vision.CascadeObjectDetector;
        face_bboxes = step(FaceDetect, f);
        onebox = face_bboxes(1,:);
        for i=1:size(face_bboxes,1)
        if face_bboxes(i,4) > onebox(1,4)
                onebox = face_bboxes(i,:);
        end
        end
        % Crop face from original image
        f = (imcrop(f,onebox));
    happyEyesMouthTable(i,:) = extractDCT_Mouth(f,n);
end
```

Load Neutral Images

```
M = 30; % 30 Neutral Images in JAFFE dataset
neutralEyesMouthTable = zeros(M,2*(n^2));
for i = 1:M
```

```
        f = imread(['C:\Users\cbinns\Documents\MATLAB\NEU\Image Processing\Final Project
Files\JAFFE\neutral\NE' num2str(i) '.tiff']);
        FaceDetect = vision.CascadeObjectDetector;
        face_bboxes = step(FaceDetect, f);
        onebox = face_bboxes(1,:);
        for i=1:size(face_bboxes,1)
        if face_bboxes(i,4) > onebox(1,4)
                onebox = face_bboxes(i,:);
        end
        end
        % Crop face from original image
        f = (imcrop(f,onebox));
    neutralEyesMouthTable(i,:) = extractDCT_Mouth(f,n);
end
```

Load Sadness Images

```
M = 30; % 30 Sadness Images in JAFFE dataset
sadEyesMouthTable = zeros(M,2*(n^2));
for i = 1:M
        f = imread(['C:\Users\cbinns\Documents\MATLAB\NEU\Image Processing\Final Project
Files\JAFFE\sadness\SA' num2str(i) '.tiff']);
        FaceDetect = vision.CascadeObjectDetector;
        face_bboxes = step(FaceDetect, f);
        onebox = face_bboxes(1,:);
        for i=1:size(face_bboxes,1)
        if face_bboxes(i,4) > onebox(1,4)
                onebox = face_bboxes(i,:);
        end
        end
        % Crop face from original image
        f = (imcrop(f,onebox));
    sadEyesMouthTable(i,:) = extractDCT_Mouth(f,n);
end
```

Load Surprise Images

```
M = 30; % 30 Surprise Images in JAFFE dataset
surpriseEyesMouthTable = zeros(M,2*(n^2));
for i = 1:M
        f = imread(['C:\Users\cbinns\Documents\MATLAB\NEU\Image Processing\Final Project
Files\JAFFE\surprise\SU' num2str(i) '.tiff']);
        FaceDetect = vision.CascadeObjectDetector;
        face_bboxes = step(FaceDetect, f);
        onebox = face_bboxes(1,:);
        for i=1:size(face_bboxes,1)
        if face_bboxes(i,4) > onebox(1,4)
                onebox = face_bboxes(i,:);
        end
        end
        % Crop face from original image
        f = (imcrop(f,onebox));
        surpriseEyesMouthTable(i,:) = extractDCT_Mouth(f,n);
end
```

Randomly Extract test image from each image set

```
% Extract random Anger Image and delete it from Test Images
r1 = randi(size(angerEyesMouthTable,1));
TestImageAnger = angerEyesMouthTable(r1,:);
angerEyesMouthTable(r1,:) = [];

% Extract random Disgust Image and delete it from Test Images
r2 = randi(size(disgustEyesMouthTable,1));
TestImageDisgust = disgustEyesMouthTable(r2,:);
disgustEyesMouthTable(r2,:) = [];

% Extract random Fear Image and delete it from Test Images
r3 = randi(size(fearEyesMouthTable,1));
TestImageFear = fearEyesMouthTable(r3,:);
```

```
fearEyesMouthTable(r3,:) = [];

% Extract random Happy Image and delete it from Test Images
r4 = randi(size(happyEyesMouthTable,1));
TestImageHappy = happyEyesMouthTable(r4,:);
happyEyesMouthTable(r4,:) = [];

% Extract random Neutral Image and delete it from Test Images
r5 = randi(size(neutralEyesMouthTable,1));
TestImageNeutral = neutralEyesMouthTable(r5,:);
neutralEyesMouthTable(r5,:) = [];

% Extract random Sad Image and delete it from Test Images
r6 = randi(size(sadEyesMouthTable,1));
TestImageSad = sadEyesMouthTable(r6,:);
sadEyesMouthTable(r6,:) = [];

% Extract random Surprise Image and delete it from Test Images
r7 = randi(size(surpriseEyesMouthTable,1));
TestImageSurprise = surpriseEyesMouthTable(r7,:);
surpriseEyesMouthTable(r7,:) = [];
```

All the Test Image Matrices are now 1 image less, which was extracted for the "Leave-One-Out" method for testing the trained classifier model

## Combine Extracted DCT Coefficients into FeatureTable with Emotion Labels

```
featureEyesMouthTable = vertcat(angerFaceTable, disgustFaceTable, fearFaceTable, happyFaceTable,
neutralFaceTable, sadFaceTable, surpriseFaceTable);
featureEyesMouthTable = array2table(featureEyesMouthTable);
for i = 1:n^2
        fname = ['E',num2str(i - 1)];
    featureEyesMouthTable.Properties.VariableNames(i) = {fname};
end
for i = n^2+1:2*n^2
        fname = ['M',num2str(i - 1)];
    featureEyesMouthTable.Properties.VariableNames(i) = {fname};
end
clear('fname');
% Add labels to the Emotion Class Cell Array
featureEyesMouthTable.EmotionClass = cell(size(featureEyesMouthTable,1),1);
featureEyesMouthTable.EmotionClass(1:size(angerEyesMouthTable,1)) = {'anger'};
featureEyesMouthTable.EmotionClass(size(angerEyesMouthTable,1)+1:size(angerEyesMouthTable,1)+size(disgust
EyesMouthTable,1)) = {'disgust'};
featureEyesMouthTable.EmotionClass(size(angerEyesMouthTable,1)+size(disgustEyesMouthTable,1)+1:size(anger
EyesMouthTable,1)+size(disgustEyesMouthTable,1)+size(fearEyesMouthTable,1)) = {'fear'};
featureEyesMouthTable.EmotionClass(size(angerEyesMouthTable,1)+size(disgustEyesMouthTable,1)+size(fearEye
sMouthTable,1)+1:size(angerEyesMouthTable,1)+size(disgustEyesMouthTable,1)+size(fearEyesMouthTable,1)+siz
e(happyEyesMouthTable,1)) = {'happy'};
featureEyesMouthTable.EmotionClass(size(angerEyesMouthTable,1)+size(disgustEyesMouthTable,1)+size(fearEye
sMouthTable,1)+size(happyEyesMouthTable,1)+1:size(angerEyesMouthTable,1)+size(disgustEyesMouthTable,1)+si
ze(fearEyesMouthTable,1)+size(happyEyesMouthTable,1)+size(neutralEyesMouthTable,1)) = {'neutral'};
featureEyesMouthTable.EmotionClass(size(angerEyesMouthTable,1)+size(disgustEyesMouthTable,1)+size(fearEye
sMouthTable,1)+size(happyEyesMouthTable,1)+size(neutralEyesMouthTable,1)+1:size(angerEyesMouthTable,1)+si
ze(disgustEyesMouthTable,1)+size(fearEyesMouthTable,1)+size(happyEyesMouthTable,1)+size(neutralEyesMouthT
able,1)+size(sadEyesMouthTable,1)) = {'sad'};
featureEyesMouthTable.EmotionClass(size(angerEyesMouthTable,1)+size(disgustEyesMouthTable,1)+size(fearEye
sMouthTable,1)+size(happyEyesMouthTable,1)+size(neutralEyesMouthTable,1)+size(sadEyesMouthTable,1)+1:size
(angerEyesMouthTable,1)+size(disgustEyesMouthTable,1)+size(fearEyesMouthTable,1)+size(happyEyesMouthTable
,1)+size(neutralEyesMouthTable,1)+size(sadEyesMouthTable,1)+size(surpriseEyesMouthTable,1)) =
{'surprise'};
% Convert Emotion Class to a Categorical type for Classifier Variable in
% Classification Learner App
featureEyesMouthTable.EmotionClass = categorical(featureEyesMouthTable.EmotionClass);
```

Output Trained Classification Model to the Workspace

```
testTable =
vertcat(TestImageAnger,TestImageDisgust,TestImageFear,TestImageHappy,TestImageNeutral,TestImageSad,TestIm
ageSurprise);
testTable = array2table(testTable);
for i = 1:n^2
       fname = ['E',num2str(i - 1)];
    testTable.Properties.VariableNames(i) = {fname};
end
for i = n^2+1:2*n^2
       fname = ['M',num2str(i - 1)];
    testTable.Properties.VariableNames(i) = {fname};
end
clear('fname');
% Add labels to the Emotion Class Cell Array
testTable.EmotionClass = cell(size(testTable,1),1);
testTable.EmotionClass = {'anger';'disgust';'fear';'happy';'neutral';'sad';'surprise'};
testTable.EmotionClass = categorical(testTable.EmotionClass);
testResults = trainedModel.predictFcn(testTable)
```

3. Annotating The Mouth and Eye Regions

```
%Detect Eyes and Mouth
EyeDetect = vision.CascadeObjectDetector('EyePairBig','MergeThreshold',7);
MouthDetect = vision.CascadeObjectDetector('Mouth','MergeThreshold',150);
eyes_bbox = step(EyeDetect,face);
mouth_bbox = step(EyeDetect,face);
% Annotate detected Eye Region
IFaces1 = insertObjectAnnotation(face, 'rectangle', eyes_bbox, 'Eyes');
IFaces2 = insertObjectAnnotation(face, 'rectangle', eyes_bbox, 'Mouth');
figure('Units','inches','Position',[0,0,12,4]);
subplot(1,2,1), imshow(IFaces1)
subplot(1,2,2), imshow(IFaces2)
```

Appendix B. MATLAB Functions

1. Face Region - DCT Coefficient Extraction

```
function [featureVector, face] = extractDCT_Face(f,n)
%EXTRACTDCT_FACE function is used to detect the face of the image, f, and
%   crop the image based on face detection. Once the face region is cropped, the 2D DCT
%   is performed on and then a n x n subset block of the low frequency information is
%   extracted and stored into a single feature Vector to be used for
%   Emotion Classification
%
if(length(size(f)) == 3) % If f is an RGB image, convert to grayscale and scale intensity values
    f = intensityScaling(rgb2gray4e(f));
else
    f = intensityScaling(f); % If f is a grayscale image, scale the intensity values
end
% Use Computer Vision Toolbox Functions to extract Face crop image
FaceDetect = vision.CascadeObjectDetector;
face_bboxes = step(FaceDetect, f);
onebox = face_bboxes(1,:);
for i=1:size(face_bboxes,1)
    if face_bboxes(i,4) > onebox(1,4)
        onebox = face_bboxes(i,:);
    end
```

```
end
% Crop face from original image
face = imcrop(f,onebox);
% Once face is cropped from original image, perform DCT-II operation to
% transform image into sequence of cosines, where low frequency information
% contains most of the vital information about the image
DCT_face = dct2(face);
% Generate plot of Image with DCT-II transform image
figure('Units','inches','Position',[0,0,12,4]);
subplot(1,2,1); imshow(face)
subplot(1,2,2); imshow(DCT_face)
% Extract an n x n block from the low frequencies of the DCT coefficients
faceTable = DCT_face(1:n,1:n);
% Use zigzag() to convert matrix of DCT coefficients into row vector based
% on sequential order of frequency components
featureVector = zigzag(faceTable);
end
```

## 2. Eye and Mouth Region – DCT Coefficient Extraction

```
function [featureVector, eyes, mouth] = extractDCT_EyesMouth(f,n)
% EXTRACTDCT function is used to detect the face of the image, f, and
%   crop the eye and mouth regions. Once these regions are cropped, the 2D DCT
%   is performed on each region and then a n x n block from the low
%   frequencies is extracted and stored into a single appended array for both
%   the mouth and eye regions.
%
if(length(size(f)) == 3) % If f is an RGB image, convert to grayscale and scale intensity values
    f = intensityScaling(rgb2gray4e(f));
else
    f = intensityScaling(f); % If f is a grayscale image, scale the intensity values
end
%[A,B] = size(f)
% Use Computer Vision Toolbox Functions to Detect Eyes and Mouth Regions
EyeDetect =  vision.CascadeObjectDetector('EyePairBig');
MouthDetect = vision.CascadeObjectDetector('Mouth','MergeThreshold',30);
% Detect Eyes and Mouth using step()
eyes_bboxes = step(EyeDetect,f)
%eyes_bboxes = eyes_bboxes(1,:);
mouth_bboxes = step(MouthDetect,f)


testEyes = strjoin(string(eyes_bboxes));
if (testEyes == "") || exist(testEyes)
    bounds = [48 98 198 49]
    eyes_bbox = bounds;
else
    eyes_bbox = eyes_bboxes(1,:);
    for i = 1:size(eyes_bboxes,1)
        if eyes_bboxes(i,2) < eyes_bbox(2)
            eyes_bbox = eyes_bboxes(i,:);
        end
    end
end


testMouth = strjoin(string(mouth_bboxes));
if (testMouth == "") || exist(testMouth)
    bounds = [eyes_bbox(1)+20 eyes_bbox(2)+95 eyes_bbox(3)-40 eyes_bbox(4)-20 ]
    mouth_bbox = bounds;
else
    mouth_bbox = mouth_bboxes(1,:);
    for i = 1:size(mouth_bboxes,1)
        if mouth_bboxes(i,2) > mouth_bbox(2)
            mouth_bbox = mouth_bboxes(i,:);
        end
    end
end


% Increase Height of Eye bbox to include eyebrows
```

```
eyes_bbox(4) = eyes_bbox(4) + 15;
eyes_bbox(2) = eyes_bbox(2) - 15;
IFaces = insertObjectAnnotation(f, 'rectangle', mouth_bboxes, 'Mouth');
%figure, imshow(IFaces);
% Crop regions based on dimension of bboxes for each region
eyes = imcrop(f,eyes_bbox);
mouth = imcrop(f,mouth_bbox);
% Generate figure of Eye/Mouth Images
figure('Units','inches','Position',[0,0,12,4]);
subplot(1,2,1); imshow(eyes)
subplot(1,2,2); imshow(mouth)
% Once regions are cropped from original image, perform DCT-II operation to
% transform image into sequence of cosines, where low frequency information
% contains most of the vital information about the image
DCT_eyes = dct2(eyes);
DCT_mouth = dct2(mouth);
% Extract an n x n block from the low frequencies of the DCT coefficients
mouthTable = DCT_mouth(1:n,1:n);
eyesTable = DCT_eyes(1:n,1:n);
% Use zigzag() to convert matrix of DCT coefficients into row vector based
% on sequential order of frequency components
mouthTable = zigzag(mouthTable);
eyesTable = zigzag(eyesTable);
% Form one row vector by appending one region to the end of the other
featureVector = horzcat(eyesTable, mouthTable);
end
```