Evasion Attacks on PyPi face-recognition

Anna Gerchanovsky

Abstract

The PyPi face-recognition library is a facial recognition library that has been downloaded hundreds of thousands of times and used for the purposes of facial recognition and detection. In this paper, I analyze the performance of face-recognition on publicly available face databases and evaluate its vulnerability to a variety of evasion attacks in three separate experiments. [1]

1 INTRODUCTION

Although techniques used in this project may be applicable to other facial recognition APIs, the scope of this project included only face-recognition, an open source project available via PyPi. The project is built on dlib's deep learning facial recognition model.

1.1 Usage

The face-recognition project on GitHub has close to 50k stars and 1.6k watching, at time of writing[5]. Within the past year, the face-recognition project has been downloaded 637,306 times from PyPi[1]. A search of GitHub to view uses of face-recognition shows 12.5k Python files that `import face_recognition`. A project using this that seemed of note to me was Moswag's FacialRecognitionForPolice, which, according to its README, claims to "Recognize faces of thieves captured with CCTV using the world's simplest face recognition library"[8]. While this repository is not largely popular, it does speak to the potential uses of facial recognition and the importance of fixing bugs that may cause misrecognition, as does the wide usage as seen in the number of GitHub imports and PyPi downloads.

1.2 Syntax

The face-recognition project has a few methods that are relevant in this paper - used both for facial recognition and for applying specific transformations to faces later.

`face_encodings` returns a list of 128-dimension face encodings given an image with some amount of faces.

`face_landmarks` returns a list of facial dictionaries, with facial features like left eye, nose, etc as keys and their coordinates as values, given an image with some amount of faces.

`compare_faces` returns a list of booleans given a list of face encodings and a face encoding for them to be compared to, where each boolean represents whether or not each face encoding element of the list matches the given encoding.

`face_distance` returns a list of floats given a list of face encodings and a face encoding for them to be compared to, where each float represents the distance between the face encoding element of the list and the given encoding.

1.3 Adversarial Machine Learning

Adversarial machine learning, or attacks on machine learning algorithms, come in a variety of forms[4].

[1] Some code used in and results of this project are available at `https://github.com/agercha/EvasionAttacksFaceRecognition`

Model Poisoning: providing specific inputs to a model under training to cause it to be trained incorrectly or underperform in deployment, which is particularly possible in an active learning model.

Model Theft: extracting either the model itself or the training data of the model through usage.

Evasion Attacks: cause a model to purposefully misclassify inputs. Examples of evasion attacks may be getting past a spam filter in a mailbox (or vice versa) or tricking road recognition of AV systems[10][11]. This kind of attack is the focus of this paper. Specific evasion attacks on facial recognition will be described in the next subsection.

1.4 Evasion Attacks on face-recognition

Because there are a variety of tools and capabilities of face-recognition, there is also a variety of evasion attacks that can be performed. I will define the failures of facial recognition of face-recognition as such, even though they may be referred to in another way in other literature:

Misdetection: This failure consists of a face no longer being detected by face-recognition. This consists of an empty list being returned by `face_encodings`, `face_landmarks`, or a variety of other functions, meaning no face was found in an image. In the wild, this attack would allow a person to remain undetected by facial recognition. This attack means facial recognition is less powerful in security contexts.

Misrecognition: This failure consists of a face encoding of a certain person no longer matching another face encoding of this person (or not matching a list of all known face encodings of this person). This will consist of a false being returned by `compare_faces`, or a distance of over 0.5 returned by `face_distance`. In the wild, this attack would allow an image of a person to not be matched back to themself.

Misidentification: This failure consists of a face being matched to the wrong person. This will consist of `face_distance` returning a larger distance between some image and the encoding of the same person versus of some target person. In the wild, this attack is likely the most dangerous, as it not only allows a person to evade facial recognition, but it matches another person to that person's image. This could be especially harmful in security or legal situations, where an innocent person may be matched to the face of an attacker.

2 TESTING DATASETS

For the context of this project, I considered a few different datasets of faces from MIT's list of face databases[2].

LFW (Labeled Faces in the Wild): This is the largest available dataset of faces, with 13,233 images and 1,680 individuals. This is a collection of faces scraped from the web and dlib's model claims a 99.38% accuracy on faces in this dataset. While this dataset is large, it is not perfect and nonexhaustive. The website disclaimer warns that certain age groups and ethnicities are underrepresented or not represented, and I've found this to be especially the case when considering only individuals with multiple images of them in the database[6].

MUCT: This dataset is smaller than LFW, with 3,756 images and 276 individuals. This dataset is somewhat diverse. Each subject is labeled via whether or not they wear glasses and their gender, and are shot from a variety of angles and lightings, resulting in 10 or 15 images of each subject. However, the lighting change is not necessarily intense, with mild changes between dark and light, as well as one lighting with a blueish tint[7].

Yale: This is the smallest dataset, with 165 images and only 15 individuals. This dataset is also less diverse than the other two, including only one woman and with many ethnicities not represented at all. However, there are 11 images of every subject, under conditions that MUCT doesn't provide: centerlight, glasses, happy, leftlight, no glasses, rightlight, sad, sleepy, surprised, and wink.

Especially valuable, in this case, are the extreme lightings (leftlight and rightlight), which are not covered by MUCT[3].

Ultimately, I performed most of my analysis and testing on MUCT. I chose it because it was fairly large, fairly diverse, well labeled, and accessible to use. However, I did perform some experiments on the Yale dataset as well.

3 EXPERIMENT 0: PRELIMINARY ANALYSIS

Prior to attempting evasion attacks on facial-recognition, I performed some exploratory experiments on its performance on the MUCT and Yale face databases to both get a baseline of its performance, and understand where it tends to lack, even without any changes.

No misdetections occurred on the MUCT database, but some were found on Yale. Subjects 1, 5, and 11 were not detected under rightlight. Additionally, when it was tested on the normalized Yale dataset, which includes the same images but rotated so the eyes are aligned, centered, so the center of the eyes is in the center, and cropped to just keep the faces, misrecognitions occurred also on subjects 4 and 8.

Figure 1: Yale Subjects 1, 4, 5, 8, and 11 under rightlight, where they are misdetected

Next, I analyzed the distances between images of the same person under the different conditions provided by the datasets. For the Yale dataset, on average, noglasses and normal performed best (these two are essentially identical), while all others were worse, but no distance went over 0.4, with most hovering around 0.3.

For the MUCT dataset, the distances between images of subjects with glasses was higher than the distances between images of subjects without, indicating a lower level of confidence for users with glasses. The

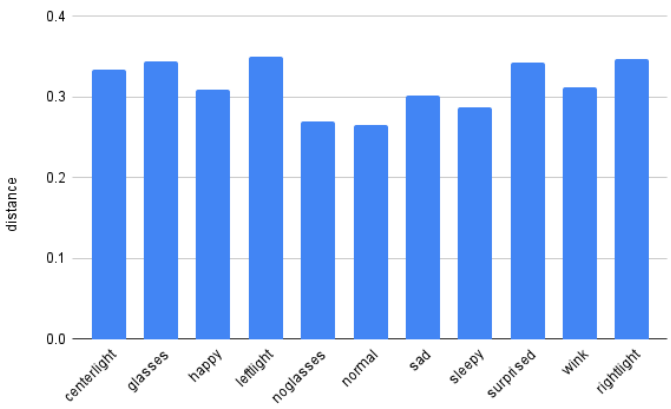

Figure 2: Average distances over different variations of images in the Yale dataset

distance between men and women was negligible. Images with blue tinted lighting x, v, f, and z all performed well when compared to images under the same lighting, but badly when compared to images of another lighting. View had relatively little effect.

Figure 3: pairs of photos of MUCT subjects 45, 50, 58, and 265 that had a distance of over 0.5

There are also 9 comparisons within subjects of MUCT that return a distance of over 0.5, which I define as a misrecognition. They occur for subjects 45, 50, 59, and 265.

Another observation of note is that the distances between images in MUCT were lower than that of images in the Yale dataset. My guess for this is that the images in MUCT are higher quality, allowing for higher

certainty.

4 EXPERIMENT 1: REALISTIC FILTERS

Figure 4: filters, labeled left to right, top to bottom: blurweak, blurstrong, clown, darkeneyes, darkenshadowsfast, darkenshadowsslow, contrast, lighten, darken, rotate45, rotate90, and flip

4.1 Implementation

The first attempt at evasion attacks on face-recognition consisted of applying a variety of "realistic" filters on images and attempting to compare them to a "reference face". For MUCT, this reference face was the "a" view (straight on) and the first lighting, which tended to be the most natural. This experiment focused on achieving misdetection and misrecognition failures on face-recognition.

I created a variety of transforms, consisting of blurring transforms (blurstrong and blurweak), rotating transforms (rotate45, rotate90, and flip, which rotates by 180 degrees), image adjustments (darken, contrast, lighten), facial feature aware filters (clown, darken eyes, and two darken shadows filters), and two filters that noise (random1 and random2). All these filters performed various amounts of changes, so those that were milder to the human eye were considered more heavily. Overall, blurstrong and both noise filters tended to add enough change to make recognition difficult for me, while the rest were overall reasonable.

While preliminary exploration had been done exhaustively, the application of some of these filters took a long time, so testing every image of a subject with every transformation applied to it against the reference image of that subject became unfeasible. Instead, the base encodings were recorded, then a random subject, view, lighting, and transformation were chosen and applied, and compared to the base encoding on a continuous basis. At time of writing, over 2000 comparisons have been made.

4.2 Results

Results of the experiment showed that six of the filters ever caused misdetections, those being all the rotation based filters, all the noise filters, and blurstrong. As mentioned earlier, blurstrong and the noise filters I consider to add too much change to the image to be considered successes. However, all the rotational filters, in my opinion, remain recognizable by the human eye, which was interesting. Unsurprisingly, rotating by 45 degrees caused a misdetection less than any other rotation. Somewhat surprisingly, rotating 90 degrees caused a misdetection 50% more than rotating 180 degrees. The effects of rotations on misdetections and misrecognitions are considered in the following section on rotation experiments.

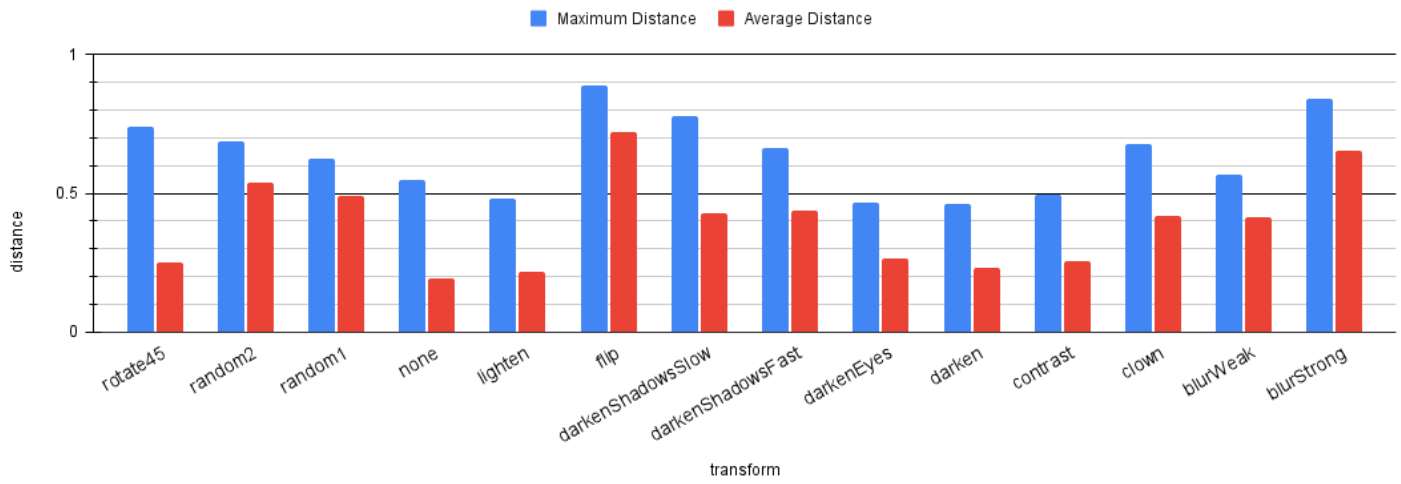

Figure 5: distances during experiment 1

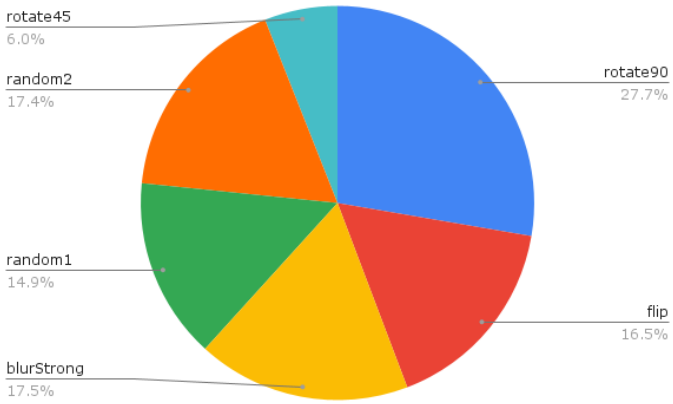

Figure 6: misrecognitions during experiment 1

When it comes to comparing the distances when transformed with certain filters, three filters had a median value of over 0.5, which meant they caused a misrecognition more than half the time: flip, random2, and blurstrong. Additionally, random1 came very close to 0.5. Again, the only result necessarily of note is the success of flip, with mean and median distances of over 0.7.

When analyzing the maximum distances achieved with a certain filter, rotate45, blurweak, darkenshadowsfast, darkenshadowsslow, clown, and random1 all reach a value over 0.5, meaning they all achieve at least one misrecognition. However, a value of 0.5 is achieved even with no filter applied, as mentioned in the preliminary results, so this is not necessarily impressive. However, further analysis showed that each of these filters was able to achieve misrecognition on a subject other than 45, 50, 59, and 265, which had been misrecognized without a filter.

Additionally, all the implemented filters had potential adjustments to their strength or scope that could affect how well they perform. The image adjustment filters and darkeneyes all achieved close to a 0.5 distance at maximum, meaning that they very well could successfully cause a misrecognition if strengthened. The filters were kept static as I judged their current state to not be too strong to cause misrecognition by human eyes, but their strength could be toggled with and adjusted. This experiment was not ultimately one I chose to pursue for this project, but may be worth exploring.

5 EXPERIMENT 2: ROTATIONS

Following the success of the rotational transformations, I continued further testing in this area. For me, this was particularly interesting, as these transformations were a

lot "milder" than some of the others, i.e. they are still very easily recognizable to the human eye.

5.1 Implementation

Here, the fuzzing was again continuous and random, and around 6,000 combinations were tested. The only difference was, rather than randomly selecting a transformation from a list, a random degree of rotation was chosen between 0 and 360.

5.2 Results

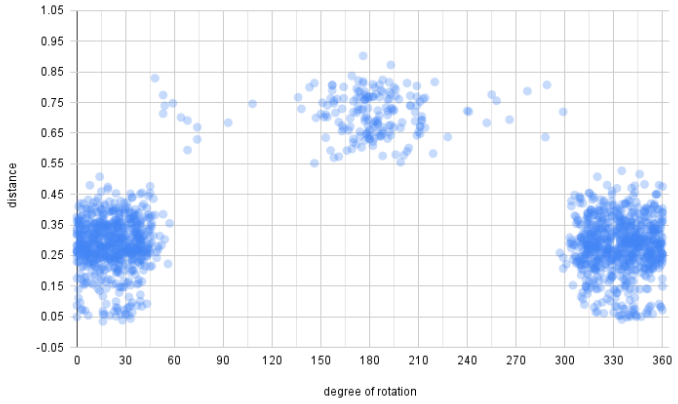

Figure 7: distances during experiment 2

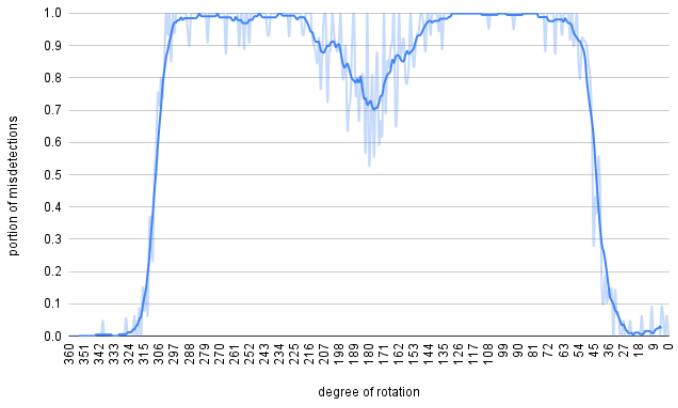

Figure 8: misrecognitions during experiment 2

Again, the results of these experiments were analyzed in terms of misdetections and misrecognitions. The results showed that, for rotations of up to 60 degrees in either direction, and especially for 45 degrees or less, there were relatively few misrecognitons and misdetections. This was indicated by the results in the previous two sections - as some misrecognitions occur even without filtered and some misdetections occurred with rotations of 45 degrees. Beyond these rotations, misdetections are overwhelmingly common. Almost all datapoints rotated between 60 and 135 degrees in either direction could not have a face be detected by face-recognition. For rotations between 135 and 225 degrees, there were actually notably more faces recognized, but this remained low. When faces were recognized at any of these large angles, the distance was always over 0.5, which was classified as a misrecognition.

6 EXPERIMENT 3: MISIDENTIFICATION WITH GENETIC ALGORITHM

All previous experiments had provided interesting analysis aimed to successfully cause misdetections or misrecognitions, but had not focused on misidentifications, which could potentially be the most dangerous attack. Because this was a more difficult target to hit, I focused less on the realism of transformations (i.e. this could occur in a photo without deliberate attacker intervention), while still minimizing changes to the image (i.e. the face remains recognizable to the human eye).

6.1 Implementation

In this experiment, the same image is transformed every time. Over some number of generations, the best performing transformations are recorded and used to create the next generation of transformations via a loose "genetic algorithm", explained further later.

6.1.1 Goal For misidentifications, a specific starting image of a specific subject and a target subject were

chosen. The aim is in general to make the average distance between the images of the subject in the starting image be higher than the average distance between the images of the target subject. At this point, the modified image is considered misidentified as the target subject.

Additionally, some experiments were run just to perform misrecognitions and misdetections, for the purpose of developing the transformations and algorithm. The results of this are discussed in the next section and the method is similar to that of misidentifications, with some pretty straightforward modifications (i.e. what the best performing filter is just depends on distance from the starting image, no target image is selected)[2].

6.1.2 Transformations There were a few iterations of this part of the experiment. First, I attempted variations using noise, but at much lower opacities, but this did not end up yielding good results. Ultimately, the transformations consisted of rectangles, which were randomly chosen to be filled or not, of random dimensions, colors, and line thicknesses, that covered up to 1% of the image. The locations and sizes and rectangles were chosen within certain ranges, but this varied by experiment.

```
def mate_filters(f0, f1, iter):
    filled = 0
    rects = []
    while filled < max_cover:
        curr_rect = random.choice(rects_old)
        # update filled
        rects.append(curr_rect)
    if len(rects) > 0: rects.pop(-1)
    return FilterIm(rects, "mate", iter)
```

Figure 9: Simplified Mating Function

[2]video of progression by generation: https://drive.google.com/file/d/1kr1409mLMfL_bE43iImvq15oX4GSKm0u/view?usp=sharing

6.1.3 Genetic Algorithm At the start of the experiment, a predetermined number of filters is generated. They are all tested, and the average distance between the start subject and target subject is recorded. This becomes the original pool of best performing filters.

During each generation, a certain number of filters are created and tested. A filter is created in one of three ways.

Random Generation: A filter may be randomly generated, as it was in the beginning.

Mating: Two filters are chosen from the pool of best filters. Then, individual rectangles from these filters are randomly chosen and added to the new filter, until the maximum coverage of 1% is reached.

Perturbation: A filter is chosen from the pool of best filters. Then, individual rectangles from this filter are randomly chosen and perturbed by changing by coordinates and color of the rectangles within a certain bound.

The way a certain filter is generated is again random, but weighted higher towards mating or perturbation. For mating and perturbation, when a filter is chosen from the current pool of best filters, they are weighted by how close to the top of the list they are.

```
save_amt = int(num_filters/10)
top_filters +=
    top_filters_overall[-save_amt:]
if iter%2 == 0: # 50% of time
    func = lambda x : x.dist
elif iter%4 == 1: # 25% of time
    func = lambda x : x.start_dist
else: # 25% of time
    func = lambda x : -x.target_dist
top_filters.sort(key = func)
top_filters_overall =
    top_filters[-num_filters:]
```

Figure 10: Selection of top filters at the end of each generation

6.2 Results

Figure 11: Image causing misidentification of subject 0 as subject 14, created via mating, along with unedited images of subject 0 and 14

With this method, I was again able to achieve misrecognition and misdetections. I was also able to eventually achieve a misidentification. Here, subject 0 under lighting q and view a has an average distance of 0.359 from images of herself and 0.346 from images of subject 14[3].

The amount of iterations I attended was limited by the scope of the project. Many more iterations of this experiment were possible, varying values like the number of transformations per generation, per pool, the likelihood of any generation method, range of rectangle size, range of perturbation value, weight of previous filters, and percentage of image covered, as well as how filters are sorted, and what the starting image and target subject are. With more iterations, as well as experiments that are ran for a longer period of time, I think it would be possible to achieve more failures of all three kinds, but especially misidentification. Ultimately, however, I was able to achieve success in this small scope.

One observation of note is that I had a really hard time achieving misrecognition and misdetections when less than 0.5% of the image was covered. While, ultimately, I was able to achieve them, it was less likely, and no misidentifications occurred.

[3]video of progression by generation: `https://drive.google.com/file/d/1aUoYu4q63-S_S7_Z2KiNXUF6qH8F2isr/view?usp=sharing`

Another interesting observation was that generating new images via mating outperformed all other methods. Each generation, the best transformation was recorded and saved, applied to the image, as well as which method generated it. While this makes sense for random generation, this surprised me for perturbations. In further experiments, it would be interesting to see other methods of generating new transformations, as well as combining the perturbation and mating methods.

At the end of each generation, the top few filters of all time and all filters from this generation are sorted by performance. In this case, the goal is maximizing the difference between the distance to the target subject and the distance to the starting subject, until it is negative (i.e. there is a larger distance to the starting subject) and a misidentification occurs. To do so, however, we must both maximize the distance from the starting subject and minimize the distance from the target subject. After some experimentation, the method I landed on was sorting by the difference in distances on 50% of generations, sorting by distance from the starting face on 25% of generations, and sorting by negative distance from the target face on the remaining 25% of generations. Then the top performers are kept as the pool of best filters for the next generation.

Figure 12: Best performing filters of over generations

7 ANALYSIS

7.1 Takeaways

Unsurprisingly, misidentifications were the most difficult to achieve. This is also good news, as these could be the most dangerous and specifically harmful. They were most rare, and only occurred with the most unrealistic transformations. Additionally, they required time specifically spent targeting a specific pair of subjects. However, the fact that they were potentially possible at all was encouraging.

The other failures were much easier to achieve and much more generalizable. The results were overall expected - glasses caused worse performance, as did unusual lighting, and transformations were able to achieve these attacks, at a rate generally related to how strong the transformations were.

The success of rotations at achieving these failures was one of the most interesting results, as these transformations were simple and realistic. It was especially interesting that rotating near 90 degrees tended to perform somewhat worse than rotating 180 degrees.

7.2 Threats to Validity

The biggest threat to the validity of this project is whether or not the transformations applied are actually valid concerns for evasion attacks.

As mentioned throughout the paper, I ultimately only considered transformations decided to be mild enough to keep the image recognizable by the human eye. This means that, if a face is not recognized at all or not recognized as a certain person by face-recognition, it is truly an incorrect classification. However, this was decided somewhat subjectively by me, and could be disputed. Whether any of the filters keep the face reasonably recognizable is not a statement of fact, rather a judgment on my part.

Secondly, and perhaps more importantly, is whether or not these images are reasonable candidates for evasion attacks. This means that these images may reasonably be used as inputs for face-recognition. Other papers do a much better job of this than mine[9]. By my own admission, the random rectangle transformations performed in the last experiment are unrealistic, and would unlikely be able to apply to the real world. Other filters, however, are generally realistic. This includes darken, lighten, contrast, blurweak (to some extent), and some rotational filters. The facial feature aware filters - darkenshadowsfast, darkenshadowsslow, darkeneyes, and clown - are generally meant to be close approximations of scenarios that can occur in real life by wearing makeup or standing in harsh lighting, but are ultimately not ideal.

Transformations that are unlikely to occur in real life are less dangerous when it comes to evasion attacks. However, I do still consider them to be failures of face-recognition, but perhaps to a lesser extent. Additionally, these unrealistic filters succeeding in attacks indicates that attacks are possible, and more realistic filters can be created and tested, maybe even using the results of these.

7.3 Repairs

The most obvious and simplest repair to take away from this report is to consider rotations in face-recognition. The face-recognition library already has an option for this, a `num_jitters` parameter can be added the `face_encodings` function to transform the inputted image some number of times in some variety of ways (rotating, translating, zooming) before returning an encoding. Ultimately, I didn't run experiments over different values for `num_jitters`, but I am also surprised rotations aren't automatically considered.

The strongest repair is retraining, with more variations and transformations of images considered. Beyond that, specific uses of face-recognition can build their own classifiers with specific considerations trained on the encodings returned by `face_encoding` (as suggested by the documentation add linl alters). This may be a good idea in some specific contexts, especially when some transformations are particularly likely (blurring, harsh lighting, etc).

8 FURTHER WORK

The experiments performed in this project were a baseline, minimal exploration of all that is possible. Below are all possible ways that this project can be expanded on, in no particular order.

Expanding the range and length: In general, most of these experiments could be performed for longer and on a larger scale and for more iterations. This is especially applicable to the misidentification experiment and testing some of the more time-intensive filters.

Varying filters: While the rectangle filters and rotation filters were tested more extensively, most filters were tested on a baseline configuration, or a couple variations at most. More experiments can be done, focusing on one transform at a time.

Testing num_jitters: The performance of face-generation can be analyzed with the built in transformations added on with `num_jitters`.

Applying to real life: Some filters can be applied to and tested in real life, like the clown filter and darken shadows filters.

Varying genetic algorithm: The genetic algorithm used in the misidentification experiment can be altered to consider which performs best.

Expanding the range and length: In general, most of these experiments could be performed for longer and on a larger scale. This is especially applicable to the misidentification experiment and testing some of the more time-intensive filters.

Using other datasets: While MUCT was judged to overall be the best choice, a variety of other datasets can be used, particularly ones that provide more insights to subject demographics.

Analysis by subject demographic: More work can be done analyzing how each transformation affects the performance of subjects of different genders and ethnicities. This is especially important if we want to justify the use of face-recognition in potential law-enforcement environments, where being cognizant of bias is of utmost importance.

Differential testing / testing more facial recognition libraries: To get around manually making sure transformations remain recognizable to the human eye, the differences between results of different facial recognition libraries can be studied, and specific cases where the libraries disagree can be studied. Additionally, similar testing to the testing I performed on face-recognition can be performed on other libraries.

9 CONCLUSION

The face-recognition library for Python is a facial recognition library that performs well but is vulnerable to evasion attacks where a face is not found or not recognized as the correct person. Most cases where these failures occur are expected, i.e. with blur or unusual lighting, but ultimately mean that face-recognition, like other facial recognition libraries should be used cautiously and their potential failures should be kept in mind when using their results.

References

[1] Analyzing pypi package downloads. `https://packaging.python.org/en/latest/guides/analyzing-pypi-package-downloads/`.

[2] Face databases. `https://web.mit.edu/emeyers/www/face_databases.html`.

[3] P. N. Bellhumer, J. Hespanha, and D. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Special Issue on Face Recognition*, 1997.

[4] Gaudenz Boesch. What is adversarial machine learning? attack methods in 2023, Jan 2023.

[5] Adam Geitey. face-recognition. https://github.com/ageitgey/face_recognition, 2018.

[6] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[7] S. Milborrow, J. Morkel, and F. Nicolls. The MUCT Landmarked Face Database. *Pattern Recognition Association of South Africa*, 2010. http://www.milbo.org/muct.

[8] Webster Moswa. Facialrecognitionforpolice. https://github.com/Moswag/FacialRecognitionForPolice, 2020.

[9] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security*, October 2016.

[10] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest. *Proceedings of the 40th International Conference on Software Engineering*, 2018.

[11] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018.