A Comparison of Iris Scanning Algorithms: Classical Vs. Machine Learning

Tory Farmer

Advisor: Mladen Wickerhauser

March 23 2020

Contents

1 Abstract

Humans have long used biometric identification methods to identify others. One recent method is scanning the iris, which is known to have unique patterns similar to the fingerprint. Unlike the fingerprint, irises can be examined at a distance without needing to contact a surface. Coinciding with the growing popularity of iris scanning is the growing popularity and efficacy of machine learning. As a result, many researchers have used machine learning models to determine whether two iris photographs are from the same individual or not. However, it is not well known whether these novel machine learning algorithms outperform their classical counterparts in speed, accuracy, or performance given low-resolution data.

To compare the two approaches, a classical algorithm and machine learning algorithm for iris scan matching are implemented in Matlab and Python. Both algorithms use the original CASIA Iris V3 dataset and a 5x blurred version of it. A confusion matrix is produced for each algorithm and each data set. Each algorithm is then given one half of the CASIA Iris V3 dataset as a "population" and a new iris scan. The algorithms are asked to determine who, if anyone, in the population matches the new scan. The classical algorithm outperforms the machine learning algorithm in performance in both data sets and both tests but has a longer runtime. Finally, the CASIA dataset is further blurred by factors beyond 5x, and a factor of 17.5x is identified as the machine learning-superior blur factor. The machine learning algorithm outperforms the classical algorithm at blur factors greater than 17.5x and underperforms otherwise. Despite the machine learning algorithm being generally faster than the classical version, I conclude that the classical implementation's accuracy is so far superior that it is the obvious choice for any real-world implementation.

2 Review of Literature

2.1 Classical Iris Scanning

2.1.1 Motivation

As long as humans have existed, we have used parts of our bodies to identify one another. As mentioned in [1], humans have used biometric identification in some form for over 2000 years. At first, people used easily identifiable features such as one's face or hand, but these were not detailed enough to match individuals in a large population with reasonable confidence. In the 1600's, fingerprints were shown to be unique and, in the 1800's, detectives and prosecutors began using fingerprints in criminal investigations. Irises, like fingerprints, are unique among individuals and thus can be used for identification. However, unlike fingerprints, irises can be identified at a distance without contacting a surface with potentially unaware participants. This substantially increases the applicability of the iris scan and has encouraged the development of methods to match irises as accurately as possible.

2.1.2 The Original Algorithm

In my quest to learn about the origins of iris scanning, I came across "High Confidence Visual Recognition of Persons by a Test of Statistical Independence" by John Daugman [2]. This is the formative paper in demonstrating the viability of iris scans in identifying individuals in a large population. In his work, Daugman proposed a simple algorithm for determining whether two photos of irises are matching, meaning they are from the same individual, or not:

1. Locate the iris's inner and outer boundary rings using a contour integral.
2. Segment the iris area into sectors, removing the bottom quarter of the iris due to the reflection of light off the corona.
3. Use the Gabor Transform to transform the texture pattern into a set of discrete signals.
4. Export the Gabor transformed iris as a sequence of bits to a database.
5. Compute the Hamming distance between the two iris' bit sequences for some number of bit shifts in either direction.
6. Take the minimum Hamming distance from the last step and compare against some threshold. Accept the pair as matching if the minimum distance is below the threshold and reject otherwise.

2.1.3 Segmentation and Location of the Iris

To be able to analyze the iris and determine a potential match, one first needs to separate the iris from the rest of the image. To do this, one realizes that the

iris is simply two concentric circles. The algorithm first considers the middle 25% of the photo as the area that could contain the pupil, then lays a grid over that area. For each x, y on the grid, it computes a contour integral from x, y outwards for various radii r_0. It selects a final x, y, r_0 corresponding to the maximum contour integral, which it assumes is the outer edge of the pupil. It then determines r_1, the outside edge of the iris, based on the maximum of all contour integrals centered at x, y with a radius larger than r_0.

2.1.4 The Gabor Transform

To transform the iris's texture pattern into a signal, a Gabor Transform is used. This is a variant of a short-time Fourier transform, which is a variant of a Fourier transform that divides the signal into several short time frames before ultimately conducting the transformation. This Gabor transform lets us map the recurring texture patterns to a "frequency response," of which is is much easier to find a maximum (or for a mixed signal, its maxima). Note that the iris is a 2-D image, so we the need to use a 2-D Gabor filter. The functional forms for the filter are found in formulas (5) and (6) on page 1151 of Daugman's paper [2].

2.2 Machine Learning Iris Scanning Algorithms

2.2.1 Motivation

As organizations aim to be able to detect irises at distance or with lower quality, Machine Learning-based approaches to iris recognition have emerged. Machine learning is a wide field, so multiple approaches can be taken to analyze irises and determine matches. "An Intelligent Method for Iris Recognition Using Supervised Machine Learning Techniques" by Ahmadi, Nilashi, Samad, and Rashid [3] briefly reviews various machine learning algorithms applied to iris recognition. Common techniques include:

- Neural Networks (of which there are many subcategories)
- Markov networks (a Markov model applied to a random field)
- Genetic Algorithms
- Particle Swarms
- ... and countless more.

Many modern machine learning algorithms use a mixture of the above techniques across multiple steps in the algorithm. However, they overwhelmingly retain the same high-level structure Daugman formulated [2]. Determine the inner and outer rings of the iris, conduct a 2-D Gabor transform to transform the periodic iris pattern into a more practical signal, create some encoding from the transformed data, determine a similarity metric between two encodings, and compare the results of that similarity metric to some cutoff.

3 The Data

3.1 CASIA dataset

In order to analyze the performance of any iris recognition algorithm, a dataset must be chosen. The CASIA-IrisV3 dataset [4] from the Center for Biometrics and Security Research in the Chinese Academy of Sciences (CASIA) is an obvious choice. It contains close-up photos of the eyes of 108 participants, nearly all Chinese graduate students. These were taken over two sessions, with three photos taken in the first session and four in the second session. Each photo is black and white and 320 by 280 pixels in size- see the appendix for a sample image.

3.2 Generating a Distance Datasaet

CASIA also publishes a "distance" dataset [4], which contains full facial photos taken at a larger distance than the standard dataset. However, the website contains a plethora of broken links and dead ends. Downloading the "distance" dataset turned out to be impossible. As a result, to simulate the effect of taking the image of an iris at a distance, we used python's Pillow package [5] to scale down the image to a smaller resolution, then re-scale it back up. The algorithm used for the re-scaling was bicubic interpolation [6], which is a relatively common method for 2-D interpolation for a rectangular grid of points. This resulted in a "blurred" iris. An image is defined to have a "blur factor" of X if our original 320 by 280 pixel photo is scaled down to $320/X$ by $280/X$ pixels, then re-scaled back up to 320 by 280 pixels for further analysis. A blur factor of 5x was selected to generate our "distance" dataset.

4 The Algorithms

4.1 Overview of the Classical Algorithm

The classical algorithm used for this paper is based on the implementation in [7] and mirrors Daugman's algorithm in [2]. Say we are presented photos of eyes A and B and want to determine if the irises are from the same individual or not. The algorithm executes the following steps in MATLAB for each photo:

1. Take the middle 25 percent by x coordinate of the photo. We expect to find the pupil here.
2. For each pixel and radius outward r, compute the contour integral around a circle of radius r centered at that point.
3. Compute the maximum contour integral location and radius. This is the pupil boundary.
4. Given the integral center, repeat the process above using r values larger than the pupil's. This locates the outer edge of the iris.

5. Select some small ∂r, $\partial \theta$ for use in our Gabor transform.

6. Conduct the Gabor transformation on the transformed iris. Encode the Gabor-transformed data into a 240 by 20 array called the "template."

7. Create a 240 by 20 array of "mask" bits for the iris. If the amplitude of the Gabor transformation is less than 0.0001, we assume we are seeing the reflection of light off the cornea and designate a mask bit of "1," else the mask bit is "0."

To compare the two irises A and B, we then complete the following steps in Python:

1. Determine which bits in the 240 x 20 array are unmasked for both irises A and B.

2. Compute the Hamming Distance [10] between the bit arrays of the template files A and B only considering mutually unmasked bits from step 1.

3. Repeat the above step 16 times, with the bits in template file A shifted to the right 1,2,3,4,...16 times. Repeat this process another 16 times, instead shifting the bits to the left. This accounts for potential differences in the rotations of the iris.

4. Compute the final Hamming distance, which is simply the minimum Hamming distance found above.

5. Normalize the final distance above by dividing by the number of unmasked bits and compare to a cutoff value. If the normalized distance is less than the cutoff, conclude irises A and B are from the same individual, else conclude they're from different individuals.

4.2 Overview of the Machine Learning Algorithm

Obviously, there are a large number of approaches one could take to apply machine learning to iris scanning. While Ahmadi et al. [3] discusses a wide variety of machine learning solutions to the iris matching problem, the sources it cites offer no code to accompany their articles. Because building iris-scanning code from scratch would be far too time consuming for the scope of this thesis, I elected to build my code off of Akshata Patel's code base [12]. This code partially implements a relatively fast machine learning algorithm based on spatial filtering, described in more detail in [8] and [9]. To encode a single iris, do the following:

1. Use Python's Glob package to read in the iris photo.

2. Use a Bilateral filter to remove noise using blurring.

3. Use the dimensions of the photo to estimate where the pupil's center may be, then Use the Canny edge detection algorithm to detect edges in the area.

4. Use the Hough Transformation to identify circles in the vicinity above. The circle with the smallest radius is deemed to be the iris's center.

5. Add a set distance to the radius of the pupil to get the outer radius of the iris. Note that this is designed not to capture the outer regions of the iris, as we may get interference from eyelashes.

6. Lay a set of polar coordinates over the image. This allows us to select points at locations a distance r from the pupil's center at an angle θ.

7. Run OpenCV's histogram equalization method on the image using the above coordinates. This increases the contrast by trying to assign roughly equal amounts of various black and white intensities to the image.

At this point, the algorithm has completed image processing and is ready to start matching data. Let's say we have some training data set, and we want to determine if irises A and B are a match or not.

1. Use 2 spatial filters in various locations on the polar transform to create a set of feature vectors. Each spatial filter is, in essence, a selection of an 8x8 pixel area of the iris and an application of the Gabor transformation. Each application of a filter to an area of the iris yields two components: one for the post-Gabor mean amplitude and one for the post-Gabor amplitude standard deviation. This process yields a component vector of dimension $1536 = 6 * 64 * 4$. Repeat this for the entire training dataset.

2. Use SKLearn's Linear Discriminant Analysis (LDA) function [13] to determine what linear combinations of the 1536 features most accurately identify matches. This allows us to reduce the number of components per vector, which significantly reduces runtime at minimal expense to performance. We select 107 as the number of components because we have 108 participants in our training data set and we want to avoid overfitting.

3. Compute the cosine distance [14] between the components of irises A and B, each stored in a vector.

4. Choose some cutoff for the cosine distance based on how well it separates matches and non-matches in the training data. Accept A and B as a match if the cosine distance for the whole vector is less than this cutoff, reject otherwise.

4.3 A Discussion on Cutoff Parameters

In essence, the Hamming distance [10] between two strings of bits is the number of locations the strings differ from each other. For example, "1001" and "1010"

are a Hamming distance of 2 apart. The cosine distance is, in perhaps too simple terms, the cosine of the angle between two input vectors. Vectors that are (positive) scalar multiples of each other would obviously have a cosine distance of zero, perpendicular vectors have a cosine distance of one.

A critical part of both of the above algorithms is the idea of a cutoff parameter or threshold. Both algorithms generate some metric, either Hamming or cosine distance, and accept two irises as a match if their distance is below the threshold. As a result, the errors the two algorithms produce vary based on the chosen thresholds. Setting a lower threshold will decrease our false match rate but will also increase our false non-match rate, and setting a higher threshold will have the opposite effect. This effect can be displayed graphically on a ROC (receiver operating characteristic) [11] curve. On the X and Y axes are the false match and false non-match error, and a point (x,y) is on the ROC curve if there is some setting of the threshold that creates false match and false non-match errors of x and y respectively. Some of these ROC curves for both algorithms can be found in the appendix.

Obviously, while ROC curves are helpful in explaining the performance of a model as its threshold is changed, it makes comparing algorithms more complex. This is especially true when we're already adjusting the quality of the input image as a variable. As a result, our conclusions will be drawn using the thresholds that minimize total error, aka maximize the percentage of correct match or non-match predictions. These thresholds would be different if false match and false non-match errors were given unequal weights, but this is beyond the scope of my analysis. However, I strongly suspect that the conclusions drawn would be the same or highly similar.

5 The Tests

With the algorithms coded, it is time to test their performance. The obvious question is, "How do we test the performance of an iris scanning algorithm?" Ideally, tests of the algorithms' performance should mimic the real-world applications of iris recognition. Two of the most common applications of iris scans are to *confirm identity* or *search for matches in a population*. In the former case, we aim to determine whether a given iris corresponds to a specific preexisting iris scan. We have somebody claiming to be some specific person, and the iris scan confirms or rejects that claim. In the latter case, we use an iris scan to search through a database, attempting to find a matching scan and thus determine the identity of the individual.

5.1 Short-Distance Identity Confirmation

5.1.1 The Test

This test determines how accurately a given algorithm can identify whether two iris scans are from the same individual or not. In this case, both iris scans are pulled from the standard CASIA Iris v3 dataset with no blurring. Recall that the dataset has 108 participants, each with three to four photos from two sessions. To make this test as realistic as possible, one iris scan is pulled from the first session and the other from the second session. This process is repeated 324 times for both random different individuals and the same individual.

5.1.2 The Results

In terms of accuracy, the classical algorithm outperformed the machine learning across both data sets by a wide margin. The optimal Hamming Distance cutoff for the classical algorithm was .43, and the optimal cosine distance cutoff for the machine learning algorithm was .85. The total error for the classical algorithm was .0077, while the total error for the machine learning algorithm was .1082, meaning the error for the classical algorithm was over 14 times lower. More detailed results by cutoff can be found in the appendix.

5.1.3 A Note on Runtime

Comparing the runtimes of the two algorithms is somewhat a comparison of apples and oranges, as each algorithm has a different set of tasks that need to be run once overall versus once per input iris. For example, the classical algorithm encodes each iris into a bit string once, then re-uses that same bit string whenever the iris needs to be compared to another. Similarly, the machine learning model only needs to train itself once, then can use the resulting training vectors in each comparison it makes. The machine learning algorithm took 1 minutes and 56 seconds to train itself, and a total of 41 seconds to produce the total error for a given cosine distance cutoff. The classical algorithm took 2 minutes and 51 seconds to generate the mask and template files in MATLAB, then 1 minute and 18 seconds to compute the total error for a given Hamming distance cutoff. As a result, while the algorithms vary significantly in their overall structure, the machine learning algorithm has a lower runtime in both the once per test and once per iris components. Thus, I conclude that the machine learning algorithm has a lower runtime overall.

5.2 Long-Distance Identity Confirmation

5.2.1 The Test

This test is identical to the above, but one iris is blurred by a factor of 5x. Again, to make this test as realistic as possible, the unblurred iris scan is pulled from the first session and the blurred scan from the second session. This is done to simulate,s the comparison of an iris photo in some database taken under good

conditions to a lower-quality image a person on the move or at a distance might produce.

5.2.2 The Results

Yet again, the classical algorithm outperformed the machine learning across both data sets by a wide margin. The optimal Hamming Distance cutoff for the classical algorithm was still .43, but the overall error was very close to the error of the .44 cutoff. The optimal cosine distance cutoff for the machine learning algorithm increased from .85 to .9. Both algorithms performed significantly worse than before but still clearly had predictive power. The total error for the classical algorithm was .0231, while the total error for the machine learning algorithm was .2221, meaning the error for the classical algorithm was over 9 times lower. The runtimes were the exact same in this test as they were in the prior short-distance test because no step in the algorithm has a runtime dependent on image quality.

5.3 In/Out of Population Test

5.3.1 The Test

In this test, a "population" of 54 individuals chosen from the CASIA subjects is created. 54 is selected as a population size because 108 individuals comprise the CASIA dataset. One iris scan is collected from session one for each individual in the population. Each algorithm is presented with a session two iris scan and asked who, if anyone, in the database matches the scan. Some of these session two irises will come from individuals in the population, others will not. This is the short-distance in/out of population test. The experiment is then repeated, but with the session two iris scans coming from the 5x blurred long-distance dataset. This is the aptly named long-distance in/out of population test.

5.3.2 Results

Yet again, the machine learning algorithm outperforms by its classical counterpart in both distances. For the short-distance test, the machine learning algorithm's optimal cutoff is a cosine distance of .62, which has a total error of .1921. The classical algorithm's optimal cutoff is a Hamming distance of .41, which has a total error of .0119. Similarly, for the long-distance dataset, the machine learning algorithm's optimal cutoff is tied between a cosine distance of .6 and .61, both of which have a total error of .4051. The classical algorithm's long-distance optimal cutoff is a Hamming distance of .41, which has a total error of .0418. However, had finer Hamming distance cutoffs been examined, I suspect the optimal distance would be .405. All of these results can be found for a wider range of cutoffs in the appendix.

These results are rather expected. Because we are at minimum comparing 53 non-matching iris scans from our population to our session 2 candidate iris, the

hamming distance or cosine distance cutoffs need to be slightly lower to decrease the chance of a false match. We expected our error to be larger for this test than the identity confirmation test as a result. The results supported both hypotheses. In terms of runtime, the machine learning algorithm yet again outperformed the classical algorithm by a considerable margin. This makes sense: a run of this test for a given algorithm is simply a run of its once-per-test component then 54 comparisons of pairs of irises.

5.4 Finding the Machine Learning-Superior Blur Factor

Up to this point, the classical algorithm has been more accurate than the machine learning algorithm in every test and every level of blur. The obvious question is then, "Is there any level of blur at which the machine learning algorithm outperforms the classical algorithm?" To answer this question, we re-ran the identity confirmation tests and computed the confusion matrices for 10x and 20x blur factors. The lowest total error for the 10x blur test on the machine learning algorithm was 0.3062 using cosine distance cutoff of 0.935. The lowest total error for the 20x blur test on the machine learning algorithm was 0.3913 with a cutoff of 0.965. The lowest total error for the 10x blur test with the classical implementation was 0.1358 using Hamming distance cutoff of .44. The cutoff for the 20x blur test and classical algorithm was still .44, but now with a total error of 0.4367. These results suggested that the minimum level of blur needed for the machine learning algorithm to outperform its classical counterpart was somewhere between 10 and 20x. After testing a variety of intermediate blur factors, I determined that the Machine Learning-superior blur factor was found to be roughly 17.5x, which yielded a total error of 0.3873 for the classical algorithm and 0.3780 for the machine learning algorithm.

5.5 Test of Normality

One thing to note about the in/out of population tests was the fact that the optimal distance cutoff is dependent on the size of the population. The larger the population, the smaller the cutoff needs to be in order to minimize total error. For each in/out of population performed above, the size of the population was kept constant at 54, one half of the CASIA dataset's population. However, if the exact distributions of the distances between matching and non-matching iris scans were known, one could determine the exact optimal cutoff and performance for any sized population. After looking at the distributions of the various Hamming distances, I suspected that the distribution of hamming distances between iris scans from the same individual matched a normal distribution.

To test this suspicion, I first computed the mean and standard deviation of our Hamming distances. These distances were divided into "bins" of .005 width, then a chi-squared test was ran on the expected vs. actual number of distances in each bin. For a visualization of the Hamming distance distributions of iris scans from the same or different individuals, see the appendix. Without any

modifications to our data, we get a p-value of 2.4527E-08 for scans from the same individual and 0.0009039 for scans from different individuals. This alone suggests the distributions of the Hamming distances are not normal. However, three quarters of the chi-squared statistic for the same-individual distribution came from two outlying data points. The Mahalanobis Test identified that one of these data points had a Mahalanobis Distance of over 3.5 and should be removed. After removing this data point, the same-individual distribution now had a p-value of .23, which suggests the distribution is normal. However, the different-individual distribution is clearly not normally distributed. I suspect that the Hamming distance from any random iris scan to a different iris scan is binomially distributed with parameters p = .5 and n in the tens or hundreds. However, recall that the final Hamming distance is the minimum hamming distance of 33 total bit rotations of one iris scan's encoding. This final distribution, from my research, does not have a commonly used name or particularly concise form.

After seeing this result, I re-ran the experiment using the cosine distances from the Machine Learning algorithm. It returned p-values of 0.09181 for scans of the same individual and 0.9750 of different individuals. This suggests both distributions are normal, which makes sense; unlike the classical algorithm, we don't need to take any minimum distance of some set in computing our final distance.

6 Conclusion

The classical algorithm outperforms the machine learning algorithm in both the identity confirmation and in/out of population tests for both 1x and 5x blur, although the machine learning algorithm edged out the classical version in runtime. We determined that the machine learning algorithm outperforms the classical algorithm at blur factors of 17.5x and higher but under-performs otherwise. Given these results, the classical algorithm is objectively superior to the machine learning algorithm in virtually every application. Despite taking roughly twenty percent longer to run, the errors for the classical algorithm at low blur factors are so much lower than the machine learning algorithm's that it I could not justify recommending the latter over the former. At 1x blur factor, the error for the classical algorithm is roughly 14 times lower than the machine learning algorithm's. While this ratio admittedly decreases to 9.6 times then 4.5 times as the blur factor increases to 5x and 10x, these are still massive differences in performance. Once we reach a blur factor of 17.5x where the machine learning algorithm performs as well or better than the classical algorithm, we are looking at total error of 38 percent or more in the identity confirmation test. The iris images are, at this point, 18 by 16 pixels. The error rates for both models are so incredibly high that the practical applications of the models are minimal and the performances of the algorithms are nearing a coin flip. Thus, for all reasonable levels of blur, the classical algorithm significantly outperforms the machine learning version.

That being said, this machine learning algorithm is a relatively simple one of many. As demonstrated in [3], there are a plethora of different approaches to using machine learning to match iris scans. These iris scans are far more sophisticated in how they identify what sections of the iris photo to consider, how they map features of the iris to feature vectors, and how they conduct training than the method I coded. The accuracy of these methods range from 95.36 percent to 99.99 percent; see table 9 in [3] for a table of proposed methods and performances. However, these performance boosts come at the expense of runtime. In the case of [3], in order to train their model on the same CASIA Iris dataset enough to get their desired 99.99 percent accuracy, they needed a runtime of over **56 minutes** on presumably far superior machinery to my Mac Book Pro.

6.1 Future work

If I were to extend this work but still focus on iris recognition algorithm comparison, I would invest more time in developing one of these more advanced machine learning algorithms in Python. The algorithm from [3] is exceptionally promising, but again would take an exceptionally long time to develop, utilize, and test. A more modest extension would be to acquire a more comprehensive data set than the CASIA Iris V3, perhaps including a more realistic long-distance dataset. However, there are all sorts of ways one could extend this work if we forgo focusing on identity recognition via iris scans. New research could, for example, focus on identifying disease or tiredness rather than recognizing an individual. One could even forgo the irises themselves, focusing on matching other biometric identifiers such as the whole face or fingerprint. As with many areas of research, the possibilities are endless.

7 Works Cited

1 - https://onin.com/fp/fphistory.html
2 - https://www.cl.cam.ac.uk/~jgd1000/PAMI93.pdf
3 - https://www.sciencedirect.com/science/article/pii/S0030399218320553
4 - http://www.cbsr.ia.ac.cn/english/IrisDatabase.asp
5 - https://pillow.readthedocs.io/en/stable/reference/Image.html
6- https://www.mathworks.com/help/vision/ug/interpolation-methods.html
7- https://github.com/thuyngch/Iris-Recognition
8 - https://www.periyaruniversity.ac.in/ijcii/issue/Vol3No3December2013/IJCII%203-3-114.pdf
9 - https://www.witpress.com/elibrary/wit-transactions-on-the-built-environment/151/33356
10 - https://en.wikipedia.org/wiki/Hamming_distance
11 - https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

12 - https://github.com/akshatapatel/Iris-Recognition
13 - https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html
14 - https://www.machinelearningplus.com/nlp/cosine-similarity/

8 Appendix

8.1 Code

You can find the Python code I developed at https://github.com/toryfarmer/SeniorThesis. This contains code to blur images, the machine learning algorithm, the Python portion of the classical algorithm, the identity confirmation tests, and the in/out of population tests.

8.2 Figures

Figure 1: Sample CASIA Iris photos. Left: No blur. Right: 5x blur.

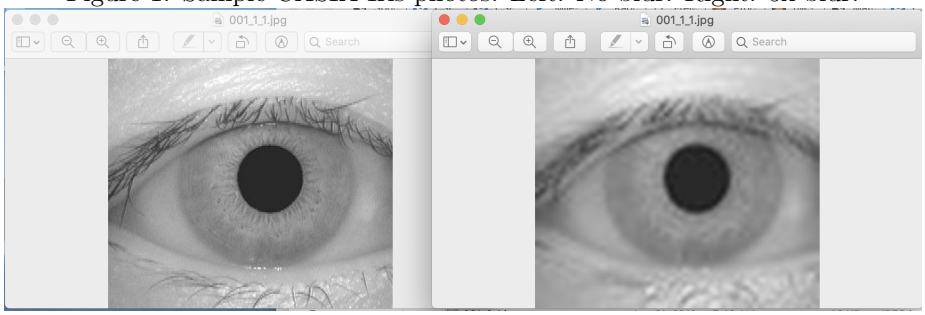


Figure 2: Confusion Matrices for various cutoff parameters in the identity confirmation test.

1x Blur						5x Blur					
Classical	Trials: 324		Machine Learning	Trials: 324		Classical	Trials: 324		Machine Learning	Trials: 324	
Cutoff:	0.3700		Cutoff:	0.8200		Cutoff:	0.4200		Cutoff:	0.8700	
	Returned Match	Returned Non-Match		Returned Match	Returned Non-Match		Returned Match	Returned Non-Match		Returned Match	Returned Non-Match
Actual Match	0.8549	0.1451	Actual Match	0.8519	0.1481	Actual Match	0.9660	0.0340	Actual Match	0.7106	0.2894
Actual Non-Match	0.0000	1.0000	Actual Non-Match	0.0674	0.9326	Actual Non-Match	0.0062	0.9938	Actual Non-Match	0.1602	0.8398
Cutoff:	0.4000		Cutoff:	0.8400		Cutoff:	0.4300	Error: .0231	Cutoff:	0.8800	
	Returned Match	Returned Non-Match		Returned Match	Returned Non-Match		Returned Match	Returned Non-Match		Returned Match	Returned Non-Match
Actual Match	0.9630	0.0370	Actual Match	0.8650	0.1350	Actual Match	0.9722	0.0278	Actual Match	0.7338	0.2662
Actual Non-Match	0.0000	1.0000	Actual Non-Match	0.0781	0.9219	Actual Non-Match	0.0185	0.9815	Actual Non-Match	0.1782	0.8218
Cutoff:	0.4100		Cutoff:	0.8400		Cutoff:	0.4400		Cutoff:	0.8900	
	Returned Match	Returned Non-Match		Returned Match	Returned Non-Match		Returned Match	Returned Non-Match		Returned Match	Returned Non-Match
Actual Match	0.9784	0.0216	Actual Match	0.8750	0.1250	Actual Match	0.9846	0.0154	Actual Match	0.7515	0.2485
Actual Non-Match	0.0000	1.0000	Actual Non-Match	0.0894	0.9106	Actual Non-Match	0.0340	0.9660	Actual Non-Match	0.1972	0.8028
Cutoff:	0.4200		Cutoff:	0.8500	Error: .1082	Cutoff:	0.4500		Cutoff:	0.9000	Error: .2221
	Returned Match	Returned Non-Match		Returned Match	Returned Non-Match		Returned Match	Returned Non-Match		Returned Match	Returned Non-Match
Actual Match	0.9815	0.0185	Actual Match	0.8858	0.1142	Actual Match	0.9938	0.0062	Actual Match	0.7739	0.2261
Actual Non-Match	0.0000	1.0000	Actual Non-Match	0.1021	0.8979	Actual Non-Match	0.0988	0.9012	Actual Non-Match	0.2182	0.7818
Cutoff:	0.4300	Error: .0077	Cutoff:	0.8600		Cutoff:	0.4600		Cutoff:	0.9100	
	Returned Match	Returned Non-Match		Returned Match	Returned Non-Match		Returned Match	Returned Non-Match		Returned Match	Returned Non-Match
Actual Match	0.9907	0.0093	Actual Match	0.8958	0.1042	Actual Match	0.9969	0.0031	Actual Match	0.7894	0.2106
Actual Non-Match	0.0062	0.9938	Actual Non-Match	0.1172	0.8828	Actual Non-Match	0.2500	0.7500	Actual Non-Match	0.2397	0.7603
Cutoff:	0.4400		Cutoff:	0.8700					Cutoff:	0.9200	
	Returned Match	Returned Non-Match		Returned Match	Returned Non-Match					Returned Match	Returned Non-Match
Actual Match	0.9907	0.0093	Actual Match	0.9035	0.0965				Actual Match	0.8110	0.1890
Actual Non-Match	0.0340	0.9660	Actual Non-Match	0.1336	0.8664				Actual Non-Match	0.2618	0.7382
Cutoff:	0.4600		Cutoff:	0.8800					Cutoff:	0.8800	
	Returned Match	Returned Non-Match		Returned Match	Returned Non-Match					Returned Match	Returned Non-Match
Actual Match	0.9907	0.0093	Actual Match	0.9151	0.0849				Actual Match	0.8302	0.1698
Actual Non-Match	0.1019	0.8981	Actual Non-Match	0.1509	0.8491				Actual Non-Match	0.2857	0.7143

Figure 3: ROC curves for the identity confirmation test.

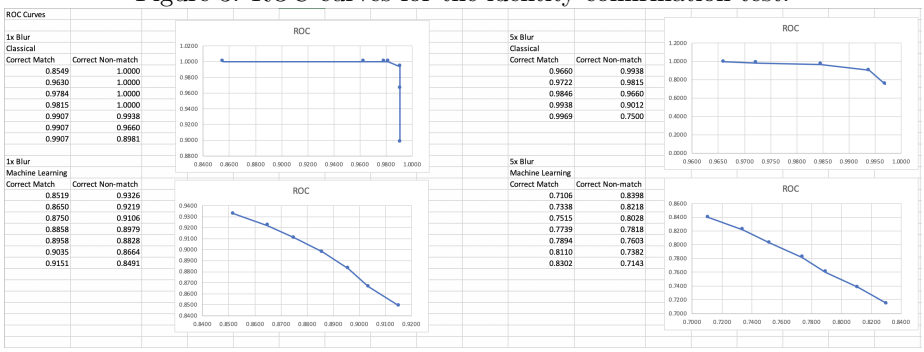


Figure 4: Performance of identity confirmation by blur factor.

Blur Factor	Adjusted Width	Adjusted Height	Classical Total Error	ML Total Error
1	320	280	0.0077	0.1082
5	64	56	0.0231	0.2221
10	32	28	0.0679	0.3062
12.5	26	22	0.1235	0.3240
15	21	19	0.2654	0.3465
17.5	18	16	0.3873	0.3780
20	16	14	0.4367	0.3913

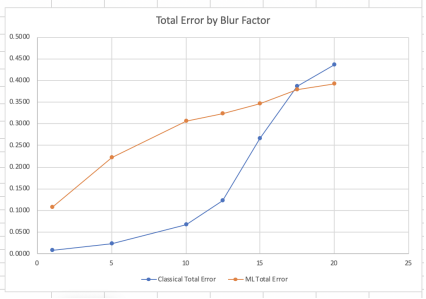
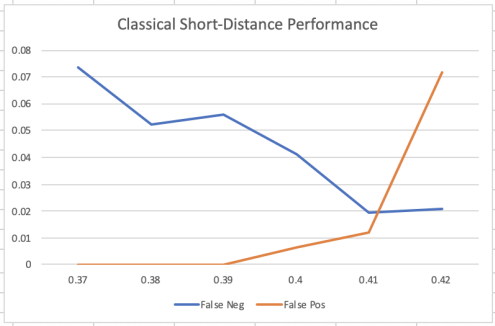


Figure 5: In/Out of Population test performance for classical algorithm.

Classical			
Short-Distance			
Cutoff	False Neg	False Pos	Total Error
0.37	0.07361963	0	0.03680982
0.38	0.05232558	0	0.02616279
0.39	0.05590062	0	0.02795031
0.4	0.04093567	0.00653595	0.02373581
0.41	0.01923077	0.01190476	0.01556777
0.42	0.02097902	0.0718232	0.04640111
Runtime:	12mins 14 secs		



Classical			
Long-Distance			
Cutoff	False Neg	False Pos	Total Error
0.4	0.0955414	0	0.0477707
0.41	0.06521739	0.01840491	0.04181115
0.42	0.05112474	0.06418219	0.05765347
0.43	0.05015674	0.14285714	0.09650694
0.44	0.04808878	0.24969098	0.14888988
0.45	0.04338843	0.36782787	0.20560815

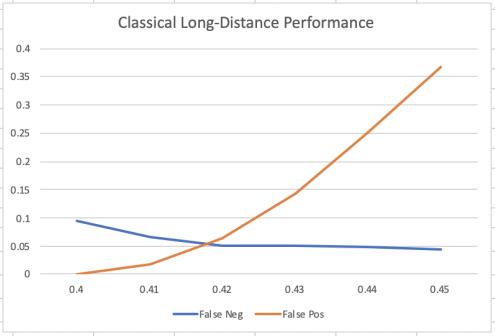
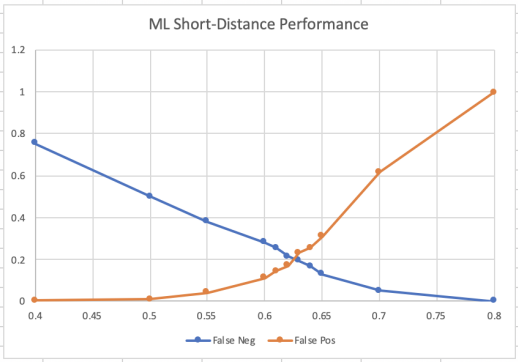


Figure 6: In/Out of Population test performance for machine learning algorithm.

Machine Learning			
Short-Distance			
Cutoff	False Neg	False Pos	Total Error
0.4	0.75462963	0.00462963	0.37962963
0.5	0.5	0.00925926	0.25462963
0.55	0.37962963	0.04166667	0.21064815
0.6	0.28240741	0.11111111	0.19675926
0.61	0.25462963	0.14351852	0.19907407
0.62	0.21296296	0.1712963	0.19212963
0.63	0.19444444	0.23148148	0.21296296
0.64	0.16666667	0.25462963	0.21064815
0.65	0.12962963	0.31018519	0.21990741
0.7	0.05092593	0.61574074	0.33333333
0.8	0	0.99537037	0.49768519



Machine Learning			
Long-Distance			
Cutoff	False Neg	False Pos	Total Error
0.4	0.95833333	0.00462963	0.48148148
0.5	0.83796296	0.07407407	0.45601852
0.55	0.68981481	0.19907407	0.44444444
0.6	0.45833333	0.35185185	0.40509259
0.61	0.40277778	0.40740741	0.40509259
0.62	0.37037037	0.44907407	0.40972222
0.63	0.30092593	0.51851852	0.40972222
0.64	0.24074074	0.59259259	0.41666667
0.65	0.19907407	0.63425926	0.41666667
0.7	0.05092593	0.91203704	0.48148148
0.8	0.00462963	1	0.50231481

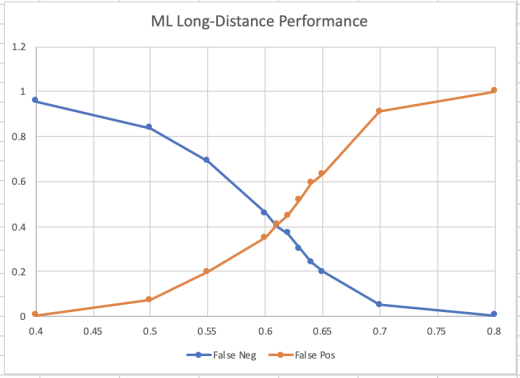


Figure 7: Distribution of Hamming Distances. Orange are same individual, blue are different.

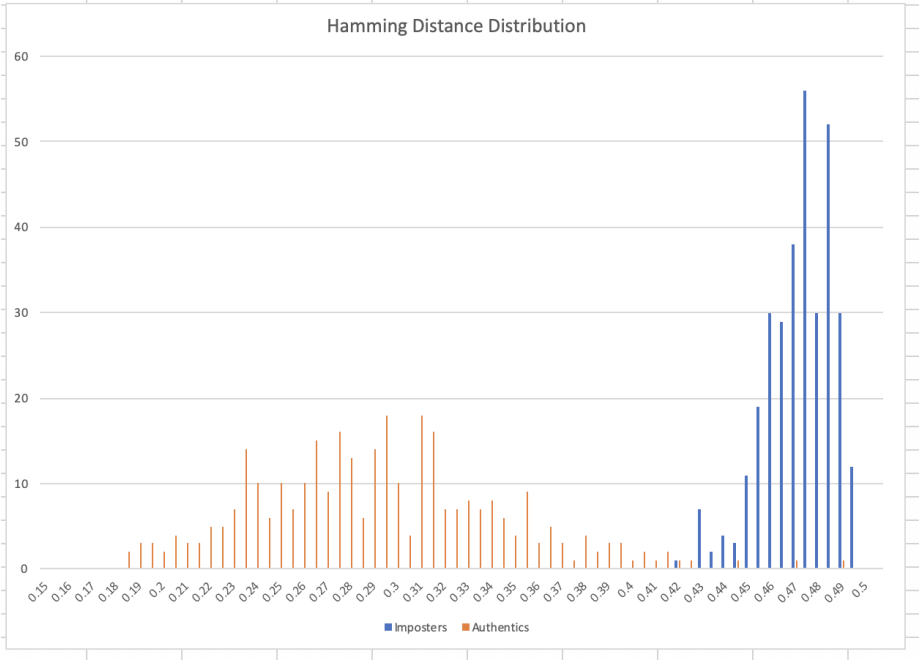


Figure 8: Distribution of Cosine Distances. Blue are same individual, orange are different.