2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA) | 978-1-7281-8470-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICMLA51294.2020.00170

# Evaluation of Local Binary Pattern Algorithm for User Authentication with Face Biometric

Tony Gwyn
Department of Computer Science
Winston-Salem State University
Winston-Salem, NC USA
tgwyn482@rams.wssu.edu

Mustafa Atay
Department of Computer Science
Winston-Salem State University
Winston-Salem, NC USA
ataymu@wssu.edu

Roy Kaushik
Department of Computer Science
North Carolina A&T State University
Greensboro, NC USA
kroy@ncat.edu

Albert Esterline
Department of Computer Science
North Carolina A&T State University
Greensboro, NC USA
esterlin@ncat.edu

***Abstract*— In the ever-changing world of computer security and user authentication, the username/password standard is becoming increasingly outdated. Using the same username and password across multiple accounts and websites leaves a user open to vulnerabilities, and the need to remember multiple usernames and passwords feels very unnecessary in the current digital age. Authentication methods of the future need to be reliable and fast, while maintaining the ability to provide secure access. Augmenting traditional username-password standard with face biometric is proposed in the literature to enhance the user authentication. However, this technique still needs an extensive evaluation study to show how reliable and effective it will be under different settings.**

***Local Binary Pattern (LBP) is a discrete yet powerful texture classification scheme, which works particularly well with image classification for facial recognition. The system proposed here strives to examine and test various LBP configurations to determine their image classification accuracy. The most favorable configurations of LBP should be examined as a potential way to augment the current username and password standard by increasing their security with facial biometrics.***

***Keywords—Local Binary Pattern, authentication, biometrics, face biometrics, facial recognition, classification methods, WebID.***

## I. Introduction

Biometrics are measurements of human characteristics that can be used for authentication purposes. Biometrics are nearly impossible to spoof, copy, or duplicate perfectly; this makes them an ideal candidate for user authentication. Facial biometrics in particular have shown great promise for authentication purposes, due to noticeable differences in user faces that can be discerned by systems [7, 12].

While single sign-on WebIDs have become the first step to providing such services [1], their security as a standalone package is lacking [2, 3]. An intruder who attains access to a user's computer or has the unique certificate of that user can quickly compromise a single sign-on WebID system with no additional security [4, 5]. WebIDs enhanced with biometric authentication are more secure, but this security can come with increased computational costs [11]. Various schemas such as Eigenfaces [13, 14] and Fisherfaces [15, 16] work well at classifying images, yet lack the ability to operate at a pace needed for real-time biometric authentication [6], which can be a problem for certain authentication systems.

Local Binary Pattern (LBP) is a discrete yet powerful texture classification schema, adept at classifying and comparing grey-scale images [7]. While LBP has been thoroughly studied, to our best knowledge, an extensive evaluation of LBP for facial recognition, and specifically for periocular recognition, lacks behind significantly. The problem at hand is intricate and complex. The following is a full description of the issues, as well as the challenges faced in meeting those problems:

1. The username and password standard is quickly becoming outdated as advances in technology continue. While some attempts to shore up the system have been made, those attempts have seen limited success. Further research into other methods should be investigated.

2. Humans loathe change to the norm; there is no exception when it comes to computer security. Rather than proposing an entirely new way to protect users from unauthorized access, implementing a system that enhances the username and password standard would be preferable.

3. Finally, the proposed enhancement to the username and password standard should be easy to implement, user-friendly, as well as lightweight, computationally speaking. If we are to change the way that authentication works with a countless number of systems, these requirements will be a necessity to allow for easy adoption by all.

The system that we propose in this study strives to solve all the problems put forth by the previous questions. A plethora of research into various authentication methods has been done prior to completing this work. Although several different methods were examined for the purpose of enhancing the current username and password standard, a full examination of the research material led us to determine that LBP was best suited to our needs.

The main contributions of this study include the following:

978-1-7281-8470-8/20/$31.00 ©2020 IEEE
DOI 10.1109/ICMLA51294.2020.00170

We developed a user authentication system with face biometric to extensively evaluate the accuracy of LBP algorithm with a rich set of various settings.

1) We create a program to test LBP for recognizing facial images using several parameters. These parameters include LBP version, classifier, neighborhood size, radius size, training image set size, and confidence level. The user sets each of those parameters, and runs the program, after which an accuracy of that configuration is given.

2) We enhance the program by providing with various different testing modes for the user to choose from, which allow the user to test the efficacy of various parameters and plot its graph. Those testing modes, are not limited to but, include radius and neighborhood size versus LBP accuracy, image set size versus LBP accuracy, classification algorithms or LBP variants versus LBP accuracy.

## II. Related Work

Although a number of studies on image classification schemes exist in the literature [6, 7, 8, 9, 12, 13, 15, 16, 17, 18, 19, 20], each has its own drawbacks and limitations when compared to our proposed study. While several of these documented studies are expansive in regards to their amount of Local Binary Pattern variants [8, 9, 12, 16, 17], they fall behind the sheer number of options and configurations we put forth in our study. We include not only several LBP variants for testing, but also allow for the adjustment of neighborhood and radius size, as well as the ability to choose between six different classification methods.

Of the LBP studies explored by the literature, to our knowledge none have a mechanism for effectively dealing with unknown images. Our program has a built-in confidence measurement and algorithm for handling both images that are known and labelled, as well as those that are unknown. These mechanisms increase the overall difficulty of our image classification, but also create results that are more robust.

The considerable amount of LBP studies have not dealt specifically with biometric authentication with respect to facial recognition. Our program makes use of a facial image database for the express purpose of evaluating LBP as a biometric authentication method. Other image classification methods are also well studied in the literature [6, 13, 14, 15, 16, 19, 20]. However, these methods are unable to keep pace with LBP, in regards to computational complexity or program run time. LBP deals with the issue of feature extraction and image classification in novel ways that help to both reduce the time the program needs to run to completion, as well as the amount of computational power needed to run said program.

## III. Preliminaries

As mentioned previously, our study makes use of five different LBP variants, as well as six different classification methods. When added to the wide array of neighborhood and radius values that can be adjusted, as well as the variable image training set size and confidence level values, our study is able to produce an impressive number of configurations for the purpose of evaluating LBP as a biometric authentication technique.

### A. LBP Algorithms

There are easily dozens, if not hundreds, of LBP variations explored in the literature [8, 9]. The variants that are used in our proposed study are default, extended, uniform, non-rotationally invariant (NRI), and variance. The differences in each of these LBP variants are as follows [21]:

- *Default* – As its name implies, the default variant of LBP. This version is gray scale invariant; however, it is not rotation invariant.

- *Extended* – This variant of LBP is an extension of the default variant. This version is both gray scale and rotation invariant.

- *Uniform* – This variant of LBP is a version of the default variant, which has increased rotation invariance when considering uniform patterns. It also has an improved quantization when dealing with the angular space. It is both gray scale and rotation invariant [22].

- *Non-Rotationally Invariant (NRI)* – This variant of LBP is a version of the uniform variant previously discussed. As its name implies, it is non rotation-invariant; however, it is gray scale invariant [23].

- *Variance* – The final variant of LBP that we use in our proposed study is called variance. It takes variance measures of the contrast of local image texture. It is rotation invariant, but not gray scale invariant.

The major differences for each of these versions of LBP lie in how they deal with rotation and gray scale. Rotation invariant variations of LBP are unaffected by changes in image position, even when the image is rotated in ways that are different to the original image. However, rotation variant versions of LBP that we used in our study, such as Default or NRI, will give different values when an image is rotated, versus when it is not. This can potentially have a profound effect on the overall accuracy rate of the different LBP versions.

Just as with rotation, gray scale invariant variations of LBP are unaffected by differences in lighting or other illumination conditions. Gray scale variant LBP versions that we used in our study, such as Variance, will see different values depending on the illumination conditions of an image, such as when in a low light setting or when an image is overexposed.

*B. Classification Methods*

Classification algorithms play an integral role with the LBP algorithm, helping to categorize the images and determine their relationship to each other. Our proposed study uses a total of six different traditional classification methods that are found in the literature to enhance the LBP function. They include Support Vector Classifier (SVC), Linear Discriminant Analysis (LDA), K-Nearest Neighbors (KNN), Decision Trees (DT), Naïve-Bayes (NB), and Logistic Regression (LR).

- *Support Vector Classifier (SVC)* – Support Vector Classifier is an image classifier that is designed for binary problems, and can also be extended to handle multi-class problems. SVC ensures high generalization by mapping inputs non-linearly to high-dimensional feature spaces, which allows linear decision surfaces to be constructed

  Originally, SVC was to have data separable by a hyperplane without error. However, later versions include a 'soft-margin' which allows for a permittable minimal subset of error [27]. In the literature, SVC performed admirably when compared to both Decision Tree (DT) and Histogram of Gradients (HoG) [27]. In our testing, SVC was difficult to work with, due to the sheer number of parameters that are available in its algorithm. However, after the correct parameters were found, SVC gave impressive accuracies among multiple configurations.

- *Linear Discriminant Analysis (LDA)* – Linear Discriminant Analysis (LDA), as well as its counterpart, Principle Component Analysis (PCA), are very well-known classification techniques. While PCA is an unsupervised technique, and therefore not of use to us in our study, LDA is supervised.

  LDA is both a dimensionality reduction technique, as well as a linear classifier, but for our purposes we focus on the latter. Linear Discriminant Analysis focuses on projecting the data to maximize class separability. This works well for our study, due to the previously mentioned fact that our dataset has a normal distribution.

  In the research, LDA is often used for dimensionality reduction. However, many of the papers where LDA was used for image classification did see some rather impressive results, especially with larger sizes of datasets [28].

- *K-Nearest Neighbors (KNN)* – K-nearest neighbors (KNN) is a common classification method used both for data mining, as well as for image classification. While KNN can be used for both classification and regression, we focus on KNN classification for our study.

  In KNN classification, the output is a class membership. An object is classified by a plurality vote of its nearest neighbors. The object is assigned to the class most common among those $k$ nearest neighbors. $k$ is a positive integer, and typically small; through the research, we have found that setting k = 3 or k = 5 both give good results.

  We found many examples of KNN classification in the literature. Most studies determined that KNN, when paired with a reasonable $k$ value, produced good accuracy results [17]. As for our testing, KNN performed worse than DT, LDA, and SVC.

- *Decision Trees (DT)* – Decision Trees are classifiers that are represented by a flowchart-like structure. Decision Trees are unlike Support Vector Classifiers and neural networks, as they do not make statistical assumptions concerning the inputs, nor do they scale the data.

  DT models have a structure that is similar to a tree, where data is broken down into smaller subsets at each branch. In the research, Decision Trees had a reasonably respectable success rate, falling just short of the accuracy attained by SVC [27].

- *Naïve-Bayes (NB)* – There are a number of variations of Naïve Bayes available in the literature. We explored several different variations throughout the testing of our proposed study. In the end, the one that we decided to work with was the Gaussian version of Naïve Bayes.

  Our labelled classes were equally weighted and given a normal distribution, which made Gaussian Naïve Bayes an arguably better option over Multinomial or Bernoulli Naïve Bayes [25]. In the literature, Naïve Bayes performs extremely well for the purpose of image classification achieving 100 percent accuracy on training images, with 81.25 percent accuracy on training images [26].

- *Logistic Regression (LR)* – Logistic Regression is used to model the probability of a certain class or classes existing. In our study, this refers to the probability that a test image is in the same class as a training image (man_1, man_2, woman_1, etc.). In the literature, Logistic Regression has been shown to have impressive accuracy rates, with both training and testing images. This is only made more impressive due to the fact that the program used reduced image sizes when making these comparisons, in an effort to cut down on computational space and time [24].

In our tests, Logistic Regression did surprisingly well when using a small image set size mixed with a lower neighborhood and radius size. However, as the image set, neighborhood, and radius size increased, we saw smaller gains in accuracy than with other classification methods. As with Naïve Bayes, future work with LBP and biometric authentication could involve a more in-depth and thorough investigation of Logistic Regression.

## IV. Proposed Research Study

Our proposed study on evaluating LBP for the purpose of biometric authentication has two distinct processes. The first process is made specifically to take training images, and to extract the features of those images with the LBP variant and parameters that were configured. Those features are placed into a database, to be compared against the testing images later.


Figure 1. Examples Individuals from the FERET Database

We then take testing images, and extract their features with the same LBP configuration used by the training process. These features are then compared and contrasted against the features database, using the parameters established by the proposed study. This gives us the final step in the proposed study; an overall accuracy from image recognition and classification.

### A. System Architecture

Our system architecture starts with a database of images which we use to examine the accuracy of various LBP configurations. After the image training set size is decided, the proposed study first begins its run through the training data, to start extracting the features that we attempt to match against the testing dataset. The specific LBP variant is selected, and then LBP parameters are established. This configuration will be used for both the training images, and later on for the testing images.

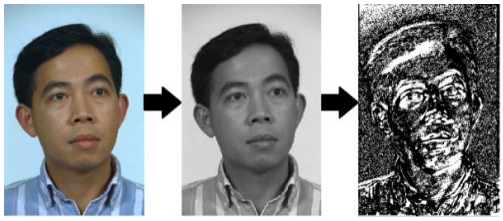

Figure 2. LBP Image Conversion

After LBP has been configured, it is then time to extract the various features from the training images. Once the features have been extracted, they are then placed in a feature database. Once we have similar feature data from the testing images, we do a comparison of those features against the ones we have placed in the feature database.

After the training pass is finished, the program moves on to the testing set of images. As mentioned, we use the exact same LBP variant and configuration for the training images and the testing images so that our results are valid. After LBP extracts the features of the test images, they are compared against the features database, with a selected classification method, as well as a similarity function. The algorithm then compares and contrasts the features taken from the testing images against those that are in the features database; this is where we attain the specific accuracy from image recognition and classification.

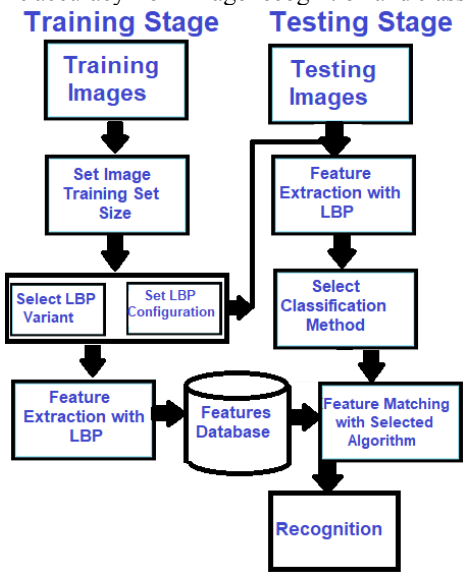


Figure 3. Overview of the System Architecture

### B. Proposed Algorithms

Figure 4 shows our algorithm for evaluating LBP for biometric authentication. First, we input a training set of *n* individuals, each with *m* images. Secondly, we input a testing set of *p* individuals, each with *q* images. In the next step, we establish if we are making a single run of our program, or if we are using the graphing function of our program. If it is a single run, then we set x and y equal to one, as there will only be a single configuration of the program to test. However, if the selected program option is the graphing option, then we need to set *x* to

the first independent variable, and *y* to the second independent variable. We will be running an *x* * *y* number of configurations.

```
01 Algorithm EvaluateLBP
02 Input: TrainingImageSet[n][m], TestingImageSet[p][q], Configurations[x][y]
03 Output: Accuracy[x][y]
04 Begin
05 Let TrainingImageSet[n][m] be m images of n individuals
06 Let TestingImageSet[p][q] be q images of p individuals
07 If Single_Run is selected then
08    Let x=1 and y=1 //one single configuration
09 Else If Generate_Graph is selected then
10    Let x be ||independent_var1|| and y be ||independent_var2|| //x*y configurations
11 End If
12 For i=1 to x do
13    For j=1 to y do
14          Set Configurations[i][j]
15          Train(TrainingImageSet, Configurations[i][j])
16          Accuracy[i][j]=Recognize(TestingImageSet, Configurations[i][j])
17    End For
18 End For
19 Return Accuracy
20 End
```

Figure 4. EvaluateLBP Algorithm

After *x* and *y* are set, we run the specified number of configurations; one for the single run function, and *x* * *y* configurations for the graph function, respectively. For each configuration, we train the images via the Train() algorithm, and then the accuracy is determined by comparing the testing image set against the training images with the Recognize() algorithm.

In Figure 5, we see our image training algorithm. It takes both the training image set of *n* individuals with *m* specific images, as well as the configurations parameters. The training image set size is set to *s* images of *n* individuals from the configurations. The LBP algorithm variant, radius, as well as the size of the neighborhood are set from the *v*, *r*, and *nh* variables of the Configurations parameters, respectively.

```
01 Algorithm Train
02 Input: TrainingImageSet[n][m], Configurations
03 Output: Features_Database FDB
04 Begin
05 Set TrainingImageSet size to s images of n individuals, where s<=m, from
Configurations
06 Set LBP_Algorithm_Variant to v from Configurations
07 Set Radius to r from Configurations
08 Set Neighborhood Size to nh from Configurations
09 For i=1 to n do
10    For j=1 to s do
11          Extracted_Features=LBP(TrainingImageSet[i][j], v, r, nh)
12          Insert Extracted-Features with label i into Features-Database FDB
13    End For
14 End For
15 Return FDB
16 End
```

Figure 5. Train Algorithm

Each training image has its unique features extracted with LBP, using the specific algorithm variant, radius, and neighborhood size configuration. Those features are then loaded into the features database, which is then later returned as the output of this algorithm. Figure 6 contains our image testing algorithm. Just like before with the image training algorithm, the LBP variant, the radius, and the size of the neighborhood are set from the *v*, *r*, and *nh* variables of the Configurations parameters, respectively. This time, the classification algorithm is also set from the *CA* variable from Configurations.

```
01 Algorithm Recognize
02 Input: TestingImageSet[p][q], Configurations, Features_Database FDB
03 Output: accuracy
04 Begin
05 Set LBP_Algorithm_Variant to v from Configurations
06 Set Radius to r from Configurations
07 Set Neighborhood Size to nh from Configurations
08 Set Classification_Algorithm to CA from Configurations
09 correct=0
10 For i=1 to p do
11    For j=1 to q do
12          Extracted_Features=LBP(TestingImageSet[i][j], v, r, nh)
13          Identity[i][j]=Features-Match(CA, Extracted-Features, FDB)
14          If Identity[i][j]==TestingImageSet[i][j] then correct++ End If
15    End For
16 End For
17 accuracy=correct/(p*q)
18 Return accuracy
19 End
```

Figure 6. Recognize Algorithm

A counter is implemented to keep track of the number of correct predictions. Each testing image has its unique features extracted with LBP, using the specific algorithm variant, radius, and neighborhood size configuration. Next, these features are compared against the features in the features database FDB, and the identity of the image is decided. If this identity value matches the actual value of the test image, then the correct prediction counter is incremented.

Once all of the test images have been completely exhausted, the overall accuracy of the algorithm is calculated by the number of correctly identified images divided by the total number of *p* * *q* images, where p denotes number of individuals and q denotes number images per individual, and this accuracy measure is returned as the output of the program.

### *C. Dataset Used*

We use the Facial Recognition Technology (FERET) database to generate the image datasets that we use in our program. We use a total of 12 individuals (10 known, 2 unknown) with 26 images per individual; 2 images each for testing, and up to 24 images each for training [29].

### *D. System Implementation*

Modules are imported from various python libraries, and contained in the four Python programs. The FERET image database is also linked to the Python programs. The version of the programming language we have used to write the code is Python 3.7, and the IDE is PyCharm Community Edition 2019.3.

Scikit-learn contains the algorithms of all of the classification methods that we use, as well as the different distance metrics. Scikit-image is used in conjunction with lbp.py to handle LBP

variations, as well as parameters. Imutils is used to handle the image path, for both our test images, as well as our training images. Numpy is utilized in conjunction with the probability function to generate the best prediction for each testing image.

OpenCV is harnessed in multiple areas of our program, including loading both our testing images and our training images, as well as for converting all of those images to gray scale. We also make use of cv2 in our testing to handle text generation on our test images, for displaying both the predicted label of each test image, as well as the actual label of each test image. Matplotlib handles the generation of the interactive graphing function in our program. Finally, pysimplegui is a graphical user interface (GUI) package that can create custom windows, while retaining a user-friendly interface. We make use of pysimplegui to create the user GUI, and for handling the process of setting or changing the LBP configurations.

## V. Experimental Study

Our program is comprised of two different specific functions; a single run function, for testing a single LBP configuration, and a multiple run function with graphing capability, for testing multiple LBP configurations, as well as graphing the results for easy comparison. The performance of the authentication system will be evaluated using the formula given in Figure 7. The accuracy of our programs are determined by calculating the number of true positives plus true negatives, divided by the total number of data points. The graphing and multiple run function of the program is intended to make multiple runs for multiple different LBP configurations, and then compare and contrast those configurations via the graph that is generated.

$$ACC = \frac{(TP + TN)}{TOTAL} \qquad Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN} \qquad F1 = 2x\frac{P * R}{P + R}$$

Figure 7. Accuracy Evaluation Metrics

### A. Experimental Setting

The GUI for the multiple run option of our program has the ability to handle multiple configurations of the program, as well as present a graph that compares those configurations. This option has seven parameters to choose from, along with a total of eight different experimental modes, which can be selected by use of a drop-down menu. After the user selects specific parameters to measure, the next segment of the program is divided into four sections. Section 1 contains the constants of the program; similar to the single run option, here you simply choose a value for each parameter available. These will be the control variables for our tests, as they remain at a constant value throughout the duration of each test.

The experimental mode selected in the previous segment of the program is divided into two sections. For the first experimental mode, 'Radius versus Classification', the radius parameter(s) are set in section 2, whereas the classification parameter(s) are set in section 3. The parameter(s) that are selected in section 2 and section 3 will be the independent variables for our testing; they will be changed throughout the duration of our testing, to determine the overall effect that they have on image classification accuracy of LBP, or our dependent variable.

Both section 2 and 3 contain checkboxes for the selection of multiple parameters. This functionality is by design; by allowing for the selection of multiple parameters, the program is instructed to run multiple times, making sure to get the accuracy of each configuration. The final section, section 4, simply contains the graph selection, which at this time is comprised of either a bar graph, or a scatter graph for plotting the results of configurations selected.

### B. Testing the Impact of Neighborhood and Radius Size

Figure 8 shows a graph of the results of various neighborhood and radius configurations that were adjusted in tandem. A neighborhood, radius configuration of (24, 2) with an image set size of 24 using the LDA classification method combined with the Uniform LBP algorithm achieved an image classification accuracy of 83.3%.

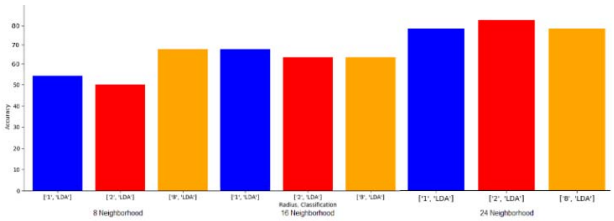

Figure 8. Neighborhood and Radius Size Configurations

| Configuration | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| (8, 1) | 54.2% | 65.0% | 76.5% | 70.28% |
| (8, 2) | 50.0% | 60.0% | 75.0% | 66.67% |
| (8, 8) | 66.7% | 75.0% | 83.3% | 78.93% |
| (16, 1) | 66.7% | 90.0% | 81.8% | 85.70% |
| (16, 2) | 62.5% | 85.0% | 81.0% | 82.95% |
| (16, 8) | 62.5% | 85.0% | 81.0% | 82.95% |
| (24, 1) | 79.2% | 95.0% | 82.6% | 88.37% |
| (24, 2) | 83.3% | 100.0% | 83.3% | 90.89% |
| (24, 8) | 79.2% | 95.0% | 82.6% | 88.37% |

Table 1. Accuracy Measures for Neighborhood and Radius Sizes

In our testing, adjusting only the neighborhood or radius size by itself had little effect on the image classification accuracy. However, here we can see that as those values are increased, that accuracy values gradually increase as well.

### C. Testing the Impact of Image Set Size

Figure 9 shows a graph of configurations with various image set sizes, adjusted in tandem with different classification methods. Both LDA and DT, with an image set size of 24, achieved best in class image classification accuracy of 83.3%.

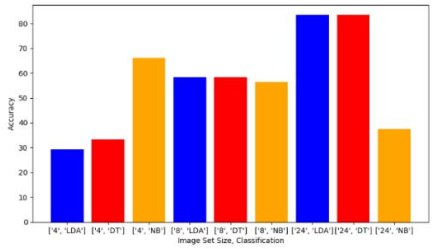


Figure 9. Image Set Size Configurations

| Configuration | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| (4, LDA) | 29.2% | 35.0% | 63.6% | 45.15% |
| (4, DT) | 33.3% | 50.0% | 71.4% | 58.81% |
| (4, NB) | 66.7% | 80.0% | 71.4% | 75.46% |
| (8, LDA) | 58.3% | 70.0% | 77.8% | 73.69% |
| (8, DT) | 58.3% | 80.0% | 80.0% | 80.00% |
| (8, NB) | 54.2% | 85.0% | 77.8% | 81.24% |
| (24, LDA) | 83.3% | 100.0% | 83.3% | 90.89% |
| (24, DT) | 83.3% | 100.0% | 83.3% | 90.89% |
| (24, NB) | 37.5% | 45.0% | 69.2% | 54.54% |

Table 2. Accuracy Measures for Image Set Size Configurations

Naïve Bayes performed well with smaller image amounts. There appears to be a clear correlation between an increase in image set size, and an increase in image classification accuracy for most cases. We were limited to a maximum of 24 training images per individual; we postulate that accuracy can be increased with a larger image training set size.

### *D. Testing the Impact of Classification Methods*

Figure 10 shows a graph of our classification methods used in conjunction with the Uniform LBP variant. While keeping all other parameters fixed, both LDA and DT showed best in class performance, identifying 83.3% of the training images.

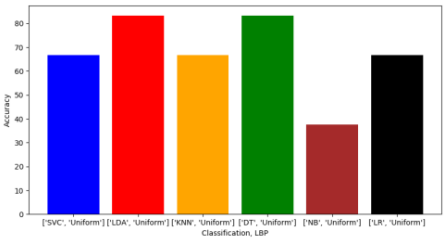


Figure 10. Classification Method Configurations

| Configuration | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| (SVC, Uniform) | 66.7% | 75.0% | 83.3% | 78.93% |
| (LDA, Uniform) | 83.3% | 100.0% | 83.3% | 90.89% |
| (KNN, Uniform) | 66.7% | 80.0% | 80.0% | 80.00% |
| (DT, Uniform) | 83.3% | 90.0% | 81.8% | 85.70% |
| (NB, Uniform) | 37.5% | 45.0% | 69.2% | 54.54% |
| (LR, Uniform) | 66.7% | 75.0% | 83.3% | 78.93% |

Table 3. Accuracy Measures for Classification Configurations

As our results show, Uniform LBP generates good results with most of the classification methods. Naïve Bayes is the sole outlier, with an accuracy level of 37.5%. We postulate that this is due to the low confidence levels that this particular classification method generates.

### *E. Testing the Impact of LBP Variants*

Figure 11 shows a graph of our LBP variants used in tandem with the LDA classification method. With all parameters fixed, the Uniform LBP variant had the best accuracy results, at 83.3%.

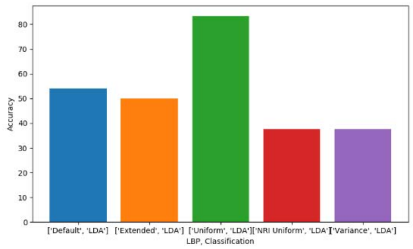


Figure 11. LBP Variant Configurations

| Configuration | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| (Default, LDA) | 54.2% | 65.0% | 76.5% | 70.28% |
| (Extended, LDA) | 50.0% | 60.0% | 75.0% | 66.67% |
| (Uniform, LDA) | 83.3% | 100.0% | 80.0% | 88.89% |
| (NRI Uniform, LDA) | 37.5% | 45.0% | 69.2% | 54.54% |
| (Variance, LDA) | 37.5% | 45.0% | 69.2% | 54.54% |

Table 4. Accuracy Measures for LBP Configurations

As we can see from the results, the Uniform LBP variant was by far the most accurate in most of the configurations we tested. However, in a few instances, the Extended LBP variant performed at or above the accuracy levels that we saw with Uniform LBP. This fact underscores the necessity for testing multiple LBP variants under different circumstances.

## VI. Conclusions

In this paper, we create a program for dealing with a single run of LBP to recognize facial images using several parameters. The user then runs the program, after which an accuracy of that configuration is given. We also create a program for dealing with multiple runs of LBP. After the user sets the parameters, the program is run an amount of times equal to the configurations given. The results are presented in an easy to follow graph, with each configuration on the x-axis, and the accuracy of those configurations on the y-axis. We enhance the proposed approach by providing it multiple different modes for the user to choose from. In our testing, both LDA and DT classification methods achieved an image classification accuracy of 83.3% when used in tandem with Uniform LBP. In future, we will apply different feature extraction methods, including Scale Invariant Feature Transform (SIFT) and Speed up Robust Features (SURF) with the appropriate deep learning models and compare the performances with the traditional texture-based approaches.

## VII. Acknowledgement

This research is funded by NSF Award #1900087. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF. Thanks to the National Institute of Standards and Technology for access to the FERET image database.

## VIII. References

[1] Toby Inkster, Henry Story, and Bruno Harbulot. WebID authentication over TLS. Last retrieved July 2019, https://www.w3.org/2005/Incubator/webid/spec/tls/.

[2] Cory Sabol, Wesley Odd, and Albert Esterline, 2016. Group Access Control using WebID. In SoutheastCon 2016. (Mar. 2016). DOI: https://doi.org/10.1109/SECON.2016.7506672.

[3] Cory Sabol, William Nick, Maya Earl, et.al, 2016. The WebID Protocol Enhanced with Group Access, Biometrics, and Access Policies. In MAICS 2016. (Apr. 22-23 2016), 89-95.

[4] William Nick, Cory Sabol, Joseph Shelton, et al. Federated Protocol for Biometric Authentication and Access Control. Computing Conference. (Jul 2017), 854-862. DOI: https://doi.org/10.1109/SAI.2017.8252195.

[5] William Nick, Cory Sabol, Albert Esterline, 2017. A federated protocol for active authentication. In SoutheastCon 2017. (Mar.2017). DOI: https://doi.org/10.1109/SECON.2017.7925377

[6] Joao P. Hespanha, David J. Kriegman, and Peter N. Belhumeur, 1997. Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. In IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 19, Issue 7 (Jul. 1997), 711-720. DOI: https://doi.org/10.1109/34.598228.

[7] Timo Ahonen, Abdenour Hadid, Matti Pietikainen, 2006. Face Description with Local Binary Patterns: Application to Face Recognition. In IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 28, Issue 13 (Dec. 2006), 2037-2041. DOI: https://doi.org/10.1109/TPAMI.2006.244.

[8] Matti Pietikainen and Guoying Zhao, 2015. Two decades of local binary patterns: A survey. In Advances in Independent Component Analysis and Learning Machines, (2015), 175-210. DOI: https://doi.org/10.1016/B978-0-12-802806-3.00009-9.

[9] Li Liu, Paul Fieguth, Yulan Guo, et. al, 2017. Local binary features for texture classification: Taxonomy and experimental study. In Pattern Recognition, Volume 62 (Feb. 2017), 135-160. DOI: https://doi.org/10.1016/j.patcog.2016.08.032.

[10] Kelvin Salton do Prado, 2017. Face Recognition: Understanding LBPH Algorithm. https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b, Nov. 10 2017.

[11] Nataliya Sukhai, 2004. Access control & biometrics. In Proceedings of the 1st annual conference on Information security curriculum development, (Oct. 2004), 124-127. DOI: https://doi.org/10.1145/1059524.1059552.

[12] K. Kishore Kumar, Movva Pavani. LBP based biometrie identification using the Periocular Region. 2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Oct. 3-5 2017.

[13] Matthew Turk. Over twenty years of eigenfaces. ACM Transactions on Multimedia Computing, Communications, and Applications, (Oct. 2013), Article 45. DOI: https://doi.org/10.1145/2490824.

[14] Salaheddin Alakkari, John James Collins. Eigenfaces for Face Detection: A Novel Study. In Proceedings of the 2013 12th International Conference on Machine Learning and Applications – Volume 01, (Dec. 2013), 309-314. DOI: https://doi.org/10.1109/ICMLA.2013.63.

[15] Peter N. Belhumeur, Joao Hespanha, David J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. In Proceedings of the 4th European Conference on Computer Vision – Volume 1, (Apr. 1996), 45-58.

[16] Banu Wirawan Yohanes, Reva Diaz Airlangga, Iwan Setyawan. Real Time Face Recognition Comparison Using Fisherfaces and Local Binary Pattern. In Proceedings of the 4th International Conference on Science and Technology (ICST), (2018), 1-5. DOI: https://doi.org/10.1109/ICSTC.2018.8528608.

[17] Oscar Garcia-Olalla, Enrique Alegre, Maria Teresa Garcia-Ordas, Laura Fernandez-Robles. Evalutation of LBP Variants Using Several Metrics and kNN Classifiers. In Proceedings of the International Conference on Similarity Search and Applications (SISAP), (Sept. 2013). DOI: https://doi.org/10.1007/978-3-642-41062-8_15.

[18] Karl Ricanek. LBP-based periocular recognition on challenging face datasets. EURASIP Journal on Image and Video Processing, (July 2013). DOI: https://doi.org/10.1186/1687-5281-2013-36.

[19] Rajkumar N. Satare, S. R. Khot. Image matching with SIFT feature. In Proceedings of the 2018 2nd International Conference on Inventive Systems and Control (ICISC), (2018). DOI: https://doi.org/10.1109/ICISC.2018.8399100.

[20] Zhang Huijuan, Hu Qiong. Fast image matching based-on improved SURF algorithm. In 2011 International Conference on Electronics, Communications and Control (ICECC), (2011). DOI: https://doi.org/10.1109/ICECC.2011.6066546.

[21] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, et. al, 2011. Scikit-learn: Machine Learning in Python. In Journal of Machine Learning Research, Vol. 12, No. 85, (2011), 2825-2830.

[22] Timo Ahonen, Abdenour Hadid, Matti Pietikainen, 2004. Face Recognition with Local Binary Patterns. In Proceedings of the 8th European Conference on Computer Vision, (May 2004). DOI: https://doi.org/10.1007/978-3-540-24670-1_36.

[23] Timo Ojala, Matti Pietikainen, Topi Maenpaa, 2002. Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns. In IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24, No. 7, (July 2002), 971-987. DOI: https://doi.org/10.1109/TPAMI.2002.1017623.

[24] Vanlalhruaia, Yumnam Kirani Singh, N. Debachandra Singh, 2017. Binary face image recognition using logistic regression and neural network. In 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), (2017), 3883-3888. DOI: https://doi.org/10.1109/ICECDS.2017.8390191.

[25] Hajer Kamel, Dhahir Abdulah, Jamal M. Al-Tuwaijari, 2019. Cancer Classification Using Gaussian Naïve Bayes Algorithm. In 2019 International Engineering Conference (IEC) (2019), 165-170. DOI: https://doi.org/10.1109/IEC47844.2019.8950650.

[26] Hein Tun Zaw, Noppadol Maneerat, Khin Yadanar Win, 2019. Brain tumor detection based on Naïve Bayes Classification. In Proceedings of the 2019 5th International Conference on Engineering, Applied Sciences and Technology (ICEAST), (2019), 1-4. DOI: https://doi.org/10.1109/ICEAST.2019.8802562.

[27] Taha J. Alhindi, Shivam Kalra, Ka Hin Ng, et. al., 2018. Comparing LBP, HOG, and Deep Features for Classification of Histopathology Images. Arxiv.org, (May 2018). https://arxiv.org/pdf/1805.06837.pdf

[28] Shaoning Pang, S. Ozawa, N. Kasabov. Incremental linear discriminant analysis for classification of data streams. In IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), (Oct. 2005), vol. 35, no. 5, 905-914. DOI: https://doi.org/10.1109/TSMCB.2005.847744.

[29] Flanagan, Patricia, 2011. Face Recognition Technology (FERET), January 2011. https://www.nist.gov/programs-projects/face-recognition-technology-feret.