Real-time Gender Recognition based on selection of Eigen-features from Facial Images

A Practice School Report submitted to Manipal University in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering

Submitted by

Sahil Ajmera

130905324

Under the guidance of

N V Subba Reddy

Professor

MANIPAL INSTITUTE OF TECHNOLOGY

A Constituent Institute of Manipal University, Manipal

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

<May/June/July> 2017

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MANIPAL INSTITUTE OF TECHNOLOGY
A Constituent Institute of Manipal University, Manipal

Manipal
17th April, 2017

CERTIFICATE

This is to certify that the project titled **Real-time Gender Recognition based on selection of Eigen-features from Facial Images** is a record of the bonafide work done by **Sahil Ajmera** (*Reg. No.130905324*) submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology (B.Tech.) in **COMPUTER SCIENCE & ENGINEERING** of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Institute of Manipal University), during the academic year 2016-17.

N V Subba Reddy	**Ashalatha Nayak**
Professor	*HOD, CSE Dept.*
	M.I.T, MANIPAL

ACKNOWLEDGMENTS

I would like to extend my heartfelt thanks to the project guide N V Subba Reddy sir for his continued support throughout the duration of this project. I would also like to thank the esteemed head of the computer science department Ashalata Nayak mam for giving me the freedom to do research in the field I am interested in .Special thanks to the Laboratory incharge’s in the Network Technology Lab, Research Lab and the Operating Systems Lab for providing me with any help needed whether it was in software setup or making sure the equipment’s for my viva were present and working before-hand. Last but not the least none of this interesting research work would have been possible without the blessings of our director Gopalakrishna Prabhu sir. I thank him for the opportunities he has provided me through this research work.

ABSTRACT

Machine Learning has important applications in industries like facial recognition, surveillance systems, optical character recognition systems, recommendation engines etc. Gender recognition, a basic application of machine learning algorithms, is used to discriminate people. A good gender recognition algorithm can help in bigger systems like human machine interaction systems and surveillance systems.

The sample images from the FERET (Facial Recognition Technology) dataset was obtained from the National Institute of Standards and Technology. The images in .ppm format were first converted to grey-scale and the information was stored in an excel file. Important features of the grey scale image were obtained through cropping of the image by the Viola Jones algorithm and final zoom of 30x30 applied to the images to reduce the calculation load. The Eigen-projects were obtained from the images by the various steps of principal component analysis(mean,original-mean,covariance matrix,eigen-values,eigen-vectors,multiplication of best eigenvector and the entire dataset after subtraction).Binary data corresponding to each of the images was stored in an excel file. This data was multiplied by the Eigen-projects to obtain the Eigen-Features which were used to train and get the accuracy rate (AR) (Value to be used in fitness value calculation) from the support vector machine using WEKA tool. The binary data was acted upon by process of selection, crossover and mutation based on their fitness values to obtain optimal subset of features to be used in final training and testing of backpropagation neural network.

During experiments Eigen-features with a fitness value as low as 0.218 was obtained .432 prominent features were identified through a single iteration of the genetic algorithm. Accuracy rate of 0.78 was obtained on training and testing by the support vector machine. Low fitness values correspond to higher accuracy rate which shows the better performance of the selected Eigen-features. Comparison with the Grey-scale way of finding the gender shows that the Eigen-features could improve accuracy if trained with a larger set.

Using Eigen-features obtained from principal component analysis and genetic algorithm goes on to show that training and testing by a subset of Eigen-features could not only improve the speed as well as accuracy of gender recognition and this technique could indeed be viable in the development of real-time gender recognition systems. The research work has been implemented using the python programming language with packages imported from Anaconda .WEKA tool has been used to calculate the value of the variable to be used in the fitness function by training and testing the SVM.WEKA tool has also been used to train and test the neural network for indication of final accuracy for the grey-scale and the Eigen-feature categorization method.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 *Introduction*

This chapter introduces the general topic and the work area chosen for research. The present scenario with respect to the work has been discussed along with the motivation to pursue the research work. The objectives of the work, the target specifications (importance of the end result), the schedule according to which the work has been carried out, and the organization of the remaining project report has also been discussed.

Machine Learning has important applications in industries like facial recognition, surveillance systems, optical character recognition systems, recommendation engines etc. Gender recognition, a basic application of machine learning algorithms, is used to discriminate people. A good gender recognition algorithm can help in bigger systems like human machine interaction systems and surveillance systems.

Work by Tamura, S., Kawai, H., Mitsumoto, H. titled “Male/female identification from 8 x 6 very low resolution face images by neural network. Pattern Recognition, 331-335” in which a three-layer neural network is used to identify sex from a group of 8 X 8 low resolution face images, which include neither the head hair nor the outline of the faces. Hasnat, A., Haider, S., Bhattacharjee, D., and Nasipuri, M. proposed “a system for gender classification using lower part of face image” wherein A fast gender classification method was developed using features selected from mouth and chin only, which could reduce the number of features and make the identification system simpler.

The proposed method was tested on a small size dataset containing 75 male and 35 female face images with 94.34% accuracy. Work of Shukla, R., Shukla, R., et.al. Titled “Gender identification in human gait using neural network” wherein gender recognition can be achieved via the measurement of people’s gait. Li, B., Lian, X. C., and Lu, B. L. proposed gender classification by combining clothing, hair and facial component classifiers.

1.2 *Motivation*

Previous work has focused on small datasets. A scrutinized view of time taken during training and testing is also to be taken into account when large datasets are considered. Also the classifier should be adaptive for images collected in low light conditions, different facial expressions and even for skewed heads.

This work highlights a feature selection method which is based on facial images. The grayscale images are first converted to the Eigen-features using Principal component analysis and genetic algorithms. The features are then used for training of neural network and to assess the feasibility of the work. The Eigen-features show better flexibility and adaptability.

Genetic algorithm has been used to reduce the number of features for an image that need to be taken into account while training a neural network. This way the time to train a neural network reduces substantially as compared to the previous work with same or even better accuracy.

1.3 *Objectives*

The various objectives of the work are:

- To explore various application of machine learning
- To know about the domain of the selected application
- To implement the application based on the following
    - Viola Jones Algorithm
    - Principal Component Analysis
    - Genetic Algorithm
    - Back-Propagation Neural Network
- To suggest improvements over the existing work by doing research on better/faster algorithms

1.4 *Project schedule*

The project schedule is as follows:

- January 2017
    - Research on existing applications of the domain(Machine Learning)
    - Finding existing work done on each and every application
    - Finalizing the application to work on taking into consideration each and every aspect of implementation.
    - Submission of synopsis for the research work intended.
- February 2017
    - Starting the work on the selected application
    - Finishing the preprocessing of the dataset to be used for further work.
- March 2017
    - Starting work over principal component analysis for the dataset.
    - Finishing work over principal component analysis for the dataset.
    - Preparation of project report for the mid-term evaluation
    - Mid-term evaluation
- April 2017
    - Starting work over genetic algorithm on the Eigen-projects gathered from the principal component analysis

    - Using support vector machine for training and testing the Eigen-features obtained from the Eigen-projects to get value of parameter to be used in the fitness function calculation.
    - Finishing work over genetic algorithm to obtain the optimal subset of Eigen-features to be used for neural network training
    - Using back propagation neural network for training and testing the Eigen-Features and comparing the result with the neural network trained using Gray-scale images.
    - Documentation
- May 2017
    - Submission of report and evaluation

1.5 *Project Report Structure*

The rest of the project report is structured in the following way

- Chapter 2 talks about the literature review/background review related the project to understand the various terms used throughout the project and to better understand the motivation, working and conclusion of the project
- Chapter 3 talks about the methodology used in the completion of the project, the tools that have been used to speed up the completion of the project. This also includes the assumptions made through the completion of the project and the preliminary results and analysis.
- Chapter 4 talks about the various results and analysis done and the various graphs, tables etc. produced as a result. The deviations from the original results have been mentioned along with the valid reasons.
- Chapter 5 is the future work and conclusion section. This talks about the objective and methodology used in brief , along with the conclusions gathered from the project along with the significance of the results. This section also includes the future scope of work that is possible for the research work taken up.

CHAPTER 2
BACKGROUND THEORY / LITERATURE REVIEW

2.1 *Introduction*

This chapter deals with specific discussion on the research topic which briefly includes an introduction to the topic, the present state/recent developments on the topic, a brief background history related to the topic followed by a literature survey on the same. The outcomes as seen by different researchers in the area are also mentioned .The chapter ends with a thorough theoretical discussion on the topic along with conclusions.

Gender recognition in simple words is the ability to be able to discriminate among various genders. Gender recognition systems are part of bigger systems like human-machine interaction systems as well as surveillance systems. Viola Jones algorithm is an object detection framework that detects important features of the face like eyes, ears, nose etc. proposed in 2011 by Paul Viola and Michael Jones motivated primarily by the problem of face detection. It has become very popular as a fast and easy face detection algorithm. Matlab has already implemented object vision.CascadeObjectDetector that makes use of the Viola Jones algorithm and readily detects the important features in an image. Principal component analysis is a method to find a small number of uncorrelated variables called "principal components" from the data. The goal is to give explanation of maximum variance using a small number of principal components. The output of principal component analysis is usually Eigen vectors and Eigen values which when multiplied by the modified data set give us what is called as "feature vectors". Genetic algorithms work on the principal of evolution "Survival of the fittest". Solutions which have high fitness survive and move the generation forward through crossover and mutation to produce even better individuals while the ones with low fitness are wiped off in every iteration. The output of genetic algorithms is an optimal subset of solutions which are most important in further analysis.

2.2 *Literature Review*

A good gender recognition algorithm can help in building bigger systems like surveillance systems and human-machine interaction systems. Over the years many successful gender recognition algorithms have been found and reviewed .Work has been perennial in this field of face and gender recognition technology. Biometric systems are actually getting implemented and used in many parts of the world now.

Gender recognition in simple words is the ability to be able to discriminate among various genders. Gender recognition systems are part of bigger systems like human-machine interaction systems as well as surveillance systems. Viola Jones algorithm is an object detection framework that detects important features of the face like eyes, ears, nose etc. proposed in 2011 by Paul Viola and Michael Jones motivated primarily by the problem of face detection. It has become very popular as a fast and easy face detection algorithm. Matlab has already implemented object vision.CascadeObjectDetector that makes use of the Viola Jones algorithm and readily detects the important features in an image. Principal component analysis is a method to find a small number of uncorrelated variables called "principal components" from the data. The goal is to give explanation of maximum variance using a small number of principal components. The output of principal component analysis is usually Eigen vectors and Eigen values which when multiplied by the modified data set give us what is called as "feature vectors". Genetic algorithms work on the principal of evolution "Survival of the fittest". Solutions which have high fitness survive and move the generation forward through crossover and mutation to produce even better individuals while the ones with low fitness are wiped off in every iteration. The output of genetic algorithms is an optimal subset of solutions which are most important in further analysis.

Shukla,R.,Shukla R.,et.al in their work "Gender identification in human gait using neural network" and Shan,C.,Gong,S. and McOwan in their work "Fusing gait and face cues for human gender recognition"used the features of the person's gait for identification of gender.Li,B.,Lian,X. C.,and Lu,B.L. in their work "Gender classification by combining clothing , hair and facial component classifiers" gender recognition was specifically focused on clothing, posture or voice.Nazir,M.,Anwar.M.Mirza in their work "Multi-view gender classification using hybrid transformed features"where two methods were suggested for front face gender recognition.The geometry based approach took estimates of distance of feature points for instance, the width and height of the face,eye distance etc. which are used to form a feature dataset for further analysis.Jaswanthe,A.,Khan,Gour,B.,A.U in their work "Gender classification technique based on facial features using neural network" use the Viola jones algorithm to crop eyes, nose and mouth from facial images and then calculate the measured distances from eyebrow to eye,eyebrow to nosetop,nosetop to mouth and then input the following to a neural network as feature vectors to be trained.TAMURA,S.,KAWAI,H.,MITSUMOTO,H. in their work "Male/female identification from 8x6 very low resolution images by neural network". In this work a three-layer neural network is used to identify sex from a group of 8x8 low resolution face images which include neither the head hair nor the outline of the faces.Moghaddam,B.,Yang,M.H. in their work "Gender classification with support vector machines" presented a comprehensive evaluation of various classification methods for determination of gender and found out that a Gaussian kernel Support Vector Machine can achieve the least error rate. Khan , S.A.,Ahmad,M.,et.al in their work "A comparative analysis of gender classification techniques" and Ng,B.,Tay,Y, and Goi,B. in their work "Vision-based human gender recognition" have focused on features that represent gender and has the least correlation to other facial characteristics,such as color,race,expression,age,wearable.In the work of Rai,P.,Khanna,P. titled "Appearance based gender classification with pca and pc on approximate face image” it was concluded that a combination of dual-directional dimension of Principal Component Analysis and SVM could obtain a high performance.Sun,Z.Bebis,G.et.al. in their work “Gender feature subset selection for gender classification”wherein a feature subset was extracted from the PCA Eigen-vectors with genetic algorithms and was applied to four classifier models for comparison.In the work of Tapia,J.E. and Perez,C.A titled “Gender classification based on fusion of different spatial scale features selected by mutual information from histogram of lbp,intensity and shape” found out that when a mixed features were taken for analysis,including intensity,shape and texture a best classification rate of 99.13% was obtained on the FERET database which included 199 female and 212 male images,but then a larger number of features were used in the experiments.In the work of Hasnat,A.,Haider,S.,Bhattacharjee,D. and Nasipuri,M. titled “A proposed system for gender classification using lower part of face image” A classifier which used features from mouth and chin only was developed.When tested on a small dataset containing 75 male and 35 female images it gave 94.34% accuracy.

CHAPTER 3
METHODOLOGY

This chapter gives information about the detailed methodology adopted in completion of the research work.Hence, it also includes the assumptions made, any design and modelling eg.block diagrams, module specifications, tools used and preliminary result analysis.

3.1 *Detailed Methodology*

The methodology adopted for this research work is described below:

- The first level of cropping is done with the gray scale image using Viola Jones algorithm
- The second level of cropping is done with the image by zooming the image to 30x30.
- The 30x30 pixel matrix is converted to 1x900 pixel matrix for finding initial feature vectors.
- The calculation of the mean image is done by taking mean of the 900 pixel intensities of all images taken into consideration.
- The mean image is subtracted from each of the images and the 1x900 matrix is reconverted to 30x30 for further steps of the PCA(Principal Component Analysis)
- From the 30x30 matrix the covariance matrix and the corresponding Eigen-values corresponding to that image is calculated.
- From the Eigen values obtained from each of the images, the maximum Eigen value is found out and each of the 30x30 matrix is multiplied with the Eigen-vectors corresponding to the maximum Eigen-value.
- These form the feature vectors obtained from the principal component analysis.
- A binary gene sequence (chromosome) corresponding to every feature in an image is carved out.This binary sequence is multiplied with the Eigen-projects (feature vectors) obtained in the previous step to get the Eigen-features for the first step.2/3 of the Eigen-features are used for training and 1/3 are used for testing our Support Vector Machine (SVM) implemented by WEKA tool.The value of accuracy rate (AR) is taken down from the results of the SVM.
- The following fitting function is implemented for each and every gene sequence

  $F=a\ (1-AR)\ +b\ (FG/SG)$

  AR=Accuracy rate
  a (Coefficient) =0.9
  b (Coefficient)=0.1
  FG=selected feature genes (Genes in a chromosome that are 1)
  SG=900

The equation shows that higher accuracy rate can be achieved with less feature genes hence the fitness value will be lower which identifies better performance of the Eigen-features

- A particular value of fitness is chosen randomly and chromosomes having better value are chosen from this iteration and the chromosomes with less fitness than this value are wiped out in this iteration.
- Among the chromosomes selected , a particular value is chosen and the chromosomes having more fitness than this value are chosen for crossover and those having less fitness than this value are passed on to the next generation as it is.(Rank Selection Algorithm)
- Crossover operation taken into account is single point crossover. Crossover happens at a single point wherein the first half comes from the first parent and the second half comes from the second parent. Hence, a new chromosome is formed which is placed in place of one of the wiped out chromosome.
- The new offspring's produced undergo mutation (bit-change) at random point as an improvement over the offspring.
- Iterations of the genetic algorithm follow the exact same procedure from wiping out chromosome to selection to crossover to mutation.
- After a particular condition is met the genetic algorithm is stopped and we get a better solution for our further analysis unlike before the genetic algorithm.
- Apply the final Eigen-features obtained to the back propagation neural network for training the neural network.
- Apply gray-scale values of the images as input to the neural network
- Compare the results obtained in the above two steps.
- Recording the results obtained in tables etc.
- Do research on optimizing the current setup for better results if time permits.

3.2 *Assumptions made*

- Crossover probability is 0.8 so it has been assumed that the parents selected for crossover will in fact undergo crossover if another parent is present and will not undergo crossover if another parent is not present
- Mutation probability if 0.1 so it has been assumed that the first 1' that is seen in the bit sequence of the offspring chromosome will be converted to a 0.
- Assumed value of fitness function while wiping out chromosomes or for selecting parents
- Only the Eigen-vector corresponding to the maximum Eigen-value is taken for analysis as it is enough indicator of the maximum variance in the data.

3.3 *Block Diagrams*

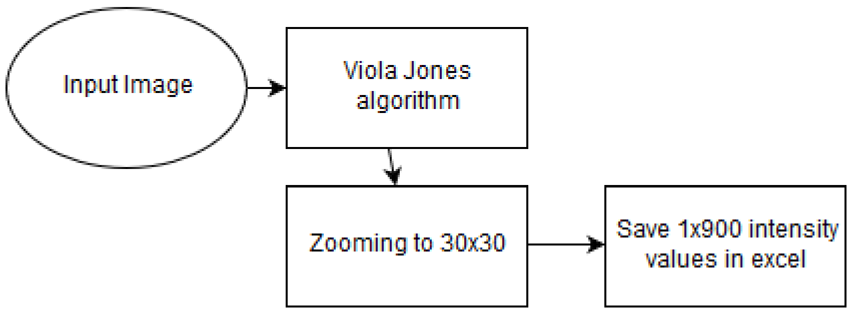
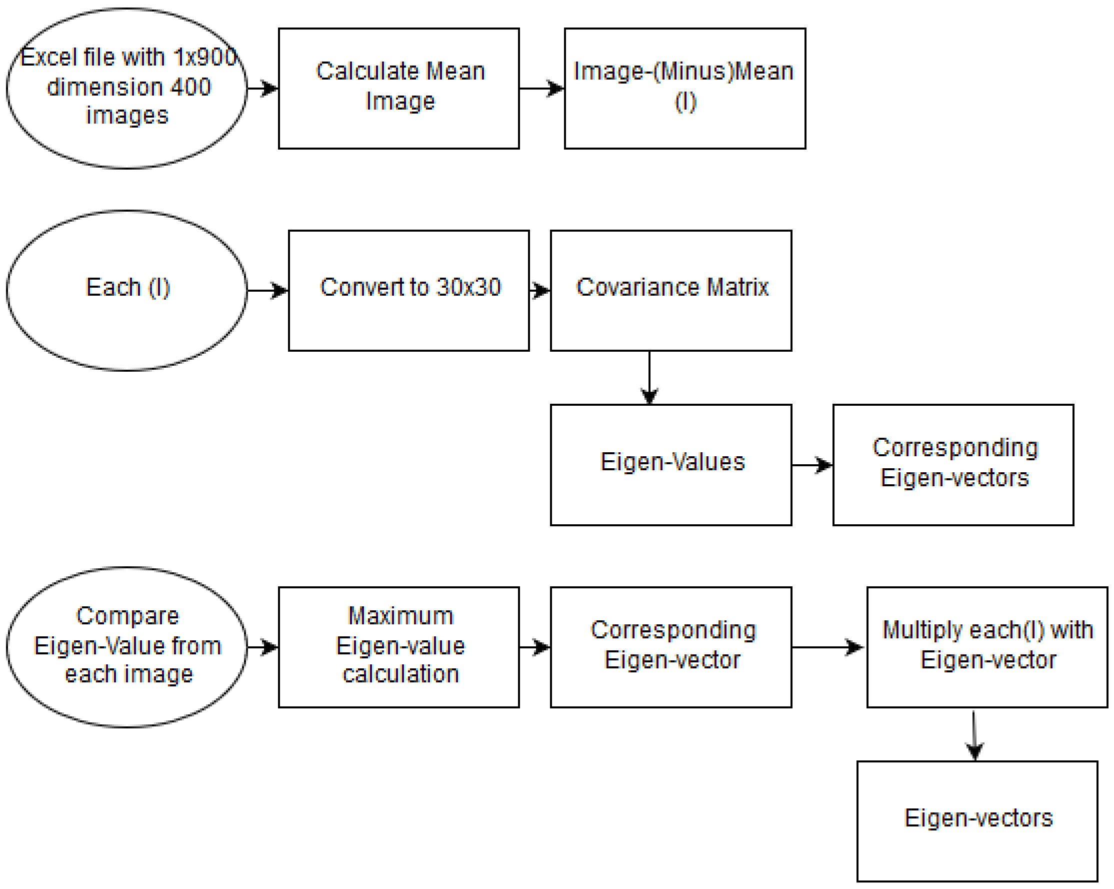


Figure 1 Pre-processing



Excel file with 1x900 dimension 400 images
Calculate Mean Image
Image-(Minus)Mean (I)
Each (I)
Convert to 30x30
Covariance Matrix
Eigen-Values
Corresponding Eigen-vectors
Compare Eigen-Value from each image
Maximum Eigen-value calculation
Corresponding Eigen-vector
Multiply each(I) with Eigen-vector
Eigen-vectors

Figure 2Principal Component Analysis

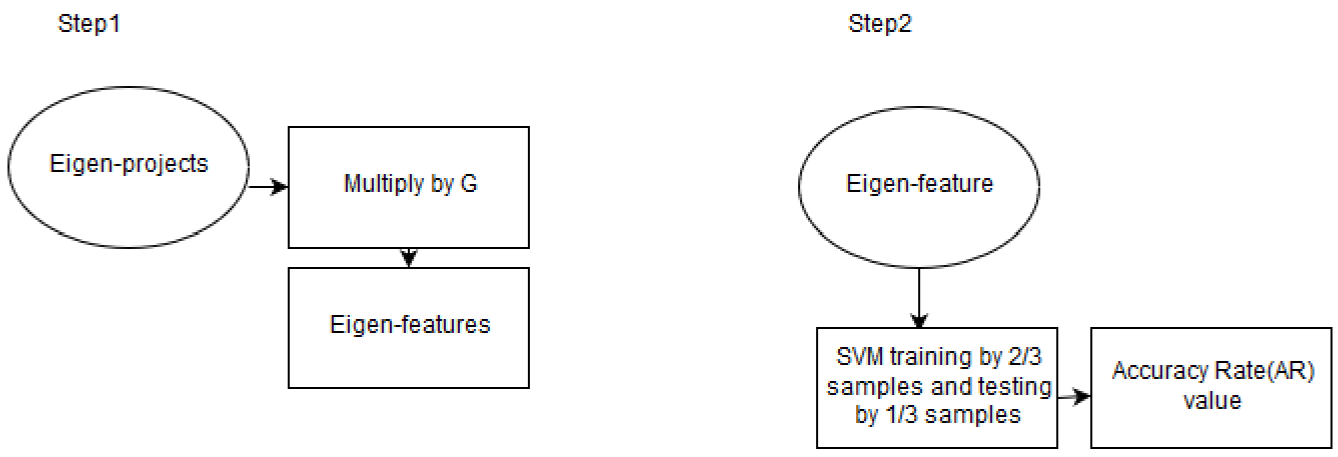


Figure 3Step1-2GeneticAlgorithm

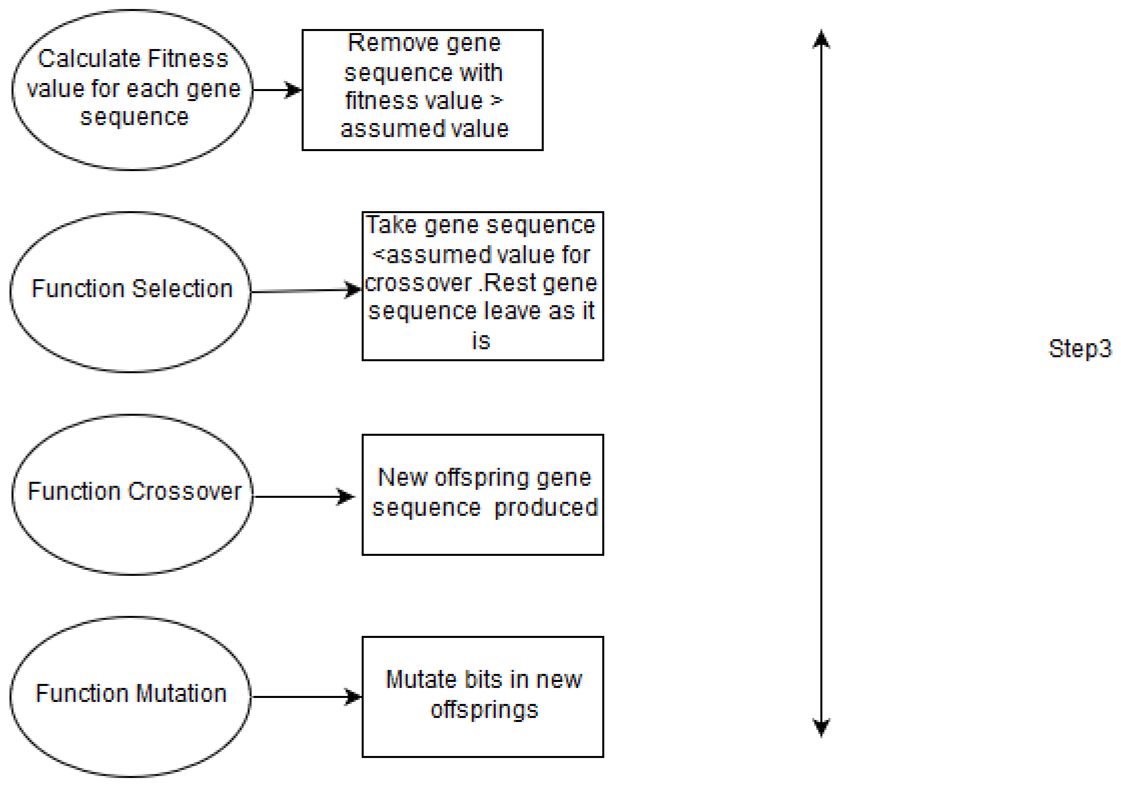

Calculate Fitness value for each gene sequence
Remove gene sequence with fitness value > assumed value
Function Selection
Take gene sequence <assumed value for crossover .Rest gene sequence leave as it is
Function Crossover
New offspring gene sequence produced
Function Mutation
Mutate bits in new offsprings
Step3

Figure 4Step3GeneticAlgorithm

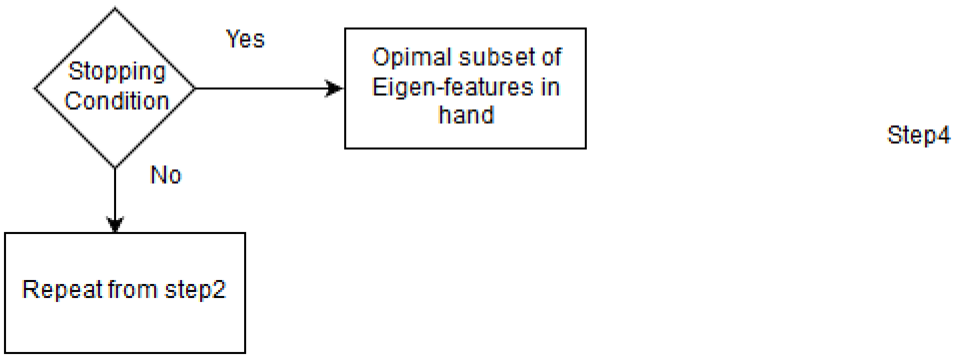


Figure 5Step4GeneticAlgorithm

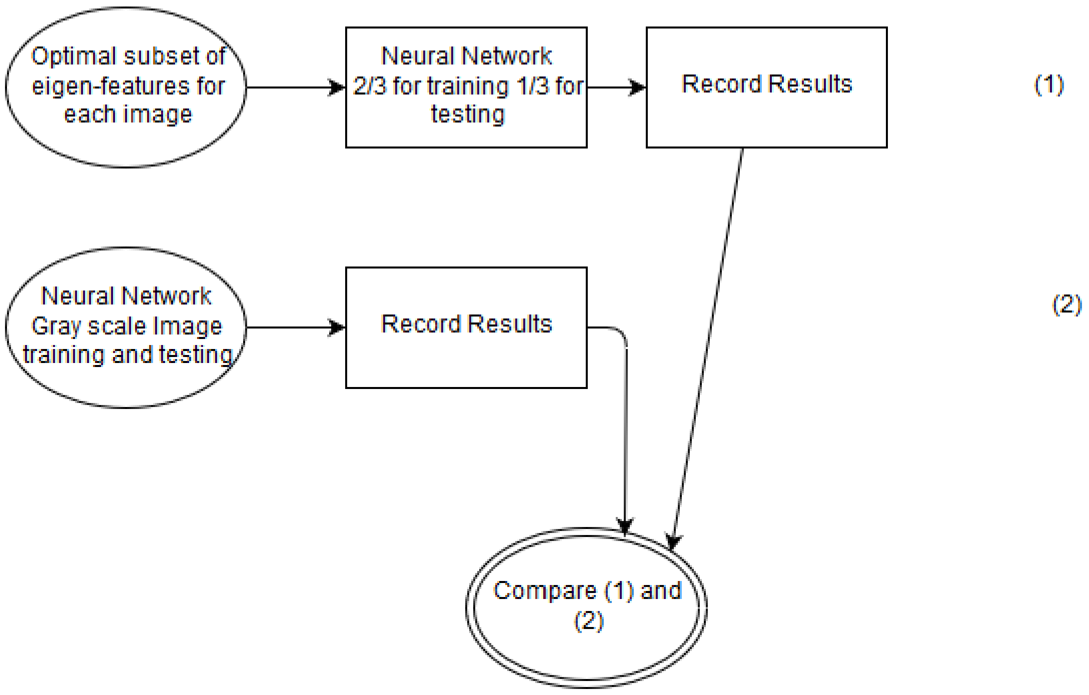
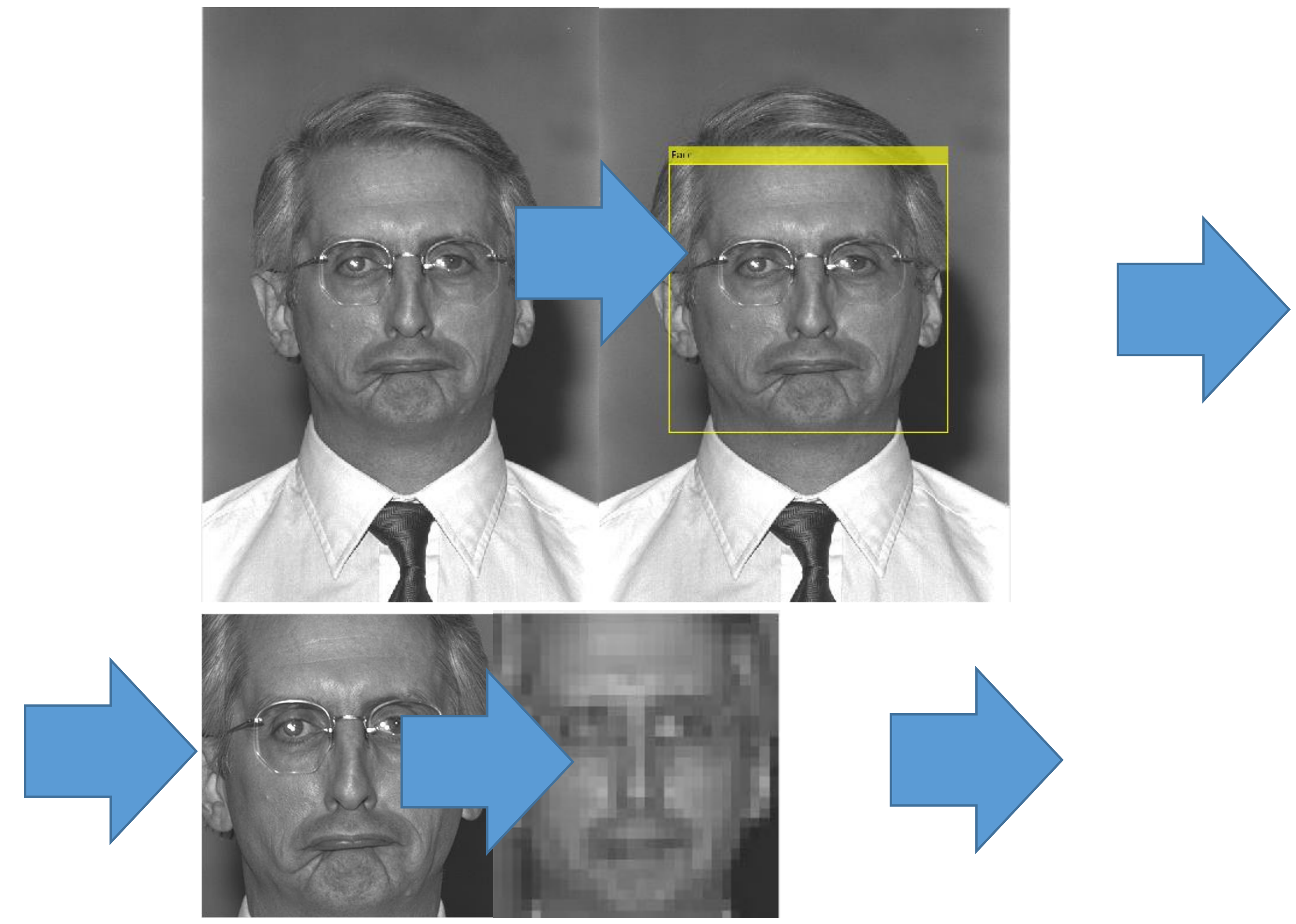

Optimal subset of eigen-features for each image
Neural Network 2/3 for training 1/3 for testing
Record Results
(1)
Neural Network Gray scale Image training and testing
Record Results
(2)
Compare (1) and (2)

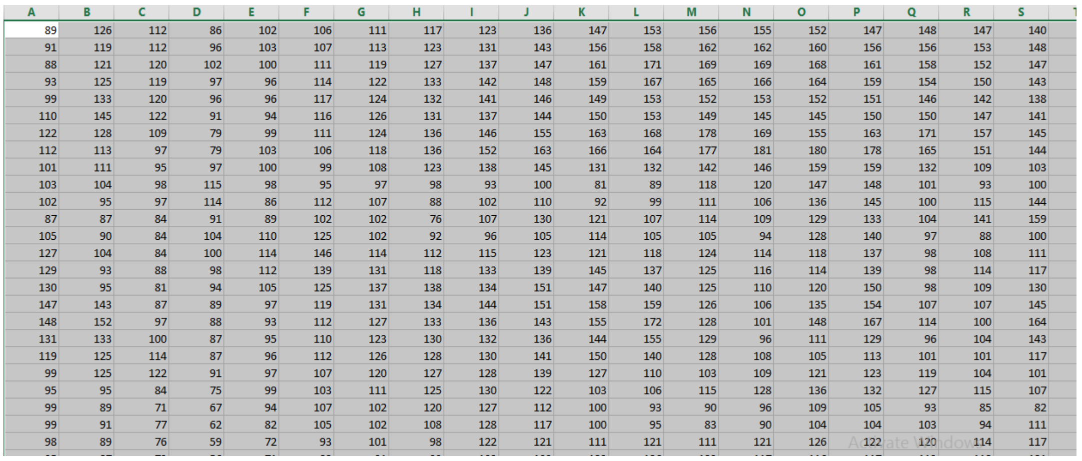
Figure 6Back-propagation neural network

3.4 *Tools Used*

- Programming language-Python
- Python IDE-Wing
- Anaconda
- WEKA tool
- Matlab

3.5 *Preliminary Result*





A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
89	126	112	86	102	106	111	117	123	136	147	153	156	155	152	147	148	147	140
91	119	112	96	103	107	113	123	131	143	156	158	162	162	160	156	156	153	148
88	121	120	102	100	111	119	127	137	147	161	171	169	169	168	161	158	152	147
93	125	119	97	96	114	122	133	142	148	159	167	165	166	164	159	154	150	143
99	133	120	96	96	117	124	132	141	146	149	153	152	153	152	151	146	142	138
110	145	122	91	94	116	126	131	137	144	150	153	149	145	145	150	150	147	141
122	128	109	79	99	111	124	136	146	155	163	168	178	169	155	163	171	157	145
112	113	97	79	103	106	118	136	152	163	166	164	177	181	180	178	165	151	144
101	111	95	97	100	99	108	123	138	145	131	132	142	146	159	159	132	109	103
103	104	98	115	98	95	97	98	93	100	81	89	118	120	147	148	101	93	100
102	95	97	114	86	112	107	88	102	110	92	99	111	106	136	145	100	115	144
87	87	84	91	89	102	102	76	107	130	121	107	114	109	129	133	104	141	159
105	90	84	104	110	125	102	92	96	105	114	105	105	94	128	140	97	88	100
127	104	84	100	114	146	114	112	115	123	121	118	124	114	118	137	98	108	111
129	93	88	98	112	139	131	118	133	139	145	137	125	116	114	139	98	114	117
130	95	81	94	105	125	137	138	134	151	147	140	125	110	120	150	98	109	130
147	143	87	89	97	119	131	134	144	151	158	159	126	106	135	154	107	107	145
148	152	97	88	93	112	127	133	136	143	155	172	128	101	148	167	114	100	164
131	133	100	87	95	110	123	130	132	136	144	155	129	96	111	129	96	104	143
119	125	114	87	96	112	126	128	130	141	150	140	128	108	105	113	101	101	117
99	125	122	91	97	107	120	127	128	139	127	110	103	109	121	123	119	104	101
95	95	84	75	99	103	111	125	130	122	103	106	115	128	136	132	127	115	107
99	89	71	67	94	107	102	120	127	112	100	93	90	96	109	105	93	85	82
99	91	77	62	82	105	102	108	128	117	100	95	83	90	104	104	103	94	111
98	89	76	59	72	93	101	98	122	121	111	121	111	121	126	122	120	114	117

Figure 7Gray scale value of a single 30x30 image

The database taken from analysis is FERET(Facial Recognition Technology).Out of 1400 images present in the database 400 have been taken for analysis,350 for training and 50 for testing the system.The images have various facial conditions, i.e., some wearing glasses. The resolution of each image is 116 x 80 pixels, with 256 gray levels per pixel

CHAPTER 4
RESULT ANALYSIS

4.1 *Introduction*

This chapter includes the results that were seen throughout the research. Any results whether in graphical/tabular form is mentioned and the explanation is also attached for the seen outcomes. Appropriate reasons have been given for any deviation in the results obtained .This chapter ends with conclusions drawn from the research work.

- *Result analysis*

Table 1FinalAccuracyResults

Method	Samples	Features	Recognition Rate		
			Correct	Incorrect	Accuracy
Grey	50	900	47	2	91.48%
Eigen	50	432	37	12	74.4%

- *Classification Result of the two categorization methods on the FERET dataset*

The above table shows that even on less trained neural network the Eigen features can show accuracy of about 74% on the FERET dataset (which only increases as the neural network is trained better).

- *Reason for Deviation*

The main reason for deviation from the expected result is that the neural network could only be trained by 350 images in the time span of this research work. Had it been trained with over 1000 images the results would certainly be in the favour of Eigen-feature to be used for classification of genders as is evident from the reference paper "Real-time Gender Recognition based on Eigen-features selection from facial images".

4.2 Conclusion

The studies done above on one of the many dataset shows that Eigen-features produced through pca and genetic algorithm not only increase the speed of the classifier but also the accuracy of the classifier. The same concept can also be applied in the generation of real-time gender recognition systems as well.

CHAPTER 5
CONCLUSION AND FUTURE SCOPE

5.1 *Problem Statement*

A good gender recognition algorithm can help in designing bigger systems like human-machine interaction systems and surveillance systems. Work done in the past in this field gives light to some issues like:

- Work done in an experimental database cannot fully represent the quality of the used method for gender recognition
- It is very important to determine the feasibility of the mechanism on a large database. For eg.Time taken during training and testing should be acceptable.
- Also the adaptability of any mechanism should be taken into account. For eg.Accuracy of the mechanism on images collected in different facial expressions, natural light conditions, even for skewed heads.

5.2 *Work Methodology*

- Obtained images are converted to grey-scale ,cropped using Viola Jones algorithm and zoomed to 30x30 in matlab software
- The 30x30(or1x900) is stored for every image. These values are used to calculate the mean face image and the difference values of an image and the mean face image.
- The difference values obtained are used to calculate the covariance matrix for an image and, in turn, the Eigen-values and Eigen-vectors.
- The maximum Eigen-value is sorted from all images and the Eigen-vector is multiplied with all the difference values (1x900) to obtain the feature-vector.
- Binary data is created for each feature for purpose of genetic algorithm. Using the fitness function ,

  $F=a(1-AR)+b(FG/SG)$

  AR=Accuracy rate
  a (Coefficient)=0.9
  b (Coefficient)=0.1
  FG=selected feature genes (Genes in a chromosome that are 1)
  SG=900

  Appropriate fitness values are given to each feature according to the fitness function.

- Based on an assumed fitness value, some features that are of less significance are wiped out and out of the remaining features those which have the highest fitness are chosen for purpose of crossover.
- Offspring's produced from crossover are placed in place of wiped out individuals and these then go for mutation.
- Iteration of the genetic algorithm repeats for every new generation until the stopping condition is met.
- The optimal subset of features is obtained in the end, which is used for training and testing the images in a back-propagation neural network.

5.3 *Conclusions*

The novel approach of using Eigen-features using Principal Component Analysis and Genetic Algorithm can show better performance over the Grey-scale in terms of speed and accuracy. Also the method seems much more adaptive than Grey-scale for images in natural light conditions, grey-scale etc.

The results show an accuracy of 74% as compared to the 90% accuracy of Grey-scale. The deviation from the above fact is a result of less trained neural network which can definitely be improved. The features taken into account for training the neural network have almost been reduced to a half with the first iteration of the genetic algorithm.

5.4 *Future scope of work*

The neural network once trained with over a 1000 images as compared to the 400 images taken for this research work can be tested with images from another dataset such as FEI and also for images from the web using the crawler tool. Results obtained from such datasets will be assurance of the fact that this work can in fact be used for the development of real-time gender recognition systems as well.

The genetic algorithm can be replaced with Particle Swarm Optimization (PSO) for faster and maybe even better results over the binary genetic data. Particle swarm optimization (PSO) is a optimization technique developed by Dr. Eberhard and Dr. Kennedy, inspired by social behaviour of bird flocking or fish schooling.

The neural network can be adjusted according to weights taken through a genetic algorithm to improve the accuracy rate observed .The neural network can be replaced with a better classifier so as to improve the accuracy rate.

REFERENCES

[1] Shukla, R., Shukla, R., et.al., 2012. Gender identification in human gait using neural network. International Journal of Modern Education and Computer Science, 70-75.

[2] Shan, C., Gong, S., and McOwan, P. W., 2008. Fusing gait and face cues for human gender recognition. Neurocomputing, 71(10), 1931-1938.

[3] Li, B., Lian, X. C., and Lu, B. L., 2012. Gender classification by combining clothing, hair and facial component classifiers. Neurocomputing,76(1), 18-27.

[4] Nazir, M., Anwar.M.Mirza, 2012. Multi-view gender classification using hybrid transformed features. International Journal of Multimedia and Ubiquitous Engineering.

[5] Jaswante, A., Khan, A.U., Gour, B., 2013. Gender classification technique based on facial features using neural network. International Journal of Computer Science and Information Technologies, 839-843.

[6] TAMURA, S., KAWAI, H., MITSUMOTO, H., 1996. Male/female identification from 8 x 6 very low resolution face images by neural network. Pattern Recognition, 331-335.

[7] Moghaddam, B., Yang, M.H., 2000. Gender classification with support vector machines, Proceedings of the 4th IEEE International Conference on Face and Gesture Recognition, pp. 306-311.

[8] Khan, S.A., Ahmad, M., et.al., 2013. A comparative analysis of gender classification techniques. International Journal of Bio-Science and Bio-Technology, 223-244.

[9] Ng,B., Tay,Y, and Goi,B., 2012. Vision-based human gender recognition: A survey. arXiv preprint arXiv:1204.1611.

[10] Rai, P., Khanna, P., 2014. Appearance based gender classification with pca and pc on approximation face image. Industrial and Information Systems.

[11] Sun, Z., Bebis,G., et.al., 2002. Genetic feature subset selection for gender classification: A comparison study. IEEE Intl. Conf. on Image Processing.

[12] Tapia, J.E. and Perez, C.A., 2013. Gender classification based on fusion of different spatial scale features selected by mutual information from histogram of lbp, intensity, and shape. IEEE Transactions on Information Forensics and Security.

[13] Hasnat, A., Haider, S., Bhattacharjee, D., and Nasipuri, M., 2015. A proposed system for gender classification using lower part of face image. International Conference on Information Processing, pp. 581-585.

[14] FERET, 2015. Feret database.[online].available. URL: http://www.nist.gov/itl/iad/ig/colorferet.cfm.

[15] FEI, 2012. Fei database.[online].available. URL: http://fei.edu.br/cet/facedatabase.html.

[16] Image-Downloader, 2015. Web image downloader tools.[online].available. URL: https://github.com/kencoken/imsearch-tools.

[17] Viola, P. and Jones, M., 2001. 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, pp.

IC511.

[18] Face-Detector, 2015. vision.cascade object detector face detect.URL:http://www.mathworks.com/help/vision/ref/vision. Cascade objectdetector-class.html.

[19] Makinen, E., Raisamo, R., 2008. An experimental comparison of gender classification methods. Pattern Recognition Letters, 1544-1556.

[20] Turk, M. and Pentland, A., 1991. Eigen faces for recognition. Journal of cognitive neuroscience.

[21] Tarwani, K.M., and Bhoyar, K.K., 2014. Approaches to gender classification using facial images. International Journal of Image Processing and Visual Communication.

[22]eig,2015.Eigen Value And Eigen Vectors URL: http://www.mathworks.com/help/matlab/ref/eig.html.

[23] Satone, M., and Kharate, G.K. 2013. Selection of eigenvectors for face recognition. International Journal of Advanced Computer Science and Applications, 95-98.

[24] Oluleye, B., Leisa, A., et.al., 2014. A genetic algorithm-based feature selection. International Journal of Electronics Communication and Computer Engineering, 889-905.

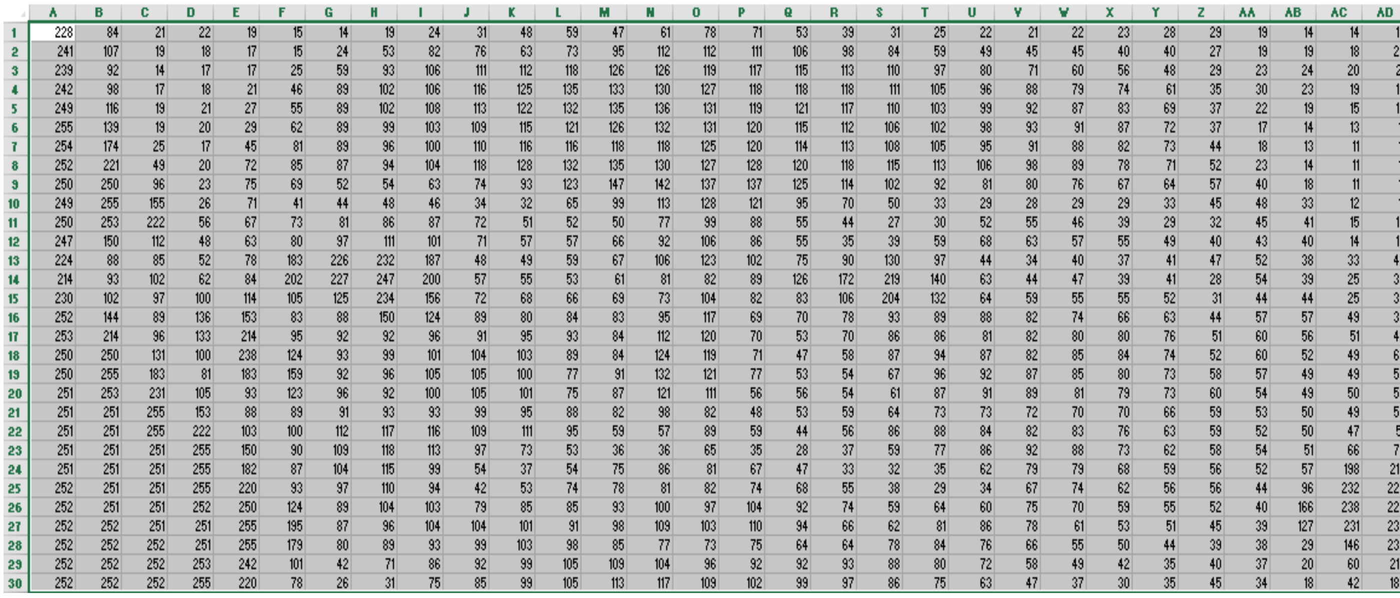
[25] Shakhnarovich, G., Viola, P. A., and Moghaddam, B., 2002. A unified learning framework for real time face detection and classification. Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition, 14-21.

ANNEXURES (optional)

Product Data Sheet

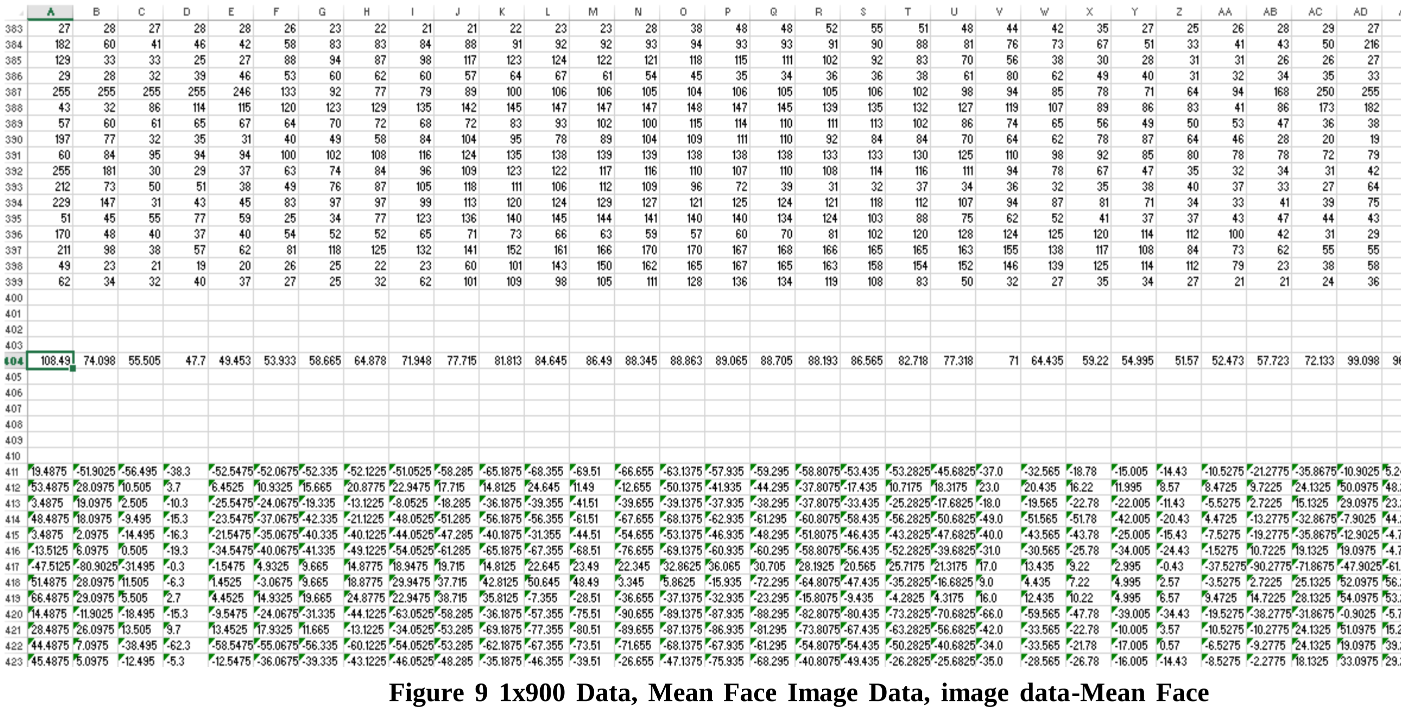
Table 2FinalAccuracyResults

Method	Samples	Features	Recognition Rate		
			Correct	Incorrect	Accuracy
Grey	50	900	47	2	91.48%
Eigen	50	432	37	12	74.4%



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
1	228	84	21	22	19	15	14	19	24	31	48	59	47	61	78	71	53	39	31	25	22	21	22	23	28	29	19	14	14
2	241	107	19	18	17	15	24	53	82	76	63	73	95	112	112	111	106	98	84	59	49	45	45	40	40	27	19	19	18
3	239	92	14	17	17	25	59	93	106	111	112	118	126	126	119	117	115	113	110	97	80	71	60	56	48	29	23	24	20
4	242	98	17	18	21	46	89	102	106	116	125	135	133	130	127	118	118	118	111	105	96	88	79	74	61	35	30	23	19
5	249	116	19	21	27	55	89	102	108	113	122	132	135	136	131	119	121	117	110	103	99	92	87	83	69	37	22	19	15
6	255	139	19	20	29	62	89	99	103	109	115	121	126	132	131	120	115	112	106	102	98	93	91	87	72	37	17	14	13
7	254	174	25	17	45	81	89	96	100	110	116	116	118	118	125	120	114	113	108	105	95	91	88	82	73	44	18	13	11
8	252	221	49	20	72	85	87	94	104	118	128	132	135	130	127	128	120	118	115	113	106	98	89	78	71	52	23	14	11
9	250	250	96	23	75	69	52	54	63	74	93	123	147	142	137	137	125	114	102	92	81	80	76	67	64	57	40	18	11
10	249	255	155	26	71	41	44	48	46	34	32	65	99	113	128	121	95	70	50	33	29	28	29	29	33	45	48	33	12
11	250	253	222	56	67	73	81	86	87	72	51	52	50	77	99	88	55	44	27	30	52	55	46	39	29	32	45	41	15
12	247	150	112	48	63	80	97	111	101	71	57	57	66	92	106	86	55	35	39	59	68	63	57	55	49	40	43	40	14
13	224	88	85	52	78	183	226	232	187	48	49	59	67	106	123	102	75	90	130	97	44	34	40	37	41	47	52	38	33
14	214	93	102	62	84	202	227	247	200	57	55	53	61	81	82	89	126	172	219	140	63	44	47	39	41	28	54	39	25
15	230	102	97	100	114	105	125	234	156	72	68	66	69	73	104	82	83	106	204	132	64	59	55	55	52	31	44	44	25
16	252	144	89	136	153	83	88	150	124	89	80	84	83	95	117	69	70	78	93	89	88	82	74	66	63	44	57	57	49
17	253	214	96	133	214	95	92	92	96	91	95	93	84	112	120	70	53	70	86	86	81	82	80	80	76	51	60	56	51
18	250	250	131	100	238	124	93	99	101	104	103	89	84	124	119	71	47	58	87	94	87	82	85	84	74	52	60	52	49
19	250	255	183	81	183	159	92	96	105	105	100	77	91	132	121	77	53	54	67	96	92	87	85	80	73	58	57	49	49
20	251	253	231	105	93	123	96	92	100	105	101	75	87	121	111	56	56	54	61	87	91	89	81	79	73	60	54	49	50
21	251	251	255	153	88	89	91	93	93	99	95	88	82	98	82	48	53	59	64	73	73	72	70	70	66	59	53	50	49
22	251	251	255	222	103	100	112	117	116	109	111	95	59	57	89	59	44	56	86	88	84	82	83	76	63	59	52	50	47
23	251	251	251	255	150	90	109	118	113	97	73	53	36	36	65	35	28	37	59	77	86	92	88	73	62	58	54	51	66
24	251	251	251	255	182	87	104	115	99	54	37	54	75	86	81	67	47	33	32	35	62	79	79	68	59	56	52	57	198
25	252	251	251	255	220	93	97	110	94	42	53	74	78	81	82	74	68	55	38	29	34	67	74	62	56	56	44	96	232
26	252	251	251	252	250	124	89	104	103	79	85	85	93	100	97	104	92	74	59	64	60	75	70	59	55	52	40	166	238
27	252	252	251	251	255	195	87	96	104	104	101	91	98	109	103	110	94	66	62	81	86	78	61	53	51	45	39	127	231
28	252	252	252	251	255	179	80	89	93	99	103	98	85	77	73	75	64	64	78	84	76	66	55	50	44	39	38	29	146
29	252	252	252	253	242	101	42	71	86	92	99	105	109	104	96	92	92	93	88	80	72	58	49	42	35	40	37	20	60
30	252	252	252	255	220	78	26	31	75	85	99	105	113	117	109	102	99	97	86	75	63	47	37	30	35	45	34	18	42

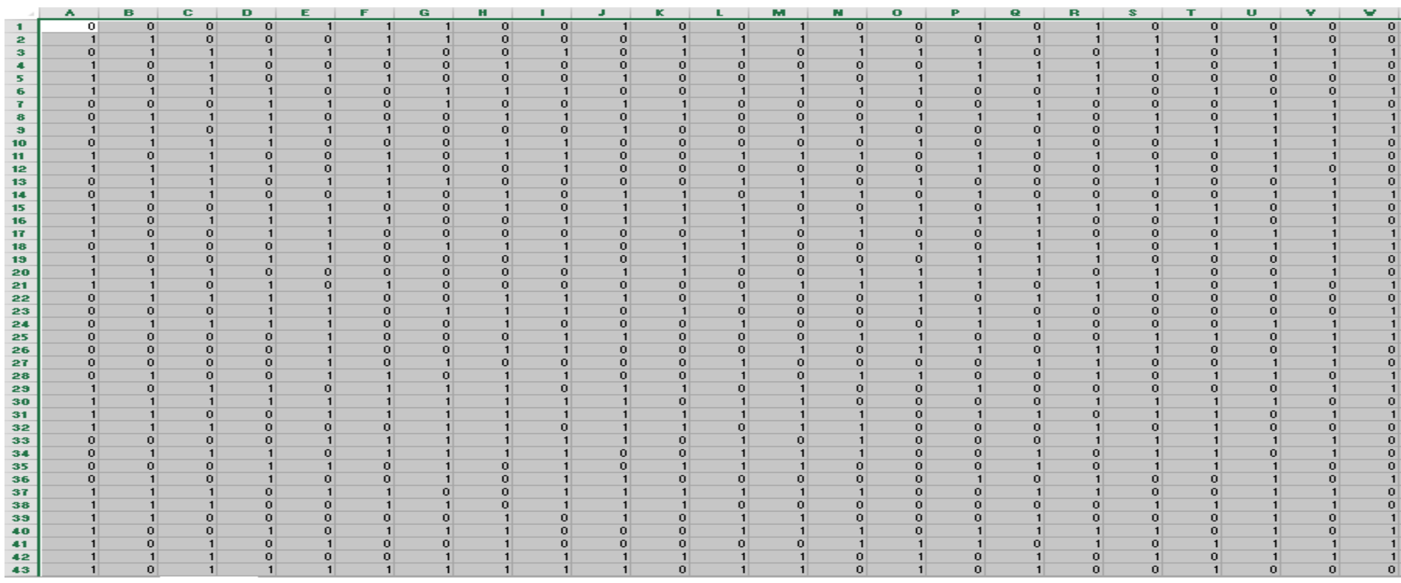
Figure 8Pre-processed Data (30x30)



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
383	27	28	27	28	28	26	23	22	21	21	22	23	23	28	38
384	182	60	41	46	42	58	83	83	84	88	91	92	92	93	94
385	129	33	33	25	27	88	94	87	98	117	123	124	122	121	118
386	29	28	32	39	46	53	60	62	60	57	64	67	61	54	45
387	255	255	255	255	246	133	92	77	79	89	100	106	106	105	104
388	43	32	86	114	115	120	123	129	135	142	145	147	147	147	148
389	57	60	61	65	67	64	70	72	68	72	83	93	102	100	115
390	197	77	32	35	31	40	49	58	84	104	95	78	89	104	109
391	60	84	95	94	94	100	102	108	116	124	135	138	139	139	138
392	255	181	30	29	37	63	74	84	96	109	123	122	117	116	110
393	212	73	50	51	38	49	76	87	105	118	111	106	112	109	96
394	229	147	31	43	45	83	97	97	99	113	120	124	129	127	121
395	51	45	55	77	59	25	34	77	123	136	140	145	144	141	140
396	170	48	40	37	40	54	52	52	65	71	73	66	63	59	57
397	211	98	38	57	62	81	118	125	132	141	152	161	166	170	170
398	49	23	21	19	20	26	25	22	23	60	101	143	150	162	165
399	62	34	32	40	37	27	25	32	62	101	109	98	105	111	128
400															
401															
402															
403															
404	108.49	74.098	55.505	47.7	49.453	53.933	58.665	64.878	71.948	77.715	81.813	84.645	86.49	88.345	88.863
405															
406															
407															
408															
409															
410															
411	19.4875	-51.9025	-56.495	-38.3	-52.5475	-52.0675	-52.335	-52.1225	-51.0525	-58.285	-65.1875	-68.355	-69.51	-66.655	-63.1375
412	53.4875	28.0975	10.505	3.7	6.4525	10.9325	15.665	20.8775	22.9475	17.715	14.8125	24.645	11.49	-12.655	-50.1375
413	3.4875	19.0975	2.505	-10.3	-25.5475	-24.0675	-19.335	-13.1225	-8.0525	-18.285	-36.1875	-39.355	-41.51	-39.655	-39.1375
414	48.4875	18.0975	-9.495	-15.3	-23.5475	-37.0675	-42.335	-21.1225	-48.0525	-51.285	-56.1875	-56.355	-61.51	-67.655	-68.1375
415	3.4875	2.0975	-14.495	-16.3	-21.5475	-35.0675	-40.335	-40.1225	-44.0525	-47.285	-40.1875	-31.355	-44.51	-54.655	-53.1375
416	-13.5125	6.0975	0.505	-19.3	-34.5475	-40.0675	-41.335	-49.1225	-54.0525	-61.285	-65.1875	-67.355	-68.51	-76.655	-69.1375
417	-47.5125	-80.9025	-31.495	-0.3	-1.5475	4.9325	9.665	14.8775	18.9475	19.715	14.8125	22.645	23.49	22.345	32.8625
418	51.4875	28.0975	11.505	-6.3	1.4525	-3.0675	9.665	18.8775	29.9475	37.715	42.8125	50.645	48.49	3.345	5.8625
419	66.4875	29.0975	5.505	2.7	4.4525	14.9325	19.665	24.8775	22.9475	38.715	35.8125	-7.355	-28.51	-36.655	-37.1375
420	14.4875	-11.9025	-18.495	-15.3	-9.5475	-24.0675	-31.335	-44.1225	-63.0525	-58.285	-36.1875	-57.355	-75.51	-90.655	-89.1375
421	28.4875	26.0975	13.505	9.7	13.4525	17.9325	11.665	-13.1225	-34.0525	-53.285	-69.1875	-77.355	-80.51	-89.655	-87.1375
422	44.4875	7.0975	-38.495	-62.3	-58.5475	-55.0675	-56.335	-60.1225	-54.0525	-53.285	-62.1875	-67.355	-73.51	-71.655	-68.1375
423	45.4875	5.0975	-12.495	-5.3	-12.5475	-36.0675	-39.335	-43.1225	-46.0525	-48.285	-35.1875	-46.355	-39.51	-26.655	-47.1375

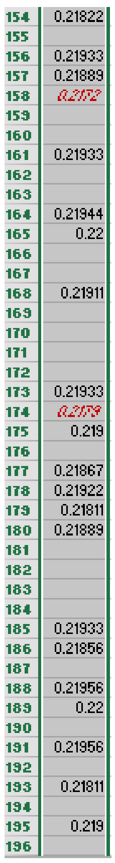
	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD
383	48	48	52	55	51	48	44	42	35	27	25	26	28	29	27
384	93	93	91	90	88	81	76	73	67	51	33	41	43	50	216
385	115	111	102	92	83	70	56	38	30	28	31	31	26	26	27
386	35	34	36	36	38	61	80	62	49	40	31	32	34	35	33
387	106	105	105	106	102	98	94	85	78	71	64	94	168	250	255
388	147	145	139	135	132	127	119	107	89	86	83	41	86	173	182
389	114	110	111	113	102	86	74	65	56	49	50	53	47	36	38
390	111	110	92	84	84	70	64	62	78	87	64	46	28	20	19
391	138	138	133	133	130	125	110	98	92	85	80	78	78	72	79
392	107	110	108	114	116	111	94	78	67	47	35	32	34	31	42
393	72	39	31	32	37	34	36	32	35	38	40	37	33	27	64
394	125	124	121	118	112	107	94	87	81	71	34	33	41	39	75
395	140	134	124	103	88	75	62	52	41	37	37	43	47	44	43
396	60	70	81	102	120	128	124	125	120	114	112	100	42	31	29
397	167	168	166	165	165	163	155	138	117	108	84	73	62	55	55
398	167	165	163	158	154	152	146	139	125	114	112	79	23	38	58
399	136	134	119	108	83	50	32	27	35	34	27	21	21	24	36
400															
401															
402															
403															
404	89.065	88.705	88.193	86.565	82.718	77.318	71	64.435	59.22	54.995	51.57	52.473	57.723	72.133	99.098
405															
406															
407															
408															
409															
410															
411	-57.935	-59.295	-58.8075	-53.435	-53.2825	-45.6825	-37.0	-32.565	-18.78	-15.005	-14.43	-10.5275	-21.2775	-35.8675	-10.9025
412	-41.935	-44.295	-37.8075	-17.435	10.7175	18.3175	23.0	20.435	16.22	11.995	8.57	8.4725	9.7225	24.1325	50.0975
413	-37.935	-38.295	-37.8075	-33.435	-25.2825	-17.6825	-18.0	-19.565	-22.78	-22.005	-11.43	-5.5275	2.7225	15.1325	29.0975
414	-62.935	-61.295	-60.8075	-58.435	-56.2825	-50.6825	-49.0	-51.565	-51.78	-42.005	-20.43	4.4725	-13.2775	-32.8675	-7.9025
415	-46.935	-48.295	-51.8075	-46.435	-43.2825	-47.6825	-40.0	-43.565	-43.78	-25.005	-15.43	-7.5275	-19.2775	-35.8675	-12.9025
416	-60.935	-60.295	-58.8075	-56.435	-52.2825	-39.6825	-31.0	-30.565	-25.78	-34.005	-24.43	-1.5275	10.7225	19.1325	19.0975
417	36.065	30.705	28.1925	20.565	25.7175	21.3175	17.0	13.435	9.22	2.995	-0.43	-37.5275	-90.2775	-71.8675	-47.9025
418	-15.935	-72.295	-64.8075	-47.435	-35.2825	-16.6825	9.0	4.435	7.22	4.995	2.57	-3.5275	2.7225	25.1325	52.0975
419	-32.935	-23.295	-15.8075	-9.435	-4.2825	4.3175	16.0	12.435	10.22	4.995	6.57	9.4725	14.7225	28.1325	54.0975
420	-87.935	-88.295	-82.8075	-80.435	-73.2825	-70.6825	-66.0	-59.565	-47.78	-39.005	-34.43	-19.5275	-38.2775	-31.8675	-0.9025
421	-86.935	-81.295	-73.8075	-67.435	-63.2825	-56.6825	-42.0	-33.565	-22.78	-10.005	3.57	-10.5275	-10.2775	24.1325	51.0975
422	-67.935	-61.295	-54.8075	-54.435	-50.2825	-40.6825	-34.0	-33.565	-21.78	-17.005	0.57	-6.5275	-9.2775	24.1325	19.0975
423	-75.935	-68.295	-40.8075	-49.435	-26.2825	-25.6825	-35.0	-28.565	-26.78	-16.005	-14.43	-8.5275	-2.2775	18.1325	33.0975

Figure 9 1x900 Data, Mean Face Image Data, image data-Mean Face image data



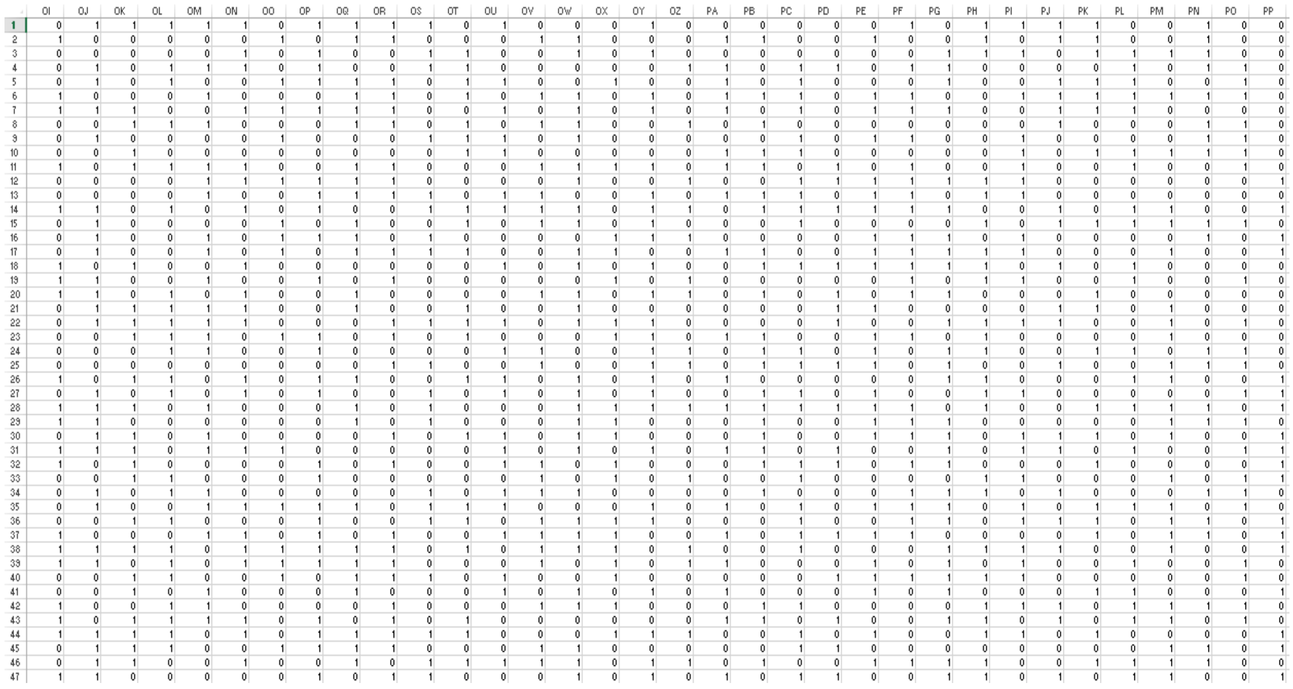
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	0	0	0	0	1	1	1	0	0	1	0	0	1	0	0	1	0	1	0	0	0	0	0
2	1	1	0	0	0	1	1	0	0	0	1	1	1	1	0	0	1	1	1	1	1	0	0
3	0	1	1	1	1	1	0	0	1	0	1	1	0	1	1	1	0	0	1	0	1	1	1
4	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1	1	0	1	1	0
5	1	0	1	0	1	1	0	0	0	1	0	0	1	0	1	1	1	1	0	0	0	0	0
6	1	1	1	1	0	0	1	1	1	0	0	1	1	1	1	0	0	1	0	1	0	0	1
7	0	0	0	1	1	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1	0
8	0	1	1	1	0	0	0	1	1	0	1	0	0	0	1	1	1	0	1	0	1	1	1
9	1	1	0	1	1	1	0	0	0	1	0	0	1	1	0	0	0	0	1	1	1	1	1
10	0	1	1	1	0	0	0	1	1	0	0	0	0	0	1	0	1	0	0	1	1	1	0
11	1	0	1	0	0	1	0	1	1	0	0	1	1	1	0	1	0	1	0	0	1	1	0
12	1	1	1	1	0	1	0	0	1	0	0	0	0	0	0	1	0	0	1	0	1	0	0
13	0	1	1	0	1	1	1	0	0	0	0	1	1	0	1	0	0	0	1	0	0	1	0
14	0	1	1	0	0	1	0	1	0	1	1	0	1	1	0	1	0	0	0	0	1	1	1
15	1	0	0	1	1	0	0	1	0	1	1	1	0	0	1	0	1	1	1	1	0	1	0
16	1	0	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	0	0	1	0	1	0
17	1	0	0	1	1	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	1	1	1
18	0	1	0	0	1	0	1	1	1	0	1	1	0	0	1	0	1	1	0	1	1	1	1
19	1	0	0	1	1	0	0	0	1	0	1	1	0	0	0	1	1	1	0	0	0	1	0
20	1	1	1	0	0	0	0	0	0	1	1	0	0	1	1	1	1	0	1	0	0	1	0
21	1	1	0	1	0	1	0	0	0	0	0	0	1	1	1	1	0	1	1	0	1	0	1
22	0	1	1	1	1	0	0	1	1	1	0	1	0	0	1	0	1	1	0	0	0	0	0
23	0	0	0	1	1	0	1	1	1	0	1	0	0	0	1	1	0	0	0	0	0	0	1
24	0	1	1	1	1	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	1
25	0	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	1	1
26	0	0	0	0	1	0	0	1	1	0	0	0	1	0	1	1	0	1	1	0	0	1	0
27	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	1	1	0	0	1	1	0
28	0	1	0	0	1	1	0	1	1	0	0	1	0	1	1	0	0	1	0	1	1	0	1
29	1	0	1	1	0	1	1	1	0	1	1	0	1	0	0	1	0	0	0	0	0	1	1
30	1	1	1	1	1	1	1	1	1	1	0	1	1	0	0	0	0	1	1	1	1	0	0
31	1	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	0	1	1
32	1	1	1	0	0	0	1	1	0	1	1	0	1	1	0	0	0	1	0	1	0	0	0
33	0	0	0	0	1	1	1	1	1	1	0	1	0	1	0	0	0	1	1	1	1	1	0
34	0	1	1	1	0	1	1	1	1	0	0	1	1	1	0	0	1	0	1	1	0	1	0
35	0	0	0	1	1	0	1	0	1	0	1	1	1	0	0	0	1	0	1	1	1	0	0
36	0	1	0	1	0	0	1	0	1	1	0	0	0	0	0	1	0	1	0	0	1	0	1
37	1	1	1	0	1	1	0	0	1	1	1	1	1	1	0	0	1	1	0	0	1	1	0
38	1	1	1	0	0	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	1	1	0
39	1	1	0	0	0	0	0	1	0	1	0	1	1	0	0	0	1	0	0	0	1	0	1
40	1	0	0	1	0	1	1	1	0	0	0	1	1	1	0	1	1	1	1	0	1	0	1
41	1	0	1	0	1	0	0	1	0	0	0	0	0	1	1	1	0	1	0	1	1	1	1
42	1	1	1	0	0	0	1	1	1	1	1	1	1	0	1	0	1	0	1	0	1	1	1
43	1	0	1	1	1	1	1	1	1	1	0	1	1	0	1	0	1	0	0	1	0	0	0

Figure 10 Genetic Data for each feature



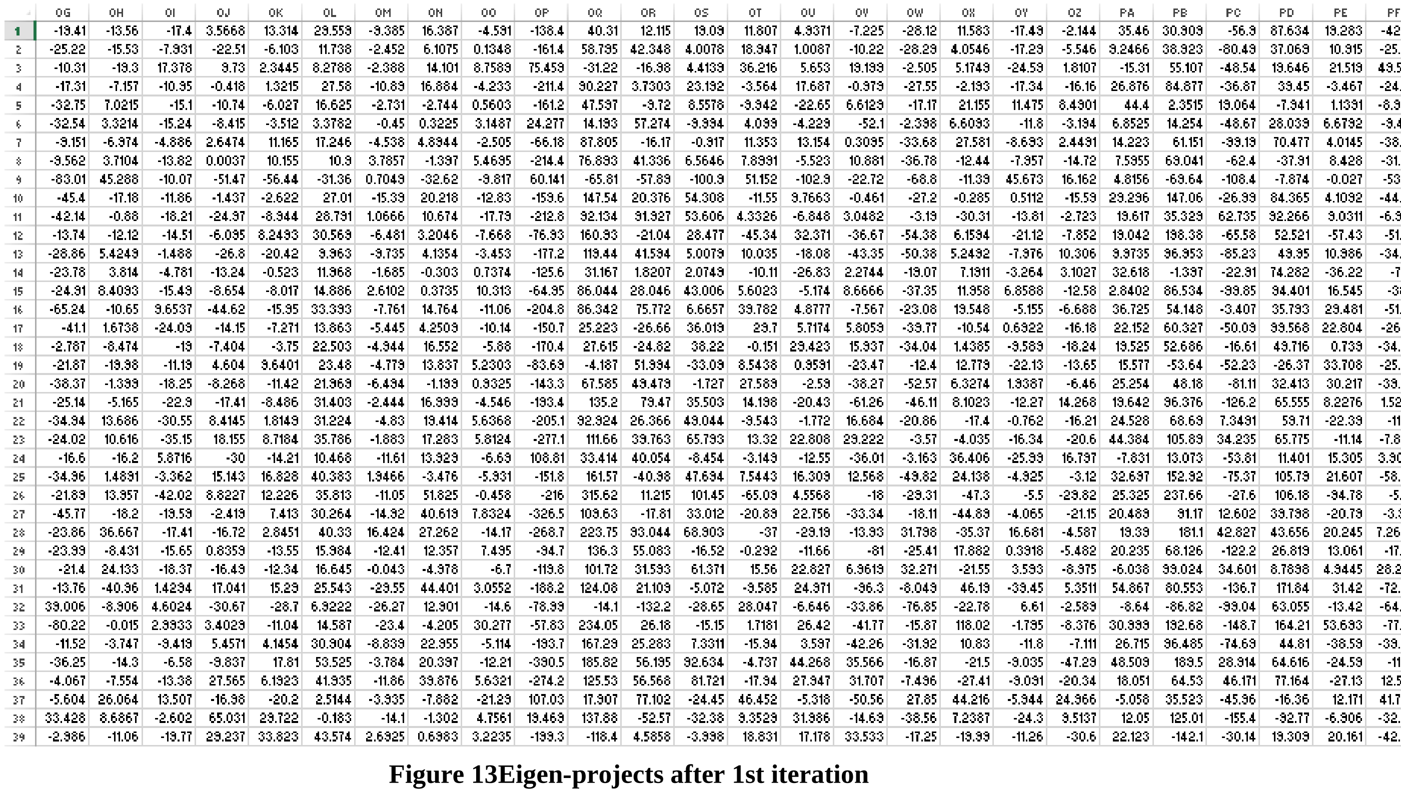
154	0.21822
155	
156	0.21933
157	0.21889
158	*0.2172*
159	
160	
161	0.21933
162	
163	
164	0.21944
165	0.22
166	
167	
168	0.21911
169	
170	
171	
172	
173	0.21933
174	*0.2179*
175	0.219
176	
177	0.21867
178	0.21922
179	0.21811
180	0.21889
181	
182	
183	
184	
185	0.21933
186	0.21856
187	
188	0.21956
189	0.22
190	
191	0.21956
192	
193	0.21811
194	
195	0.219
196	

Figure 11Fitting Function Values After wiping out fitness values<0.22



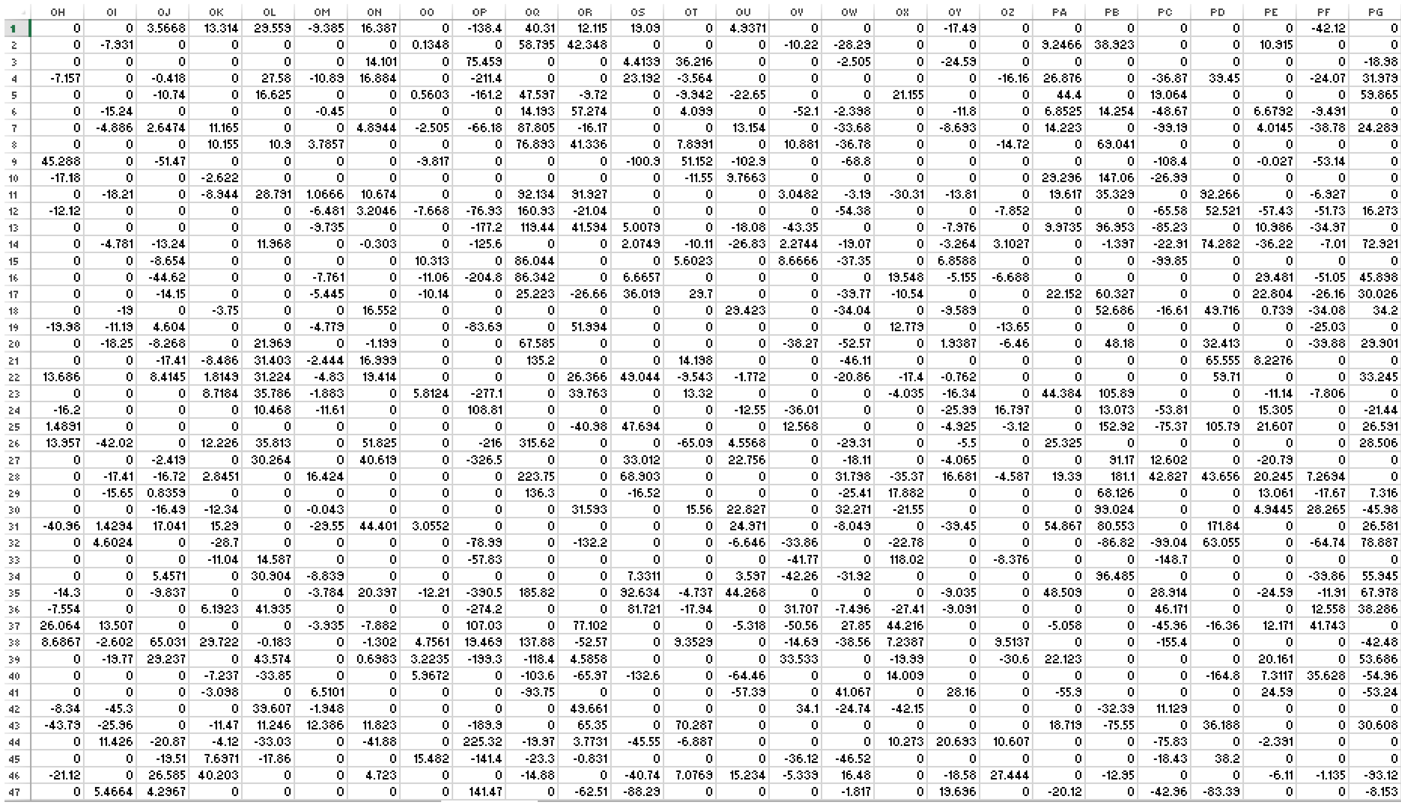
	OI	OJ	OK	OL	OM	ON	OO	OP	OQ	OR	OS	OT	OU	OV	OW	OX	OY	OZ	PA	PB	PC	PD	PE	PF	PG	PH	PI	PJ	PK	PL	PM	PN	PO	PP
1	0	1	1	1	1	1	0	1	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	1	0	1	1	1	1	0	0	1	0	0
2	1	0	0	0	0	0	1	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	1	0	0	1	0	1	1	0	0	1	0	0
3	0	0	0	0	0	1	0	1	0	0	1	1	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	0	0
4	0	1	0	1	1	1	0	1	0	0	1	1	0	0	0	0	0	1	1	0	1	1	0	1	1	0	0	0	0	1	0	1	1	0
5	0	1	0	1	0	0	1	1	1	1	0	1	1	0	0	1	0	0	1	0	1	0	0	0	1	0	0	1	1	1	0	0	1	0
6	1	0	0	0	1	0	0	0	1	1	0	1	0	1	1	0	1	0	1	1	1	0	1	1	0	0	1	1	1	1	1	1	1	0
7	1	1	1	0	0	1	1	1	1	1	0	0	1	0	1	0	1	0	1	0	1	0	1	1	1	0	0	1	1	1	0	0	0	0
8	0	0	1	1	1	0	0	0	1	1	0	1	0	1	1	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	1	1	0
9	0	1	0	0	0	0	1	0	0	0	1	1	1	0	1	0	0	0	0	0	1	0	1	1	0	0	1	0	0	0	0	1	1	0
10	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	1	0	1	1	1	1	1	0
11	1	0	1	1	1	1	0	0	1	1	0	0	0	1	1	1	1	0	1	1	0	1	0	1	0	0	1	0	0	1	0	0	1	0
12	0	0	0	0	1	1	1	1	1	1	0	0	0	0	1	0	0	1	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	1
13	0	0	0	0	1	0	0	1	1	1	1	0	1	1	0	0	1	0	1	1	1	0	1	1	0	1	1	0	0	0	0	0	0	0
14	1	1	0	1	0	1	0	1	0	0	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	0	0	1	0	1	1	0	0	1
15	0	1	0	0	0	0	1	0	1	0	0	1	0	1	1	0	1	0	0	0	1	0	0	0	0	1	0	1	1	1	1	1	1	0
16	0	1	0	0	1	0	1	1	1	0	1	0	0	0	0	1	1	1	0	0	0	0	1	1	1	0	1	0	0	0	0	1	0	1
17	0	1	0	0	1	0	1	0	1	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	1	1	1	0	0	0	1	0	0	1
18	1	0	1	0	0	1	0	0	0	0	0	0	1	0	1	0	1	0	0	1	1	1	1	1	1	1	0	1	0	1	0	0	0	0
19	1	1	0	0	1	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	1	0	0	1	0	0	1	0
20	1	1	0	1	0	1	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0	1	0	1	1	0	0	0	1	0	0	0	0	0
21	0	1	1	1	1	1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	0	0
22	0	1	1	1	1	1	0	0	0	1	1	1	1	0	1	1	1	0	0	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0
23	0	0	1	1	1	0	1	1	0	1	0	1	0	0	0	1	1	0	1	1	0	0	1	1	0	1	0	0	0	0	1	0	0	0
24	0	0	0	1	1	0	0	1	0	0	0	0	1	1	0	0	1	1	0	1	1	0	1	0	1	1	0	0	1	1	0	1	1	0
25	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	1	1	0	1	1	1	1	0	1	0	0	1	0	0	1	1	1	0
26	1	0	1	1	0	1	0	1	1	0	0	1	1	0	1	0	1	0	1	0	0	0	0	0	1	1	0	0	0	1	1	0	0	1
27	0	1	0	1	0	1	0	1	0	0	1	0	1	0	1	0	1	0	0	1	1	0	1	0	0	1	1	0	0	0	0	0	1	1
28	1	1	1	0	1	0	0	0	1	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	0	1	0	0	1	1	1	1	0	1
29	1	1	0	0	0	0	0	0	1	0	1	0	0	0	1	1	0	0	0	1	0	0	1	1	1	0	0	0	0	0	1	1	1	0
30	0	1	1	0	1	0	0	0	0	1	0	1	1	0	1	1	0	0	0	1	0	0	1	1	1	0	1	1	1	0	1	0	0	1
31	1	1	1	0	1	1	1	0	0	0	0	0	1	0	1	0	1	0	1	1	0	1	0	0	1	0	1	1	0	1	0	0	1	1
32	1	0	1	0	0	0	0	1	0	1	0	0	1	1	0	1	0	0	0	1	1	1	0	1	1	0	0	0	1	0	0	0	0	1
33	0	0	1	1	0	0	0	1	0	0	0	0	0	1	0	1	0	1	0	0	1	0	0	0	0	1	1	0	0	0	1	0	1	1
34	0	1	0	1	1	0	0	0	0	0	1	0	1	1	1	0	0	0	0	1	0	0	0	1	1	1	0	1	0	0	0	1	1	0
35	0	1	0	0	1	1	1	1	1	0	1	1	1	0	0	0	1	0	1	0	1	0	1	1	1	0	1	0	1	1	1	0	1	0
36	0	0	1	1	0	0	0	1	0	0	1	1	0	1	1	1	1	0	0	0	1	0	0	1	1	0	1	1	1	0	1	1	0	1
37	1	0	0	0	1	1	0	1	0	1	0	0	1	1	1	1	0	0	1	0	1	1	1	1	0	0	0	0	1	0	1	1	0	1
38	1	1	1	1	0	1	1	1	1	1	0	1	0	1	1	1	0	1	0	0	1	0	0	0	1	1	1	1	0	0	1	0	0	1
39	1	1	0	1	0	1	1	1	1	1	0	0	0	1	0	1	0	1	1	0	0	0	1	0	1	0	1	0	0	1	0	0	1	0
40	0	0	1	1	0	0	1	0	1	1	1	0	1	0	0	1	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	1	0
41	0	0	1	0	1	0	0	0	1	0	0	0	1	0	1	0	1	0	1	0	0	0	1	0	1	0	0	0	1	1	0	0	0	1
42	1	0	0	1	1	0	0	0	0	1	0	0	0	1	1	1	0	0	0	1	1	0	0	0	0	1	1	1	0	1	1	1	1	0
43	1	0	1	1	1	1	0	1	0	1	0	1	0	0	0	0	0	0	1	1	0	1	0	0	1	1	0	1	0	1	1	0	1	0
44	1	1	1	1	0	1	0	1	1	1	1	1	0	0	0	1	1	1	0	0	1	0	1	0	0	1	1	0	1	0	0	0	0	1
45	0	1	1	1	0	0	1	1	1	1	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	1	0	1
46	0	1	1	0	0	1	0	0	1	0	1	1	1	1	1	0	1	1	0	1	0	0	1	1	1	1	0	0	1	1	1	1	0	0
47	1	1	0	0	0	0	0	1	0	1	1	0	0	0	1	0	1	0	1	0	1	1	0	0	1	1	0	1	0	1	1	0	0	1

Figure 12Genetic Data after 1st iteration



	OG	OH	OI	OJ	OK	OL	OM	ON	OO	OP	OQ	OR	OS	OT	OU	OV	OW	OX	OY	OZ	PA	PB	PC	PD	PE	PF
1	-19.41	-13.56	-17.4	3.5668	13.314	29.559	-9.385	16.387	-4.591	-138.4	40.31	12.115	19.09	11.807	4.9371	-7.225	-28.12	11.583	-17.49	-2.144	35.46	30.909	-56.9	87.634	19.283	-42
2	-25.22	-15.53	-7.931	-22.51	-6.103	11.738	-2.452	6.1075	0.1348	-161.4	58.795	42.348	4.0078	18.947	1.0087	-10.22	-28.29	4.0546	-17.29	-5.546	9.2466	38.923	-80.49	37.069	10.915	-25.
3	-10.31	-19.3	17.378	9.73	2.3445	8.2788	-2.388	14.101	8.7589	75.459	-31.22	-16.98	4.4139	36.216	5.653	19.199	-2.505	5.1749	-24.59	1.8107	-15.31	55.107	-48.54	19.646	21.519	49.5
4	-17.31	-7.157	-10.95	-0.418	1.3215	27.58	-10.89	16.884	-4.233	-211.4	90.227	3.7303	23.192	-3.564	17.687	-0.979	-27.55	-2.193	-17.34	-16.16	26.876	84.877	-36.87	39.45	-3.467	-24.
5	-32.75	7.0215	-15.1	-10.74	-6.027	16.625	-2.731	-2.744	0.5603	-161.2	47.597	-9.72	8.5578	-9.942	-22.65	6.6129	-17.17	21.155	11.475	8.4901	44.4	2.3515	19.064	-7.941	1.1391	-8.9
6	-32.54	3.3214	-15.24	-8.415	-3.512	3.3782	-0.45	0.3225	3.1487	24.277	14.193	57.274	-9.994	4.099	-4.229	-52.1	-2.398	6.6093	-11.8	-3.194	6.8525	14.254	-48.67	28.039	6.6792	-9.4
7	-9.151	-6.974	-4.886	2.6474	11.165	17.246	-4.538	4.8944	-2.505	-66.18	87.805	-16.17	-0.917	11.353	13.154	0.3095	-33.68	27.581	-8.693	2.4491	14.223	61.151	-99.19	70.477	4.0145	-38.
8	-9.562	3.7104	-13.82	0.0037	10.155	10.9	3.7857	-1.397	5.4695	-214.4	76.893	41.336	6.5646	7.8991	-5.523	10.881	-36.78	-12.44	-7.957	-14.72	7.5955	69.041	-62.4	-37.91	8.428	-31.
9	-83.01	45.288	-10.07	-51.47	-56.44	-31.36	0.7049	-32.62	-9.817	60.141	-65.81	-57.89	-100.9	51.152	-102.9	-22.72	-68.8	-11.39	45.673	16.162	4.8156	-69.64	-108.4	-7.874	-0.027	-53
10	-45.4	-17.18	-11.86	-1.437	-2.622	27.01	-15.39	20.218	-12.83	-159.6	147.54	20.376	54.308	-11.55	9.7663	-0.461	-27.2	-0.285	0.5112	-15.59	29.296	147.06	-26.99	84.365	4.1092	-44.
11	-42.14	-0.88	-18.21	-24.97	-8.944	28.791	1.0666	10.674	-17.79	-212.8	92.134	91.927	53.606	4.3326	-6.848	3.0482	-3.19	-30.31	-13.81	-2.723	19.617	35.329	62.735	92.266	9.0311	-6.3
12	-13.74	-12.12	-14.51	-6.095	8.2493	30.569	-6.481	3.2046	-7.668	-76.93	160.93	-21.04	28.477	-45.34	32.371	-36.67	-54.38	6.1594	-21.12	-7.852	19.042	198.38	-65.58	52.521	-57.43	-51.
13	-28.86	5.4249	-1.488	-26.8	-20.42	9.963	-9.735	4.1354	-3.453	-177.2	119.44	41.594	5.0079	10.035	-18.08	-43.35	-50.38	5.2492	-7.976	10.306	9.9735	96.953	-85.23	49.95	10.986	-34.
14	-23.78	3.814	-4.781	-13.24	-0.523	11.968	-1.685	-0.303	0.7374	-125.6	31.167	1.8207	2.0749	-10.11	-26.83	2.2744	-19.07	7.1911	-3.264	3.1027	32.618	-1.397	-22.91	74.282	-36.22	-7
15	-24.91	8.4093	-15.49	-8.654	-8.017	14.886	2.6102	0.3735	10.313	-64.95	86.044	28.046	43.006	5.6023	-5.174	8.6666	-37.35	11.958	6.8588	-12.58	2.8402	86.534	-99.85	94.401	16.545	-3
16	-65.24	-10.65	9.6537	-44.62	-15.95	33.393	-7.761	14.764	-11.06	-204.8	86.342	75.772	6.6657	39.782	4.8777	-7.567	-23.08	19.548	-5.155	-6.688	36.725	54.148	-3.407	35.793	29.481	-51.
17	-41.1	1.6738	-24.09	-14.15	-7.271	13.863	-5.445	4.2509	-10.14	-150.7	25.223	-26.66	36.019	29.7	5.7174	5.8059	-39.77	-10.54	0.6922	-16.18	22.152	60.327	-50.09	99.568	22.804	-26
18	-2.787	-8.474	-19	-7.404	-3.75	22.503	-4.344	16.552	-5.88	-170.4	27.615	-24.82	38.22	-0.151	29.423	15.937	-34.04	1.4385	-9.589	-18.24	19.525	52.686	-16.61	49.716	0.739	-34.
19	-21.87	-19.98	-11.19	4.604	9.6401	23.48	-4.779	13.837	5.2303	-83.69	-4.187	51.994	-33.09	8.5438	0.9591	-23.47	-12.4	12.779	-22.13	-13.65	15.577	-53.64	-52.23	-26.37	33.708	-25.
20	-38.37	-1.399	-18.25	-8.268	-11.42	21.969	-6.494	-1.199	0.9325	-143.3	67.585	49.479	-1.727	27.589	-2.59	-38.27	-52.57	6.3274	1.9387	-6.46	25.254	48.18	-81.11	32.413	30.217	-39.
21	-25.14	-5.165	-22.9	-17.41	-8.486	31.403	-2.444	16.999	-4.546	-193.4	135.2	79.47	35.503	14.198	-20.43	-61.26	-46.11	8.1023	-12.27	14.268	19.642	96.376	-126.2	65.555	8.2276	1.52
22	-34.94	13.686	-30.55	8.4145	1.8149	31.224	-4.83	19.414	5.6368	-205.1	92.924	26.366	49.044	-9.543	-1.772	16.684	-20.86	-17.4	-0.762	-16.21	24.528	68.69	7.3491	59.71	-22.39	-11
23	-24.02	10.616	-35.15	18.155	8.7184	35.786	-1.883	17.283	5.8124	-277.1	111.66	39.763	65.793	13.32	22.808	29.222	-3.57	-4.035	-16.34	-20.6	44.384	105.89	34.235	65.775	-11.14	-7.8
24	-16.6	-16.2	5.8716	-30	-14.21	10.468	-11.61	13.929	-6.69	108.81	33.414	40.054	-8.454	-3.149	-12.55	-36.01	-3.163	36.406	-25.99	16.797	-7.831	13.073	-53.81	11.401	15.305	3.90
25	-34.96	1.4891	-3.362	15.143	16.828	40.383	1.9466	-3.476	-5.931	-151.8	161.57	-40.98	47.694	7.5443	16.309	12.568	-49.82	24.138	-4.925	-3.12	32.697	152.92	-75.37	105.79	21.607	-58.
26	-21.89	13.957	-42.02	8.8227	12.226	35.813	-11.05	51.825	-0.458	-216	315.62	11.215	101.45	-65.09	4.5568	-18	-29.31	-47.3	-5.5	-29.82	25.325	237.66	-27.6	106.18	-94.78	-5.
27	-45.77	-18.2	-19.59	-2.419	7.413	30.264	-14.92	40.619	7.8324	-326.5	109.63	-17.81	33.012	-20.89	22.756	-33.34	-18.11	-44.89	-4.065	-21.15	20.489	91.17	12.602	39.798	-20.79	-3.3
28	-23.86	36.667	-17.41	-16.72	2.8451	40.33	16.424	27.262	-14.17	-268.7	223.75	93.044	68.903	-37	-29.19	-13.93	31.798	-35.37	16.681	-4.587	19.39	181.1	42.827	43.656	20.245	7.26
29	-23.99	-8.431	-15.65	0.8359	-13.55	15.984	-12.41	12.357	7.495	-94.7	136.3	55.083	-16.52	-0.292	-11.66	-81	-25.41	17.882	0.3918	-5.482	20.235	68.126	-122.2	26.819	13.061	-17.
30	-21.4	24.133	-18.37	-16.49	-12.34	16.645	-0.043	-4.978	-6.7	-119.8	101.72	31.593	61.371	15.56	22.827	6.9619	32.271	-21.55	3.593	-8.975	-6.038	99.024	34.601	8.7898	4.9445	28.2
31	-13.76	-40.96	1.4294	17.041	15.29	25.543	-29.55	44.401	3.0552	-188.2	124.08	21.109	-5.072	-9.585	24.971	-96.3	-8.049	46.19	-39.45	5.3511	54.867	80.553	-136.7	171.84	31.42	-72.
32	39.006	-8.906	4.6024	-30.67	-28.7	6.9222	-26.27	12.901	-14.6	-78.99	-14.1	-132.2	-28.65	28.047	-6.646	-33.86	-76.85	-22.78	6.61	-2.589	-8.64	-86.82	-99.04	63.055	-13.42	-64.
33	-80.22	-0.015	2.9933	3.4029	-11.04	14.587	-23.4	-4.205	30.277	-57.83	234.05	26.18	-15.15	1.7181	26.42	-41.77	-15.87	118.02	-1.795	-8.376	30.999	192.68	-148.7	164.21	53.693	-77.
34	-11.52	-3.747	-9.419	5.4571	4.1454	30.904	-8.839	22.955	-5.114	-193.7	167.29	25.283	7.3311	-15.94	3.597	-42.26	-31.92	10.83	-11.8	-7.111	26.715	96.485	-74.69	44.81	-38.59	-39.
35	-36.25	-14.3	-6.58	-9.837	17.81	53.525	-3.784	20.397	-12.21	-390.5	185.82	56.195	92.634	-4.737	44.268	35.566	-16.87	-21.5	-9.035	-47.29	48.509	189.5	28.914	64.616	-24.59	-11
36	-4.067	-7.554	-13.38	27.565	6.1923	41.935	-11.86	39.876	5.6321	-274.2	125.53	56.568	81.721	-17.94	27.947	31.707	-7.496	-27.41	-9.091	-20.34	18.051	64.53	46.171	77.164	-27.13	12.5
37	-5.604	26.064	13.507	-16.98	-20.2	2.5144	-3.935	-7.882	-21.29	107.03	17.907	77.102	-24.45	46.452	-5.318	-50.56	27.85	44.216	-5.944	24.966	-5.058	35.523	-45.96	-16.36	12.171	41.7
38	33.428	8.6867	-2.602	65.031	29.722	-0.183	-14.1	-1.302	4.7561	19.469	137.88	-52.57	-32.38	9.3529	31.986	-14.69	-38.56	7.2387	-24.3	9.5137	12.05	125.01	-155.4	-92.77	-6.906	-32.
39	-2.986	-11.06	-19.77	29.237	33.823	43.574	2.6925	0.6983	3.2235	-199.3	-118.4	4.5858	-3.998	18.831	17.178	33.533	-17.25	-19.99	-11.26	-30.6	22.123	-142.1	-30.14	19.309	20.161	-42.

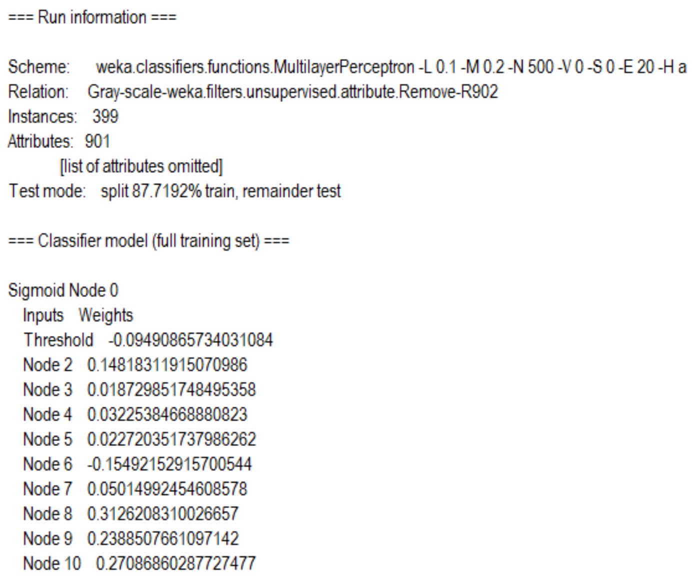
Figure 13Eigen-projects after 1st iteration



	OH	OI	OJ	OK	OL	OM	ON	OO	OP	OQ	OR	OS	OT	OU	OV	OW	OX	OY	OZ	PA	PB	PC	PD	PE	PF	PG
1	0	0	3.5668	13.314	29.559	-9.385	16.387	0	-138.4	40.31	12.115	19.09	0	4.9371	0	0	0	-17.49	0	0	0	0	0	0	-42.12	0
2	0	-7.931	0	0	0	0	0	0.1348	0	58.795	42.348	0	0	0	-10.22	-28.29	0	0	0	9.2466	38.923	0	0	10.915	0	0
3	0	0	0	0	0	0	14.101	0	75.459	0	0	4.4139	36.216	0	0	-2.505	0	-24.59	0	0	0	0	0	0	0	-18.98
4	-7.157	0	-0.418	0	27.58	-10.89	16.884	0	-211.4	0	0	23.192	-3.564	0	0	0	0	0	-16.16	26.876	0	-36.87	39.45	0	-24.07	31.979
5	0	0	-10.74	0	16.625	0	0	0.5603	-161.2	47.597	-9.72	0	-9.942	-22.65	0	0	21.155	0	0	44.4	0	19.064	0	0	0	59.865
6	0	-15.24	0	0	0	-0.45	0	0	0	14.193	57.274	0	4.099	0	-52.1	-2.398	0	-11.8	0	6.8525	14.254	-48.67	0	6.6792	-9.491	0
7	0	-4.886	2.6474	11.165	0	0	4.8944	-2.505	-66.18	87.805	-16.17	0	0	13.154	0	-33.68	0	-8.693	0	14.223	0	-99.19	0	4.0145	-38.78	24.289
8	0	0	0	10.155	10.9	3.7857	0	0	0	76.893	41.336	0	7.8931	0	10.881	-36.78	0	0	-14.72	0	69.041	0	0	0	0	0
9	45.288	0	-51.47	0	0	0	0	-9.817	0	0	0	-100.9	51.152	-102.9	0	-68.8	0	0	0	0	0	-108.4	0	-0.027	-53.14	0
10	-17.18	0	0	-2.622	0	0	0	0	0	0	0	0	-11.55	9.7663	0	0	0	0	0	29.296	147.06	-26.99	0	0	0	0
11	0	-18.21	0	-8.944	28.791	1.0666	10.674	0	0	92.134	91.927	0	0	0	3.0482	-3.19	-30.31	-13.81	0	19.617	35.329	0	92.266	0	-6.927	0
12	-12.12	0	0	0	0	-6.481	3.2046	-7.668	-76.93	160.93	-21.04	0	0	0	0	-54.38	0	0	-7.852	0	0	-65.58	52.521	-57.43	-51.73	16.273
13	0	0	0	0	0	-9.735	0	0	-177.2	119.44	41.594	5.0079	0	-18.08	-43.35	0	0	-7.976	0	9.9735	96.953	-85.23	0	10.986	-34.97	0
14	0	-4.781	-13.24	0	11.968	0	-0.303	0	-125.6	0	0	2.0749	-10.11	-26.83	2.2744	-19.07	0	-3.264	3.1027	0	-1.397	-22.91	74.282	-36.22	-7.01	72.921
15	0	0	-8.654	0	0	0	0	10.313	0	86.044	0	0	5.6023	0	8.6666	-37.35	0	6.8588	0	0	0	-99.85	0	0	0	0
16	0	0	-44.62	0	0	-7.761	0	-11.06	-204.8	86.342	0	6.6657	0	0	0	0	19.548	-5.155	-6.688	0	0	0	0	29.481	-51.05	45.898
17	0	0	-14.15	0	0	-5.445	0	-10.14	0	25.223	-26.66	36.019	29.7	0	0	-39.77	-10.54	0	0	22.152	60.327	0	0	22.804	-26.16	30.026
18	0	-19	0	-3.75	0	0	16.552	0	0	0	0	0	0	29.423	0	-34.04	0	-9.589	0	0	52.686	-16.61	49.716	0.739	-34.08	34.2
19	-19.98	-11.19	4.604	0	0	-4.779	0	0	-83.69	0	51.994	0	0	0	0	0	12.779	0	-13.65	0	0	0	0	0	-25.03	0
20	0	-18.25	-8.268	0	21.969	0	-1.199	0	0	67.585	0	0	0	0	-38.27	-52.57	0	1.9387	-6.46	0	48.18	0	32.413	0	-39.88	29.901
21	0	0	-17.41	-8.486	31.403	-2.444	16.999	0	0	135.2	0	0	14.198	0	0	-46.11	0	0	0	0	0	0	65.555	8.2276	0	0
22	13.686	0	8.4145	1.8149	31.224	-4.83	19.414	0	0	0	26.366	49.044	-9.543	-1.772	0	-20.86	-17.4	-0.762	0	0	0	0	59.71	0	0	33.245
23	0	0	0	8.7184	35.786	-1.883	0	5.8124	-277.1	0	39.763	0	13.32	0	0	0	-4.035	-16.34	0	44.384	105.89	0	0	-11.14	-7.806	0
24	-16.2	0	0	0	10.468	-11.61	0	0	108.81	0	0	0	0	-12.55	-36.01	0	0	-25.99	16.797	0	13.073	-53.81	0	15.305	0	-21.44
25	1.4891	0	0	0	0	0	0	0	0	0	-40.98	47.694	0	0	12.568	0	0	-4.925	-3.12	0	152.92	-75.37	105.79	21.607	0	26.591
26	13.957	-42.02	0	12.226	35.813	0	51.825	0	-216	315.62	0	0	-65.09	4.5568	0	-29.31	0	-5.5	0	25.325	0	0	0	0	0	28.506
27	0	0	-2.419	0	30.264	0	40.619	0	-326.5	0	0	33.012	0	22.756	0	-18.11	0	-4.065	0	0	91.17	12.602	0	-20.79	0	0
28	0	-17.41	-16.72	2.8451	0	16.424	0	0	0	223.75	0	68.903	0	0	0	31.798	-35.37	16.681	-4.587	19.39	181.1	42.827	43.656	20.245	7.2694	0
29	0	-15.65	0.8359	0	0	0	0	0	0	136.3	0	-16.52	0	0	0	-25.41	17.882	0	0	0	68.126	0	0	13.061	-17.67	7.316
30	0	0	-16.49	-12.34	0	-0.043	0	0	0	0	31.593	0	15.56	22.827	0	32.271	-21.55	0	0	0	99.024	0	0	4.9445	28.265	-45.98
31	-40.96	1.4294	17.041	15.29	0	-29.55	44.401	3.0552	0	0	0	0	0	24.971	0	-8.049	0	-39.45	0	54.867	80.553	0	171.84	0	0	26.581
32	0	4.6024	0	-28.7	0	0	0	0	-78.99	0	-132.2	0	0	-6.646	-33.86	0	-22.78	0	0	0	-86.82	-99.04	63.055	0	-64.74	78.887
33	0	0	0	-11.04	14.587	0	0	0	-57.83	0	0	0	0	0	-41.77	0	118.02	0	-8.376	0	0	-148.7	0	0	0	0
34	0	0	5.4571	0	30.904	-8.839	0	0	0	0	0	7.3311	0	3.597	-42.26	-31.92	0	0	0	0	96.485	0	0	0	-39.86	55.945
35	-14.3	0	-9.837	0	0	-3.784	20.397	-12.21	-390.5	185.82	0	92.634	-4.737	44.268	0	0	0	-9.035	0	48.509	0	28.914	0	-24.59	-11.91	67.978
36	-7.554	0	0	6.1923	41.935	0	0	0	-274.2	0	0	81.721	-17.94	0	31.707	-7.496	-27.41	-9.091	0	0	0	46.171	0	0	12.558	38.286
37	26.064	13.507	0	0	0	-3.935	-7.882	0	107.03	0	77.102	0	0	-5.318	-50.56	27.85	44.216	0	0	-5.058	0	-45.96	-16.36	12.171	41.743	0
38	8.6867	-2.602	65.031	29.722	-0.183	0	-1.302	4.7561	19.469	137.88	-52.57	0	9.3529	0	-14.69	-38.56	7.2387	0	9.5137	0	0	-155.4	0	0	0	-42.48
39	0	-19.77	29.237	0	43.574	0	0.6983	3.2235	-199.3	-118.4	4.5858	0	0	0	33.533	0	-19.99	0	-30.6	22.123	0	0	0	20.161	0	53.686
40	0	0	0	-7.237	-33.85	0	0	5.9672	0	-103.6	-65.97	-132.6	0	-64.46	0	0	14.009	0	0	0	0	0	-164.8	7.3117	35.628	-54.96
41	0	0	0	-3.098	0	6.5101	0	0	0	-93.75	0	0	0	-57.39	0	41.067	0	28.16	0	-55.9	0	0	0	24.59	0	-53.24
42	-8.34	-45.3	0	0	39.607	-1.948	0	0	0	0	49.661	0	0	0	34.1	-24.74	-42.15	0	0	0	-32.39	11.129	0	0	0	0
43	-43.79	-25.96	0	-11.47	11.246	12.386	11.823	0	-189.9	0	65.35	0	70.287	0	0	0	0	0	0	18.719	-75.55	0	36.188	0	0	30.608
44	0	11.426	-20.87	-4.12	-33.03	0	-41.88	0	225.32	-19.97	3.7731	-45.55	-6.887	0	0	0	10.273	20.693	10.607	0	0	-75.83	0	-2.391	0	0
45	0	0	-19.51	7.6971	-17.86	0	0	15.482	-141.4	-23.3	-0.831	0	0	0	-36.12	-46.52	0	0	0	0	0	-18.43	38.2	0	0	0
46	-21.12	0	26.585	40.203	0	0	4.723	0	0	-14.88	0	-40.74	7.0769	15.234	-5.339	16.48	0	-18.58	27.444	0	-12.95	0	0	-6.11	-1.135	-93.12
47	0	5.4664	4.2967	0	0	0	0	0	141.47	0	-62.51	-88.29	0	0	0	-1.817	0	19.696	0	-20.12	0	-42.96	-83.39	0	0	-8.153

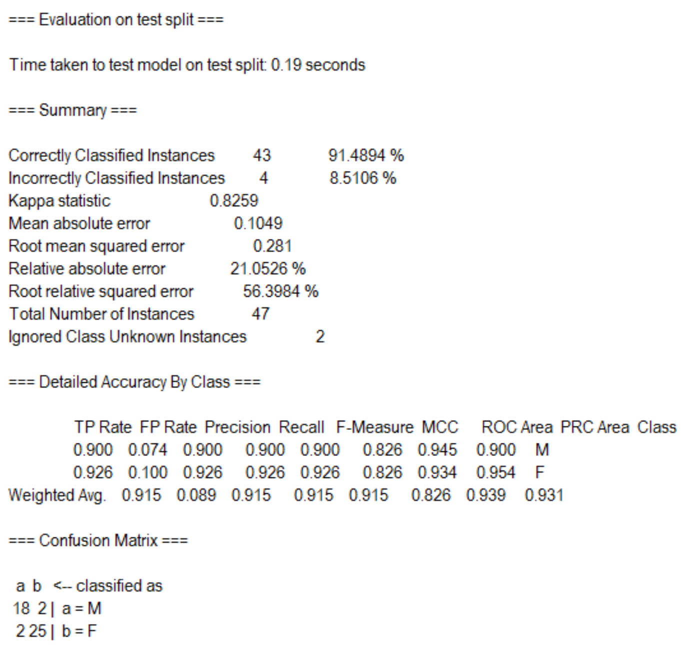
Figure 14Eigen-features after 1st iteration

Final Outputs



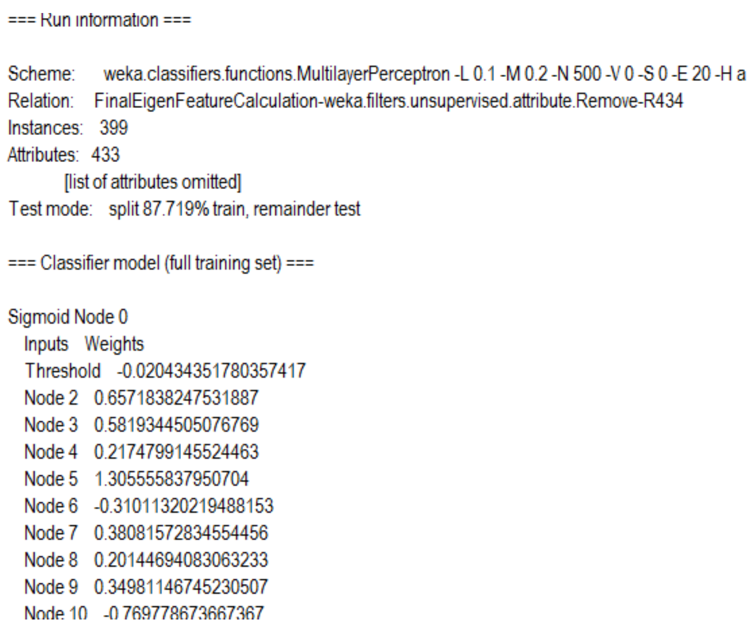
```
=== Run information ===

Scheme:      weka.classifiers.functions.MultilayerPerceptron -L 0.1 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
Relation:    Gray-scale-weka.filters.unsupervised.attribute.Remove-R902
Instances:   399
Attributes:  901
             [list of attributes omitted]
Test mode:   split 87.7192% train, remainder test

=== Classifier model (full training set) ===

Sigmoid Node 0
    Inputs    Weights
    Threshold    -0.09490865734031084
    Node 2    0.14818311915070986
    Node 3    0.018729851748495358
    Node 4    0.03225384668880823
    Node 5    0.022720351737986262
    Node 6    -0.15492152915700544
    Node 7    0.05014992454608578
    Node 8    0.3126208310026657
    Node 9    0.2388507661097142
    Node 10    0.27086860287727477
```

Figure 15Grey-scaleFinalResult



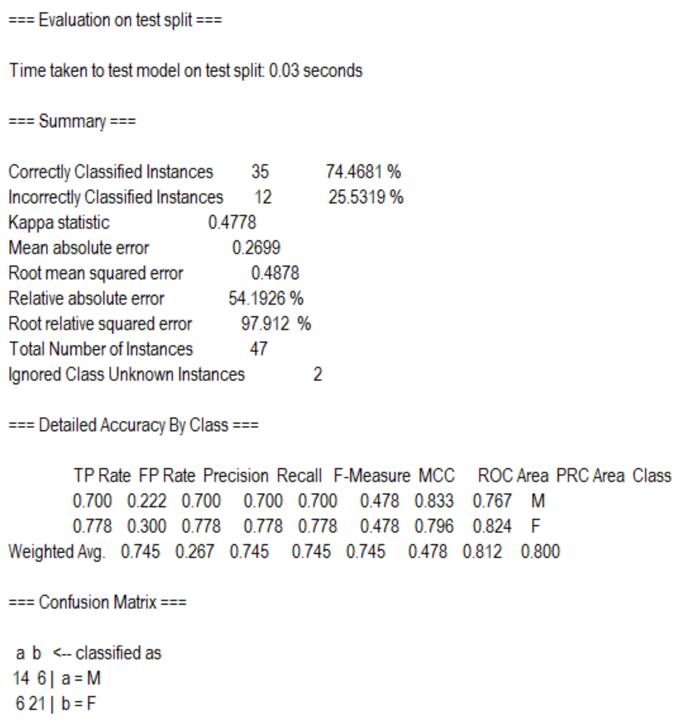
```
=== Evaluation on test split ===

Time taken to test model on test split: 0.19 seconds

=== Summary ===

Correctly Classified Instances        43          91.4894 %
Incorrectly Classified Instances       4           8.5106 %
Kappa statistic                    0.8259
Mean absolute error                   0.1049
Root mean squared error                  0.281
Relative absolute error             21.0526 %
Root relative squared error          56.3984 %
Total Number of Instances           47
Ignored Class Unknown Instances              2

=== Detailed Accuracy By Class ===

           TP Rate  FP Rate  Precision  Recall   F-Measure  MCC     ROC Area  PRC Area  Class
           0.900    0.074    0.900      0.900    0.900      0.826   0.945     0.900     M
           0.926    0.100    0.926      0.926    0.926      0.826   0.934     0.954     F
Weighted Avg.  0.915    0.089    0.915      0.915    0.915      0.826   0.939     0.931

=== Confusion Matrix ===

  a  b   <-- classified as
 18  2 |  a = M
  2 25 |  b = F
```

Figure 16Grey-scaleFinalResult



```
=== Run information ===

Scheme:      weka.classifiers.functions.MultilayerPerceptron -L 0.1 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
Relation:    FinalEigenFeatureCalculation-weka.filters.unsupervised.attribute.Remove-R434
Instances:    399
Attributes:   433
              [list of attributes omitted]
Test mode:    split 87.719% train, remainder test

=== Classifier model (full training set) ===

Sigmoid Node 0
    Inputs    Weights
    Threshold    -0.020434351780357417
    Node 2    0.6571838247531887
    Node 3    0.5819344505076769
    Node 4    0.2174799145524463
    Node 5    1.305555837950704
    Node 6    -0.31011320219488153
    Node 7    0.38081572834554456
    Node 8    0.20144694083063233
    Node 9    0.34981146745230507
    Node 10    -0.769778673667367
```

Figure 17Eigen-featureFinalResult



```
=== Evaluation on test split ===

Time taken to test model on test split: 0.03 seconds

=== Summary ===

Correctly Classified Instances        35               74.4681 %
Incorrectly Classified Instances      12               25.5319 %
Kappa statistic                        0.4778
Mean absolute error                    0.2699
Root mean squared error                0.4878
Relative absolute error               54.1926 %
Root relative squared error           97.912  %
Total Number of Instances             47
Ignored Class Unknown Instances                 2

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.700    0.222    0.700      0.700    0.700      0.478    0.833     0.767     M
                 0.778    0.300    0.778      0.778    0.778      0.478    0.796     0.824     F
Weighted Avg.    0.745    0.267    0.745      0.745    0.745      0.478    0.812     0.800

=== Confusion Matrix ===

  a  b   <-- classified as
 14  6 |  a = M
  6 21 |  b = F
```

Figure 18Eigen-featuresFinalResult

Code

Pre-processor.py

```
#Before starting change the filename in matlab script
#Also make changes wherever mentioned
#import tkinter as tk
#tkinter to get the image files
#from tkinter import filedialog
from Tkinter import *
from tkFileDialog import *
from matlab.engine import * #matlab import
import openpyxl
#Getting the file path
print("Enter the Image file to be worked on")

root = Tk()
root.withdraw()
file_path = askopenfilename()
print("the file selected is ",file_path)

#Using matlab functions on the file path for resizing to
#the size required that is 30X30
```

```
eng=start_matlab()
eng.preprocessormatlabcode(nargout=0)
#Now converting the 30X30 matrix to 1X900
excel_document = openpyxl.load_workbook('testdata-ko.xlsx')
print (type(excel_document))
sheet = excel_document.get_sheet_by_name('Sheet1')
multiple_cells = sheet['A1':'AD30']
Value= ' '
for row in multiple_cells:
    for cell in row:
        actual=str(cell.value)
        Value=Value+actual+' '
RetrievedValues=Value.split(' ');
print(RetrievedValues)
excel_document=openpyxl.load_workbook('testdata-1M.xlsx')
sheet = excel_document.get_sheet_by_name('Sheet1')
multiple_cells = sheet['A400':'AHP400']                #Here
k=1
for row in multiple_cells:
    for cell in row:
        cell.value=int(RetrievedValues[k])
        k=k+1
#observationgot=input('Training image no. is m or f')
#sheet[AHQ3].value=observationgot                        #Here
excel_document.save(filename='testdata-1M.xlsx')
print("All done")
```

pcapart-2.py

```
#To get the eigenvectors and eigenvalues from the training data
#Make changes as mentioned
import numpy as np
import openpyxl
from matlab.engine import *
import csv

#To find mean of all the values
'''
excel_document=openpyxl.load_workbook('testdata-1M.xlsx')#Make change here
sheet = excel_document.get_sheet_by_name('Sheet1')
```

```
multiple_cells = sheet['A1':'AHP1']
Value= ' '
for row in multiple_cells:
    for cell in row:
        Value=Value+cell.column+' '
RetrievedColumnValues=Value.split(' ')
multiple_cells=sheet['A404':'AHP404']#Make change here
k=1
for row in multiple_cells:
    for cell in row:
        FValue=str(RetrievedColumnValues[k])
        Fvalue1=FValue+'1'
        Fvalue2=FValue+'399' #Make change here
        cell.value='=AVERAGE('+Fvalue1+':'+Fvalue2+')'
        k=k+1
excel_document.save(filename='testdata-1M.xlsx')#Make change here
print('Done.Now manually paste ')
'''
#Manually paste values after this from original excel sheet to new1 excel
#sheet

#To find the difference between mean and the other values

'''
excel_document=openpyxl.load_workbook('new1.xlsx')#Make change here
sheet = excel_document.get_sheet_by_name('Sheet1')
Value=' '
multiple_cells=sheet['A404':'AHP404']
for row in multiple_cells:
    for cell in row:
        Value=Value+str(cell.value)+' '
RetrievedValues=Value.split(' ')
RetrievedValues.pop()
RetrievedValues.pop(0)
print(RetrievedValues)
k=0
Val=' '

multiple_cells=sheet["A1:AHP399"] #Make changes here
```

```
for row in multiple_cells:
    for cell in row:
        x=float(RetrievedValues[k])-cell.value
        Val=Val+str(x)+' '
        k=k+1
    FinalVal=Val.split(' ')
    row=cell.row
    newrow=row+410 #Make change here
    newinitial='A'+str(newrow)
    newfinal='AHP'+str(newrow)
    multiple_cells=sheet[newinitial:newfinal]
    l=1
    for row in multiple_cells:
        for cell in row:
            cell.value=FinalVal[l]
            l=l+1
    k=0
    Val=' '
excel_document.save(filename='new1.xlsx')
'''
f1=open('EigenProjects.csv','ab')
writer=csv.writer(f1)

#listofzeroes=[0]*30
#maximum=listofzeroes
maximum=np.zeros((30L,))

listofzeroes=np.zeros((30L,))

#To convert the 1X900 to 30X30
eng=start_matlab()
excel_document=openpyxl.load_workbook('new1.xlsx')#Make change here
sheet = excel_document.get_sheet_by_name('Sheet1')
multiple_cells=sheet['A411':'AHP809'] #Make changes here

for row in multiple_cells:
    k=0
    Value=' '
    for cell in row:
        Value=Value+str(cell.value)+' '
        k=k+1
        if k==30:
            Value=Value+';'+' '
            k=0
    ReshapedValues=Value.split(' ')
    ReshapedValues.pop()
    ReshapedValues.pop(0)
    for i in range(len(ReshapedValues) - 1, -1, -1):
    # iterate over reversed indices's
        if ReshapedValues[i] == ';':
            del ReshapedValues[i]
    #print(ReshapedValues)
    #x = np.array(ReshapedValues, dtype='|S4')
    #ReshapedValues = x.astype(np.float)
    a = np.array(ReshapedValues)
    a[a == ''] = 0.0
    ReshapedValues = a.astype(np.float)
    print(ReshapedValues)
    #a=np.reshape(ReshapedValues,(30,30))
    a=np.array(ReshapedValues)
    a.resize((30,30))
    #print(a.shape)
    #data=np.array(a).astype(np.float)
    data=np.matrix(a).astype(np.float)
    b=np.cov(data)
    e,w=np.linalg.eig(b)
    #print(e.shape)
    #b=np.array(e)
    #b.resize((30,1))
    #print(b.shape)
    #if np.all(np.asarray(e) > np.asarray(listofzeroes)):
    if np.all(np.matrix(e) > np.matrix(listofzeroes)):
        maximum=e
        vector=w
#print('Finallllly!!!')
#print(maximum)
#print(w.shape)
#c=np.array(maximum)
#c.resize((30,1))
#print(c.shape)
```

```
count=0
multiple_cells=sheet['A411':'AHP809'] #Make changes here

for row in multiple_cells:
    k=0
    Value=' '
    for cell in row:
        Value=Value+str(cell.value)+' '
        k=k+1
        if k==30:
            Value=Value+';'+' '
            k=0
    count=count+1

    ReshapedValues=Value.split(' ')
    ReshapedValues.pop()
    ReshapedValues.pop(0)
    for i in range(len(ReshapedValues) - 1, -1, -1):
    # iterate over reversed indices's
        if ReshapedValues[i] == ';':
            del ReshapedValues[i]
    #print(ReshapedValues)
    #x = np.array(ReshapedValues, dtype='|S4')
    #ReshapedValues = x.astype(np.float)
    a = np.array(ReshapedValues)
    a[a == ''] = 0.0
    ReshapedValues = a.astype(np.float)
    #a=np.reshape(ReshapedValues,(30,30))
    a=np.array(ReshapedValues)
    a.resize((30,30))
    #print(a.shape)
    #final=(np.array(a).astype(np.float))*(np.array(maximum).astype(np.float))
    final=np.dot(a,w)
    #print(final.shape)
    #print("Say Hi to new feature vectors")
    #print(final)

    b=np.array(final)
    b.resize((1,900))
    for values in b:
        writer.writerow(values)
f1.close()
```

Genetic.py

```
from __future__ import division
import openpyxl
import random
import csv
import numpy as np
from openpyxl import Workbook
from openpyxl.styles import Color, PatternFill, Font, Border
from openpyxl.styles import colors
from openpyxl.cell import Cell

#Saving genetic data
'''
excel_document=openpyxl.load_workbook('geneticdata.xlsx')
sheet = excel_document.get_sheet_by_name('Sheet1')
multiple_cells = sheet['A1':'AHP399']
for row in multiple_cells:
    for cell in row:
        cell.value=random.randint(0,1)
excel_document.save(filename='geneticdata.xlsx')
print('Genetic Data for the images have been saved')

#Multiplying the genetic data with the eigen projects to get the eigen features for SVM training

#Multiply two xlsx files not csv files becoz openpyxl does not support it.
#change name of sheet from finalEigenProjects to sheet1

excel_document=openpyxl.load_workbook('geneticdata.xlsx')
sheet = excel_document.get_sheet_by_name('Sheet1')
multiple_cells = sheet['A1':'AHP399']

excel_document=openpyxl.load_workbook('AccordingToFeatures3.xlsx')#InitialDif
sheet1 = excel_document.get_sheet_by_name('Sheet2')
sheet2 = excel_document.get_sheet_by_name('Sheet3')
sheet3=excel_document.get_sheet_by_name('Sheet4')
```

```
k=0
value1=' '
multiple_cells=sheet1['A1':'PP399']#Original:AHP399
for row in multiple_cells:
    for cell in row:
        value1=value1+str(cell.value)+' '
        k=k+1

multiple_cellss=sheet2['A1':'PP399']#Original:AHP399
k=0
value2=' '
for row in multiple_cellss:
    for cell in row:
        value2=value2+str(cell.value)+' '
        k=k+1

RetrievedValues1=value1.split(' ')
RetrievedValues1.pop()
RetrievedValues1.pop(0)
a = np.array(RetrievedValues1)
a[a == ''] = 0.0
ReshapedValues1 = a.astype(np.float)

RetrievedValues2=value2.split(' ')
RetrievedValues2.pop()
RetrievedValues2.pop(0)
a = np.array(RetrievedValues2)
a[a == ''] = 0.0
ReshapedValues2 = a.astype(np.float)

k=0
multiple_cells=sheet3['A1':'PP399']#Original:AHP399
for row in multiple_cells:
    for cell in row:
        cell.value=ReshapedValues1[k]*ReshapedValues2[k]
        k=k+1
print('done!')
excel_document.save(filename='AccordingToFeatures3.xlsx')

'''
```

```
#SVM Training done !
#Now move on to genetic algorithm

#First iteration fitness values <0.22 selected



a=0.9
b=0.1
SG=900
k=0
AR=0.78
FG=0
excel_document=openpyxl.load_workbook('AccordingToFeatures3.xlsx')
sheet = excel_document.get_sheet_by_name('Sheet5')
multiple_cells = sheet['A1':'OI432']

l=0
#for iteration in range(1,5):

#Removing ones with low fitness function
count=0
fitness=[]
wipeout=[]
for rows in multiple_cells:
    for cell in rows:
        if type(cell.value)=='Nonetype':
            continue
        if cell.value==1:
            FG=FG+1
    fitness.append(a*(1-AR)+b*(FG/SG))
    print(a*(1-AR)+b*(FG/SG))
    FG=0
'''
for index, item in enumerate(fitness):
                        if item<0.219:#0.22
                            count=count+1
                        elif item>0.219:#0.22
                            wipeout.append(index+1)
print(count)
```

```
for values in wipeout:
                        part1='A'+str(values)
                        part2='OI'+str(values)
                        multiple_cells=sheet[part1:part2]
                        for rows in multiple_cells:
                                                for cell in rows:
                                                                        cell.value=None
excel_document.save(filename='AccordingToFeatures.xlsx')

#Crossover operation

#Selecting Parents

a=0.9
b=0.1
SG=900
k=0
AR=0.78
FG=0
fitty=0.0
indices=[]
excel_document=openpyxl.load_workbook('AccordingToFeatures3.xlsx')
sheet = excel_document.get_sheet_by_name('Sheet1')
multiple_cells = sheet['A1':'OI900']
count=0
fitness=[]
wipeout=[]
for rows in multiple_cells:
    count=count+1
    for cell in rows:
            if cell.value==1:
                FG=FG+1
    if FG!=0:
        indices.append(cell.row)
        fitness.append(a*(1-AR)+b*(FG/SG))
    FG=0
print(min(fitness))
print(indices)

k=0
```

```
#columns specify each and every iteration

excel_document=openpyxl.load_workbook('FittingFunctionValues.xlsx')
sheet = excel_document.get_sheet_by_name('Sheet1')
multiple_cells=sheet['A1':'A423']
for values in indices:

    sheet['A'+str(values)]=fitness[k]
    k=k+1

excel_document.save(filename='FittingFunctionValues.xlsx')

#Mark the parents in the FittingFunctionValues excel file

excel_document=openpyxl.load_workbook('FittingFunctionValues.xlsx')
sheet = excel_document.get_sheet_by_name('Sheet1')
multiple_cells = sheet['A1':'A900']

Crossoverparents=[]
flag=[0]*20
for rows in multiple_cells:
    for cell in rows:
        if cell.value < 0.218 and cell.value!=None:
            Crossoverparents.append(cell.row)
            cell.font = Font(color=colors.RED, italic=True)
excel_document.save(filename='FittingFunctionValues.xlsx')

print(Crossoverparents)

excel_document=openpyxl.load_workbook('AccordingToFeatures3.xlsx')
k=0
sheet = excel_document.get_sheet_by_name('Sheet1')
l=0
index=0
count=902
while index!=18:
        values=' '
```

```
value12=' '
firstparent='A'+str(Crossoverparents[index])
half1='HO'+str(Crossoverparents[index])
multiple_cells1=sheet[firstparent:half1]
for rows in multiple_cells1:
    for cell in rows:
        value12=value12+str(cell.value)+' '
ReshapedValues=value12.split(' ')
ReshapedValues.pop()
ReshapedValues.pop(0)
firstparent='A'+str(count)
first_half='HO'+str(count)
multiple_cells_Final=sheet[firstparent:first_half]
k=0
for rows in multiple_cells_Final:
    for cell in rows:
        cell.value=int(ReshapedValues[k])
        k=k+1
k=0

secondparent='OI'+str(Crossoverparents[index+1])
half2='HP'+str(Crossoverparents[index+1])
multiple_cells2=sheet[half2:secondparent]
for rows in multiple_cells2:
    for cell in rows:
        values=values+str(cell.value)+' '

ReshapedValues=values.split(' ')
ReshapedValues.pop()
ReshapedValues.pop(0)
print(ReshapedValues)
secondparent='HP'+str(count)
second_half='OI'+str(count)
multiple_cells_Final=sheet[secondparent:second_half]
k=0
for rows in multiple_cells_Final:
    for cell in rows:
        cell.value=int(ReshapedValues[k])
        k=k+1
k=0
```

```
        count=count+1
        index=index+2
excel_document.save(filename='AccordingToFeatures3.xlsx')
#Mutation Operation

excel_document=openpyxl.load_workbook('AccordingToFeatures3.xlsx')
sheet = excel_document.get_sheet_by_name('Sheet1')
k=0
count2=902
index=0
count=0
while index!=18:
    startingindex='A'+str(count2)
    endingindex='OI'+ str(count2)
    multiple_cells=sheet[startingindex:endingindex]
    for rows in multiple_cells:
        for cell in rows:
            if cell.value==1 and count==0:
                print(cell.column)
                cell.value=0
                count=1
    index=index+2
    count=0
    count2=count2+1
excel_document.save(filename='AccordingToFeatures3.xlsx')


#Now delete the unwanted rows in genetic data and eigen projects
#multiply both and train through the back propogation neural network

parameters=[]
excel_document=openpyxl.load_workbook('AccordingToFeatures3.xlsx')#FinalEigenFeature
Calculation
sheet = excel_document.get_sheet_by_name('Sheet2')
sheet2 = excel_document.get_sheet_by_name('Sheet3')
multiple_cells=sheet['A1':'AHP399']
count=0
for rows in multiple_cells:
    for cell in rows:
        if cell.value==None:
            parameters.append(cell.column)
    break
print(parameters)

for letter in parameters:
    start=letter+str(1)
    end=letter+str(399)
    multiple_cells=sheet2[start:end]
    for rows in multiple_cells:
        for cell in rows:
            cell.value=None
excel_document.save(filename='AccordingToFeatures3.xlsx')#FinalEigenFeatureCalculation
'''
```

PROJECT DETAILS

Student Details			
Student Name	**Sahil Ajmera**		
Register Number	130905324	Section / Roll No	B/24
Email Address	sahilajmera18@gmail.com	Phone No (M)	9663575836
Project Details			
Project Title	**Real-time Gender Recognition based on selection of Eigen-features from Facial Images**		
Project Duration	4 months	Date of reporting	10th January,2017
Organization Details			
Organization Name	**Manipal Institute Of Technology**		
Full postal address with pin code	Manipal Institute Of Technology,Manipal Pin:576104		
Internal Guide Details			
Faculty Name	**Dr. N V Subba Reddy**		
Full contact address with pin code	Dept of Computer Science & Engg, Manipal Institute of Technology, Manipal – 576 104 (Karnataka State), INDIA		
Email address	nvs,reddy@manipal.edu		