- ***Steps to run the code***:
  - ***Code*** folder in the extracted folder contains the code
  - **DemoOriginal.m** file contains the demo code for the original algorithm
    - Change the values of **pathPos, pathNeg, pathTest** to point to the respective **Pos, Neg, Test** folders in the Code/Images/Genki or Smiles databases.
    - Run the file
    - The **label** variable contains the finally labelled classes
  - **OptimizedDemo.m** file contains the demo code for the optimized algorithm
    - Change the values of **pathPos, pathNeg, pathTest** to point to the respective **Pos, Neg, Test** folders in the Code/Images/Genki or Smiles databases.
    - Run the file
    - The **label** variable contains the finally labelled classes
- **Datasets used (provided with the code itself)**
  - Genki Database
  - Smiles Database
- ***Approximate time to run the algorithms***:

Algorithm	**Dataset**	**Data pre-processing and feature extraction (sec)**			**SVM training (sec)**	**SVM Prediction (sec)**
		Positive training	**Negative training**	**Testing data**		
Original	**Genki**	188.14	133.26	108.14	24.37	0.167
	Smiles	39.60	110.05	49.74	14.34	0.057
Optimized	**Genki**	209.57	145.63	117.82	29.64	0.009
	Smiles	43.93	120.03	52.66	15.87	0.010

- ***Changes in the original approach*****:** I have used my own approach for detection of skew, eyes, face (registration part). It was not very clearly mentioned in the research paper.
- ***Improvement***: I have referred 'SMILE DETECTION IN UNCONSTRAINED SCENARIOS USING SELF-SIMILARITY OF GRADIENTS FEATURES' to improve the feature extraction and classification accuracy. In the paper, a GSS (Gradient of Self-similarity) based approach was provided. Instead of getting HOG features for the whole face image at the same time, the paper proposes to divide the whole image into 16 cells and then calculate the final feature. After using the approach, I got an accuracy improvement on one dataset but there was a degradation on the second dataset. Might need to work more on the approach though.
- ***Documentation of system components***:
  - **DemoOriginal.m**: This file is the main demo file for the original method. No methods in the file.
  - **OptimizedDemo.m**: This file is the main demo file for the optimized method. No methods in the file.
  - **detectFeatures.m**: This is the main function that pre-processes the input file and then extracts HOG feature information for all the files in input folder.
    - **Inputs**: This function takes in 2 arguments:
      - pathToData: Directory path for the image
      - datatype: Type of input data. 1- Positive training, 2- Negative training, 3- Testing data
    - **Output**: This function has 2 output arguments
      - featureArr: HOG feature array for all files in input folder
      - classArr: Class label array for input files. 1- Positive, -1- Negative, 0- Testing data
    - **Operations performed:** Face detection, eye detection, angle identification, rotation, cropping, resizing, HOG feature extraction
    - **Measures used to evaluate the performance**:
      - Visually confirming the label values with the input Test folder images

- **detectOptimizedFeatures.m**: This is the main function that pre-processes the input file and then extracts HOG feature information using the optimization for all the files in input folder.
    - **Inputs**: This function takes in 2 arguments:
        - pathToData: Directory path for the image
        - datatype: Type of input data. 1- Positive training, 2- Negative training, 3- Testing data
    - **Output**: This function has 2 output arguments
        - featureArr: HOG feature array for all files in input folder
        - classArr: Class label array for input files. 1- Positive, -1- Negative, 0- Testing data
    - **Operations performed:** Face detection, eye detection, angle identification, rotation, cropping, resizing, GSS for HOG feature extraction
    - **Measures used to evaluate the performance**:
        - Visually confirming the label values with the input Test folder images

3rd party code:

- **buildDetector.m**: This file was provided by Matlab file exchange. For the eye component detection in face image.
- **detectFaceParts.m**: This file was provided by Matlab file exchange. For the eye component detection in face image.

- ***Comparison between the original and modified approach:***
  Also, the performance comparison can be seen on table on page 1.

Dataset	**Accuracy (%)**	
	Original Algorithm	**With Optimization**
Genki Dataset	**90.9313**	**45.8333**
Smiles Dataset	**50**	**61.4678**

- ***Experimental Results for original approach:***
    - **Steps to calculate the result in my code:** After running the DemoOriginal.m file, the label variable will have the classified labels for each of the images present in the Test folder. The class labels will then have to be manually verified by visually confirming with the files present in the Test folder. Images will be sorted in increasing order in the class labels array.
- ***Experimental Results for modified approach:*** The table above, has all the information regarding the accuracy and previous table on page1 has performance information by using the optimized approach. Accuracy improvement is visible when using the Smiles dataset but not with Genki dataset.
    - **Steps to calculate the result in my code:** After running the OptimizedDemo.m file, the label variable will have the classified labels for each of the images present in the Test folder. The class labels will then have to be manually verified by visually confirming with the files present in the Test folder. Images will be sorted in increasing order in the class labels array.