Attacking Voice Authentication Systems

Eshaq Jamdar
College of Science: Computer Science
San Jose State University
San Jose, California
eshaq.jamdar@sjsu.edu

***Abstract*—Voice Authentication is a common form of authentication used primarily by automated phone services, such as banking, as a second form of authenticating users. This form of authentication is rife with many common attacks, such as replay attacks, impersonation attacks, and the growing threat of deepfake audio replicated from the voice of the person whose account is attempting to be accessed by an attacker. Many solutions have been presented to defend against these types of attacks, such as using "Voice Pops," which aims to identify the unique phoneme pronunciations of a person during enrollment. While this paper presents a promising solution to unique voice authentication, it fails to show how well the system goes up against other types of attacks. This paper seeks to replicate the VoicePop system outlined by Wang et al. [2], test how well it goes up against unseen data samples, and how well the system can withstand data poisoning attacks. The VoicePop system replicated gets an average of 69% accuracy with another dataset that includes other types of attacks not covered by Wang et al.'s paper. In addition to this, data poisoning by label flipping has an unintended effect on regularization and the prevention of possible over-fitting when using a smaller dataset. Lastly, modifying spoof audio to add in synthetic VoicePop proves to be an effective method that reduces overall model accuracy to under 14%.**

***Index Terms*—voice authentication, data poisoning, biometrics, SVM, machine learning**

I. Introduction

Voice authentication (VA) is less commonly observed but is well-used within biometric security. Its most common use is over communication automated systems, such as banking over the telephone, as a secondary form of authentication. According to Zhou et al., [1], two standard attack vectors VA systems face are physical and logical attacks. Physical attacks manifest themselves as replay attacks, where an attacker has a recording of the victim's voice and replays the voice sample over a speaker towards the microphone input. Logical attacks, conversely, result from Text-to-Speech (TTS) or machine-produced voice samples that are mistaken as legitimate. Though Zhou et al. do not mention which attack vector is the most common, their focus on physical-based attacks would seem to imply that this is a common attack vector. While physical-based attacks are very plausibly used, over the past decade, the emergence of deepfake voices has added another layer of complexity regarding attacking these types of systems.

From reading many different types of literature on defending against spoofing attacks on VA systems, many papers suggest how to defend against physical-based replay attacks. One such paper is Voice Pop by Wang et al. [2], which works upon the ideas of Mochizuki et al. [4], which is the idea of using "pop-noises" to detect if a voice belongs to the claimed identity. This is done by identifying the unique way in which an individual pronounces certain types of phonemes. For example, when pronounced, the letter 'p' produces a distinct "pop" due to how the human tongue and lips push air out the mouth. The idea is that phonemes within a given phrase can be identified within audio, and the intensity of these pop noises per individual can be mapped out and used to authenticate a user. While a replay attack, in theory, has these pop noises present within the audio captured, playing these sounds over a speaker can make it harder for these pops to be recognized by Wang et al.'s VoicePop system. However, solely relying on VoicePop detection is not enough, and Wang et al. [2] expands upon Mochizuki et al.'s work by using Gammatone Frequency Cepstral Coefficient (GFCC) to scrutinize voice samples further. While some replay attacks can be quickly rejected with the initial VoicePop checks, some voice samples can confuse some audible or background noises for Pop Noises. To counter this issue, GFCC filters down the audio to get the finer details of bursts of air being spoken into a microphone. This, in essence, captures the naturalness of a human speaking into a microphone.

Wang et al. are [2] proposed architecture, while reporting a high accuracy of 93.5% to detect replay attacks, has many issues that bring into question how well the system stacks up against other types of attacks or even its robustness to data poisoning attacks. Because VoicePop is trained on replay attacks and impersonation attacks, it questions how accurate the system is to logical attacks. In addition, their testing of the system only involved 18 participants, with a majority being men, which brings into question how well the dataset truly is over larger datasets. For this reason, this paper aims to recreate the VoicePop system created by Wang et al. and investigate how this system stacks up against both replay and logical-based attacks. In addition, this paper also aims to introduce data poisoning attacks on data being fed to the recreated system and how much this affects the system's accuracy.

II. VoicePop: Detailed Explanation

A. Design of the System

From a high level, each part of the Wang et al. [2] VoicePop system can be summarized as the following

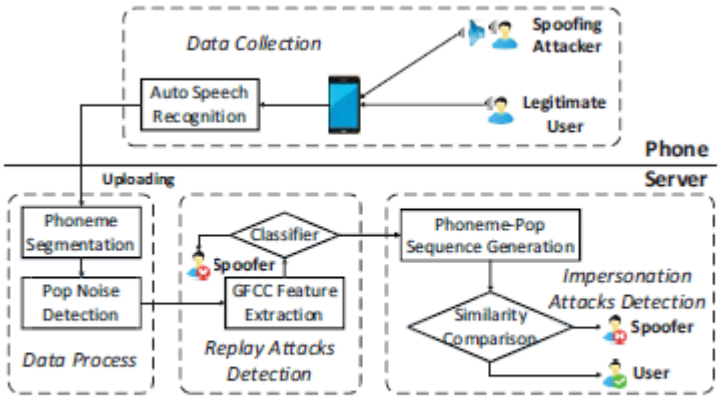

Fig. 3. The architecture of VoicePop.

Fig. 1. Architecture of VoicePop by Wang et al. [2]

*1) **Data Collection**:* In this phase, the system collects a voice sample from a legitimate user or a spoofed replay attack from an attacker. A user says their passphrase to the smartphone, and this sample is sent to the auto speech recognizer to identify whether the passphrase can be recognized.

*2) **Data Process**:* Once the passphrase has been correctly identified, the audio segment is then sent to the authentication server. The first part of this process is phoneme segmentation, where phonemes are extracted during the audio clip and where the pops occur in the audio. For example, if a phrase is ”pot,” the segmentation would output a vector containing [1,0,1] as p and t are considered phonemes. From there, pop noise is detected in the audio. Here, the audio file is transformed via a Short Time Fourier Transform (STFT); this is used to capture the short burst of energy that is expelled at the time the pop noise occurs in the audio. Pop noises, according to Wang et al., happen in the 100 Hz range, and the STFT helps narrow the low frequencies that occur below or equal to 100 Hz. The ultimate result of this step is to return the timings of all pop noises that occurred with respect to a calculated threshold for that specific audio file.

*3) **Replay Attack Detection**:* In this detection stage, the pop noise timings are sent to a GFCC extraction process. From the pop frames extracted, each part of the audio where the noise occurred is segmented at the start and end of the clip with some overlap of background noise. This segment is then sent through a log function to normalize any spikes in audio and get a smoother shape of the burst of energy. Once this is done, a mean of the overall magnitudes is taken to represent one of the feature vectors to the SVM. In addition, two other features are calculated: the timing changes from pop noise to background (delta 1) and the sudden shift in energy in the recording (delta 2). Delta 1 will indicate how the transition from pop noise to background (and vice versa) occurs by displaying the magnitude spike in energy being high during the beginning of the pop noise occurrence and the magnitude energy decreasing when going back to background noise. Delta 2, on the other hand, calculates the rate of change that occurs during these two periods. A genuine pop noise will have both of these values at high rates as the voice captured can capture these more significant details because a microphone can pick up these harsh speech transitions naturally. A replay attack, on the other hand, will have much lower values for both deltas as the noise captured by the microphone cannot pick up on these details due to the audio sample being played from a speaker, which causes these details to be silenced or less pronounced. Once the GFCC features are calculated for the audio input, the features are fed into an SVM classifier to identify if the audio sample is from a human or is a replay attack.

*4) **Impersonation Attack Detection**:* At the final stage, the audio input so far has been classified as coming from a human, however the question remains: is this an impersonation attack? Impersonation in this context refers to a person who is not the authenticated user mimicking the voice of the legitimate user. In the phoneme segmentation part of the Data Process, the positions of pop noises were denoted via a simple 1D vector. Two forms of alignment with the stored and input sequences are done. The first is the Pearson Correlation Coefficient, which computes the linear relationship between both vectors. The output is the probability of the two vectors being similar to one another (closer to 1 being close alignment and close to 0 being none at all). As a secondary form of alignment, the vector is then compared against stored user samples using a contact ratio to confirm if the sequence of pop noise presence lines up with the stored sequence. From here, the total number of matches is then divided by the total number of characters in the sequence to give us a final percentage of the alignment. Once these two numbers are computed, these numbers are compared against their threshold, and if they exceed or are equal to this threshold, then we have an exact match, and the person is fully authenticated.

B. Results

The setup of Wang et al. [2] collected data from 18 participants (13 males and five females) who were of college age. Enrolling each person required them to utter their passphrase (of varying length) five times into a microphone and then asked them to authenticate ten times to confirm that the system was working.

To test for replay attacks, they recorded the audio of the participants with a professional microphone from a distance, and then the other five were simple smartphone microphones. Then, a replay attack was tested ten times per user in the system.

As a comparison for a baseline, Wang et al. used Voice-likeness detection based on the pop-noise detector by Mochizuki et al. [5], which Wang et al. bases their paper around, although indirectly acknowledging it within the results section. The main difference between Mochizuki et al. and Wang et al.’s implementations of VoicePop comes down to the use of impersonation detection along with the use of GFCC as a form of liveliness detection.

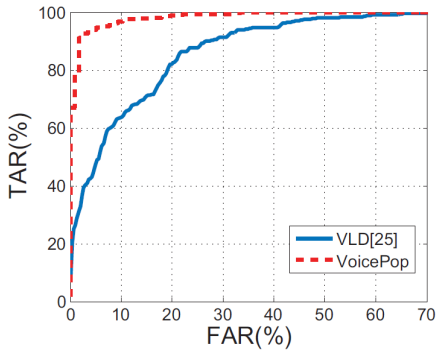

Fig. 2. Receiver Operating Characteristic from Wang et al. [2]

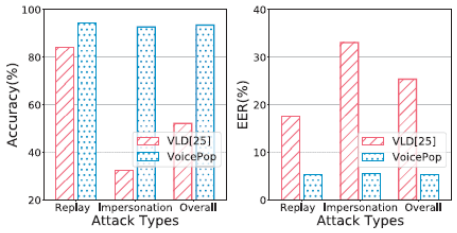

Fig. 10. Overvall accuracy and EER.

Fig. 3. Accuracy of VoicePop from Wang et al. [2]

Regarding the True Accept Rate (TAR) versus False Accept Rate (FAR) from Figure 9, VoicePop received a 93% TAR, while the number of FAR was below 2% according to the authors. Figure 9 supports this because, at a low percentage of FAR, the TAR quickly rises to a very high TAR and remains stable as the FAR increases. An exact figure for TAR and FAR is not given for VLD, but observing the graph at the same low percentage, the TAR rate is much lower and takes much more FAR over to reach a stable TAR, indicating that VLD struggles with replay attacks.

Figure 10 covers both the overall accuracy and the Equal Error Rate (EER) for both models. These figures cover replay attacks, impersonation, and overall model performance for their respective categories. Regarding VoicePop's overall performance (both replay and impersonation), the authors report an overall accuracy of 93.5%, with an EER of 5.4%. Compared to VoicePop, VLD has an overall accuracy of 52.2% accuracy with a much higher EER of 25.4%. Observing Figure 10 for VLD, we see it has a relatively high accuracy for detecting replay attacks compared to defending against impersonation, and the EER is much higher in both categories compared to VoicePop (especially pronounced with Impersonation). This is not shocking as VoicePop has this built-in mechanism to detect impersonation, and better accuracy with replay can be explained by the liveliness detection employed.

C. Potential Issues

Despite the authors of VoicePop showing a model with relative high accuracy, many questions remain about how well this system can withstand scrutiny.

1) Participant Size: The total number of people tested for their experiments was 18: 11 men and eight women. Such a small number of people raises questions about how well this data can withstand generalization, and the authors do not do any form of validation testing to confirm their findings from outside data. Relying solely on this data to train and not confirming that this data can withstand generalization raises questions about how truly effective this system can be.

Aside from the fact that the population is small, this brings into question how well this type of system captures how men and women exert VoicePops into the microphone. In addition to this, the age group is people in their twenties. Relying on this age bracket can also bring into question how well this system fares against other age groups, especially among older people whose voices may be softer over time. The last point of criticism is how well this system can handle accents from people whose native language is not English. While the demographic data is not reported in the paper, the question arises of whether accents influence how VoicePops are exerted and show up on chronographs.

2) Different Types of Attacks: Since this system is being tested with physical-based attacks, the question arises: how well can this system withstand more logical-based attacks? With the rise of DeepFake voice synthesis, it has never been easier than ever to recreate a person's voice and make them say anything. Assuming a person does know the passphrase but does not have access to a recording of the victim saying it, a logical attack using DeepFake voices can prove to be viable, and many of these DeepFake synthesizers can make these computer-generated voices much more natural sounding. Such a threat brings into question how robust the GFCC extraction truly is.

3) Data Poisoning Attacks: Due to the use of SVM training to classify audio as real or fake, data poisoning techniques start to become a problem. If a portion of audio samples fed into the SVM classifier that are labeled as fake but are modified to add in an extra sine wave to simulate the presence of abrupt pop noises in audio, can this reduce the accuracy of the overall model? In addition, can the sample principle be applied by modifying background noise to appear more natural in the presence of fake pop noises?

III. Recreating VoicePop

A. Dataset

In the VoicePop paper, the dataset Wang et al. [2] collected was never published. For the purposes of this paper, the ASVSpoof 2019 dataset was used as a substitute. ASVSpoof 2019 [4] was a competition in which participants were tested to develop a model that can differentiate between actual (bonafide) or fake (spoof) audio samples. The types of spoof audio file attacks used are the following:

- Replay Attacks
- Deepfake Audio
- Text to Speech Audio

The dataset labels do not differentiate between these types of attacks, as the idea is for the participants to build models that can perform well generally and not for any particular type of spoofing attack.

For this paper, most of the VoicePop model features have been replicated. However, the last part of VoicePop, Impersonation Attack Detection, was not recreated due to the dataset not including this form of attack.

ASVSpoof 2019 contains two labeled sets: train and evaluation. The breakdown for each set can be summed as:

- Train: 2580 Bonafide (Real), 22,800 Spoof
- Evaluation: 7355 Bonafide (Real), 63,882 Spoof

The publishers intentionally designed both datasets to be imbalanced because this is in line with their philosophy of designing models that perform well against a variety of spoof attacks.

B. Replicating VoicePop

Since the creators of VoicePop did not provide a repository for any of the code their system ran, this meant having to recreate it. A good starting point that was used was Voice Authentication and Face Recognition by Mohamad Merchant [5]. This code base provided a notebook environment where users could enroll in their system and authenticate within the system. The way it authenticates audio is based on Mel-frequency Cepstral Coefficients (MFCC), which helps turn speech into text. The code was a good base for understanding how voice authentication can be done and laid a base for how to prototype a similar system.

The main logic in Algorithm 1 demonstrates the Data Process of Pop Noise detection along with GFCC extraction of features to enable liveliness detection. For every audio file, pop noise timings are extracted, and these timings are then sent to GFCC filters to extract the essential features:

- gfcc mean: average of pop noise magnitudes within the audio
- delta 1: changes of audio energy from the start and ending of a pop noise
- delta 2: how much the pop noise signal changes over the entire signal

Once all GFCC features are extracted, the training process can begin. According to Wang et al. [2], the classifier that uses the GFCC features is an SVM classifier. Since the problem is a binary classification, where audio is either a replay attack (spoof) or real, SVMs are a clear choice in the matter. The original VoicePop paper worked on a much more balanced dataset that had an equal amount of real and fake samples being trained on their system. However, the dataset is heavily skewed towards spoofs, and to balance out the imbalance, SMOTE is used to increase the minority class of real audio in the dataset. In addition to increasing the presence of the minority real class, a balanced weight configuration was used to ensure the majority spoof did not overpower the training process. Lastly, a GridSearch was used along with the SVM to identify the proper set of hyper-parameters to use for the best possible model via cross-validation of five.

Algorithm 1 Main Logic of Feature Extraction

Require: Dataset path `path`, label dictionary `labels`
Ensure: GFCC features `gfcc_stack`, associated labels `labels`

```
Initialize gfcc_features as an empty list
Initialize associated_label as an empty list
Shuffle the items in labels and store in shuffled_labels
for (fn, label) in shuffled_labels do
    Construct voice_path as path/fn+".flac"
    if voice_path exists then
        Load audio and sample rate sr from voice_path using librosa.load
        Extract pop noise pop_noise from audio and sr
        Extract GFCC features gfcc from audio, sr, and pop_noise
        if gfcc is not None then
            Append gfcc to gfcc_features
            Append label to associated_label
        end if
    end if
end for
if gfcc_features is not empty then
    Stack gfcc_features vertically into gfcc_stack
    Convert associated_label to an array
    return gfcc_stack, labels
else
    return Empty arrays for both gfcc_stack and labels
end if
```

C. Results

Two sets of tests were run: a training of the entire dataset and a training where the maximum number of real (2580) and the same amount but for fakes were used to create a test with a more balanced dataset from the start. Each test was run with the evaluation dataset, which is much larger than the training dataset (7335 real and 63882 fake).

1) Full Training: Observing Figure 4 and the probability distribution of audio being fake or real, using a threshold of 0.61 for the evaluation set aligned the most with the dataset, mostly rejecting and accepting only a small number of the dataset. However, to observe the accuracy of the recreated model, Figure 5 shows the Confusion Matrix for how well the model aligns with the evaluation dataset. In the context of identifying fakes, the model presents a 69.29% overall accuracy in identifying fakes with a True Positive Rate of 68.6% samples being correctly identified as fake. In regards to how accurate the system is at identifying actual samples,

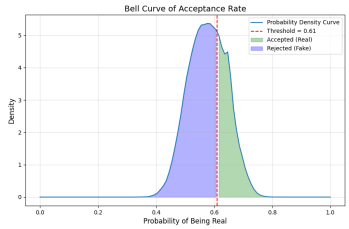

Fig. 4. AUC for Probability of Real (Full Training)

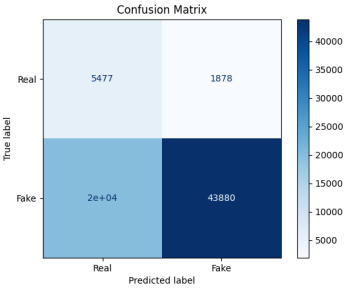

Fig. 5. Confusion Matrix (Full Training)

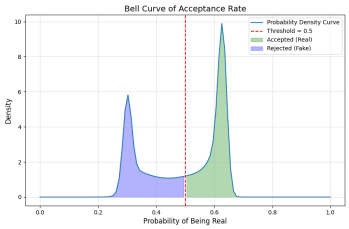

Fig. 6. AUC for Probability of Real (Even Training)

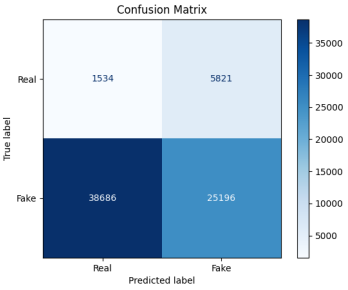

Fig. 7. Confusion Matrix (Even Training)

with a True Negative Rate (TNR) of 74.4%, the system tends to identify most of the actual samples.

The model, with full training, has an issue identifying fake samples as real. Over 20,000 samples are considered real, and a plausible explanation is that since the dataset includes logical attacks, the VoicePop algorithm focuses exclusively on physical-based attacks. However, this cannot be definitively said, as further testing with a dataset solely focused on physical-based attacks is needed to fully confirm this suspicion.

2) Even Training: The probability distribution shows a much more bimodal distribution, where the training caused a clear line of separation between real and fake. The threshold chosen for this test was 0.5, as this clear line of separation makes it easier to draw a clear line of division.

For the purposes of exploring a more balanced training set, the max number of objective in the training set, 2580, is set for both the real and fake limiters. This leaves the code with a total of 5160 labels total, unlike the previous 22800 from the whole train set. In this test, the model has a total accuracy of 37.52% in detecting fakes, with a TPR of 39.44% of fakes being classified as fake and 60.56% of FNR of fakes being classified as authentic. In regards to detecting actual samples, with a TNR of 20.86%, the balanced dataset suffers much more in identifying actual samples from fake ones and a much higher FPR of 79.14%.

Under a balanced dataset with a 0.5 threshold, the accuracy of the model suffers quite a lot in identifying real and fake. From further testing, a higher threshold of 0.6 results in more fakes being classified correctly, but the number of reals being classified correctly continues to decline. Since the number of fake samples to train on is limited by the total number of real ones to test on, and the samples are randomly chosen (meaning we cannot control for a logical or replay-based attack), the model suffers from a form of under-training, or somewhat imbalanced within the confines of the training. A more realistic approach is to stick with the complete training of the model with the entire dataset and fine-tuning to improve the model.

IV. Data Poisoning VoicePop

Since we have established a baseline for the system accuracy, where on a full training, 74.4% of real samples are classified as real and 68.6% of fakes are classified as fakes, this opens up the ability to test how well the system can withstand data poisoning attacks. In terms of what type of attacks we can test out, the following have been chosen

1) Label Flipping: This method, according to Sagar et al. [7], label flipping is a straightforward method that can target a certain amount of labeled data within a population and negate the labels. These types of attacks are very commonly used with Binary Classification problems, making it a good candidate for VoicePop.

2) Adding Synthetic PopNoise: According to Ramirez et al., [8], an SVM can be targeted by altering the data of training data in the hopes that these modifications are enough to disrupt the classification process. Considering there is an abundance of fake samples during training, a small subset of the population can be taken, added in some form of synthetic pop noise, and then trained with this modified data.

The idea with algorithm 2 is that since PopNoise is a sudden burst of energy that is below or equal to 100 Hz on lower bands, a subtle sine wave can be added to resemble PopNoise in a fake audio sample. The configuration of the algorithm dictates that the amplitude and frequency values of the signal are noticeable enough to be picked up by the PopNoise and

Algorithm 2 Modify Pop Frequency in Audio Signal

Require: Audio signal and sampling rate
Ensure: Modified audio signal with an amplified pop noise

1: Set amplitude for pop noise = 0.5
2: Set frequency for pop noise = 90
3: Calculate the duration of the audio signal
4: Create a time vector based on the duration and signal length
5: Generate a sine wave representing the pop noise
6: Add the sine wave to the audio signal and normalize it
7: **return** Modified audio signal

GFCC extraction to take place. There is an assumption that there are some pop noises in fake audio samples that occur even in fake audio clips, so the frequency and amplitude are at moderate levels throughout the entire audio clip.

V. Data Poisoning Results

1) Label Flipping: Two tests were employed to observe the effects of label flipping: 20% poisoning and 70% poisoning. Each test will randomly obtain a label and flip it every time it is running. For simplicity's sake and due to the long training of the entire dataset, the setup for this test is the maximum of 2580 real from the dataset and 2580 fakes that are randomly obtained due to the abundance of fake samples in the training set. For the sake of fair comparison, Figures 6 & 7 will be the standard baseline for comparison, as all the poisoning tests use the same number of fake and actual samples.

A. Results

Two sets of tests were run: a training of the entire dataset and a training where the maximum number of real (2580) and the same amount but for fakes were used to create a test with a more balanced dataset from the start. Each test was run with the evaluation dataset, which is much larger than the training dataset (7335 real and 63882 fake). Figure 8 shows the overall acceptance threshold for this model, which is not too far off from Figure 6. It shows a clear line of division for fakes and reals, similar to Figure 7 (even training, no label flipping). It should be noted that the threshold was slightly lowered by -0.2 compared to the 0.5 seen in Figure 6, as this was the reported AUC score provided at the end of training.

Figure 9's results are very similar to one another but with some inaccuracies. Calculating accuracy based on Figure 9, poisoning 20% of the samples yields an overall accuracy of 37.21%, a -0.31 difference from the even training in Section III Part II. Furthermore, the TPR of fakes being classified as fakes is 39.04%, a difference of -0.4%. In regards to FNR and misclassified fakes, Figure 9 reports 60.96% of fake samples classified as accurate, an increase of 0.4%. From a preliminary glance, 20% of this population gave a slight decrease in overall accuracy as the results still mostly line up with the results of no poisoning with Section III Part II. In contrast to fake samples suffering from accurate identification, Figure 7 reports a TNR of actual samples classified as accurate at 21.32, a 0.46

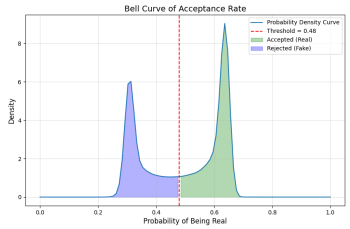

Fig. 8. Acceptance Rate of 20% Poison

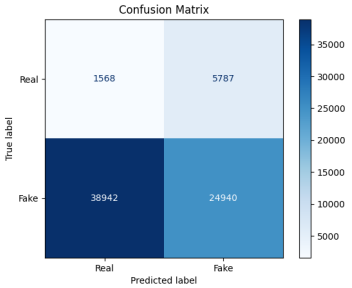

Fig. 9. Confusion Matrix with 20% Poison

increase over Section III Part II. In addition, the FPR for real classified as fake reported 78.86%, a 0.28 increase over the previous non-poison version.

It would seem that the data poisoning at 20% is very subtle, and even with the increases in identifying accurate samples, the difference is very small. To reinforce this finding, the threshold was decreased from 0.5 with no poisoning to 0.48 with poisoning. This small decrease in threshold could explain these slight changes in the numbers and could suggest that a small data poison may not be as effective in this small percentage.

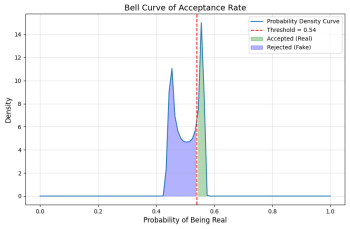

Fig. 10. Acceptance Rate of 70% Poison

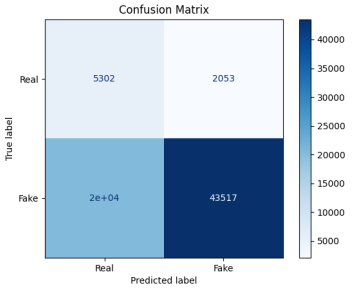

Fig. 11. Confusion Matrix with 70% Poison

Figure 10 shows an interesting trend: Most of the samples are classified as fake, with a smaller percentage considered real. This would indicate a reflection of the actual dataset and could suggest performance along the lines of the full training seen in Figure 4 despite being trained on only 5160 samples for these runs.

Figure 11 reports an overall accuracy of 68.88%, an increase of 31.36% of accuracy from Section III Subsection 2. In addition, the TPR for fakes classified as fakes is 68.51%, an increase of 29.43% for identifying fakes correctly, with 31.49% FNR of fakes classified as authentic. From observing the accuracy of identifying fakes, the baseline from Section III Subsection 2 under-performs, and the flipping of labels caused an increase in performance similar to the complete training of the model with Section III Subsection 1. Furthermore, classification of actual samples also went up, with classifying reals correctly, the TNR is reported as 72.09%, a 51.23% increase in correctly identifying samples as accurate. The system reports an FPR of 27.91% of classifying actual samples as fake, decreasing from the baseline by 50.67%.

In terms of label flipping, this had the unintended effect of causing accuracy to go up. The possible reasoning behind this is due to using 5160 samples total instead of the entire dataset (so that testing can happen faster). The model used here comes very close to the performance of the accuracy of full training with no label flipping. This behavior could describe some form of implicit regularization, described by Song et al. [9]; implicit regularization is when noisy labels are present; the model can overcome generalization issues, such as over-fitting, by introducing stochasticity. The randomness of flipping labels helps the model rethink some of its predictions. Introducing noisy label flipping over a more significant part of this limited dataset could have possibly prevented over-fitting, which was likely the case with the original testing done in Section II Part 2.

1) Pop Noise Poisoning: For the purposes of this test, 20% of audio samples labeled spoof were randomly selected and modified according to Algorithm 2.

Observing Figure 12, the distribution of probabilities is mostly centered around 0.44 and shows a strong bias towards real samples being accepted over fake. The behavior exhibited here is interesting, and Figure 13 provides much better insight into how well this attack worked.

Calculating the accuracy for Figure 13, the model's accuracy has fallen to 14.01%, with a TPR of fakes classified correctly being 4.11%, while the 95.89% FNR is reported for fakes classified as real. In addition to this, real samples being classified correctly received a TNR of 100%, while the number of False positives classified correctly was scored 0%.

Introducing pop noise throughout spoof samples across the entire time segment caused the model to classify many fakes as genuine. The number of fakes being classified as accurate is almost a majority of faked labeled samples, and only a smaller number of fakes were correctly identified as fake. Considering that this was only 20% of the dataset being poisoned if a larger

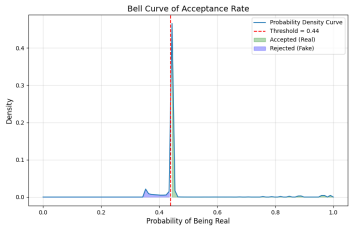

Fig. 12. Pop Noise Poison Attack Acceptance Rate

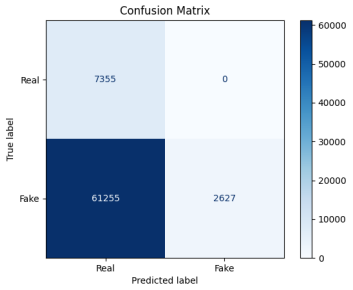

Fig. 13. Confusion Matrix with 20% Pop Poison

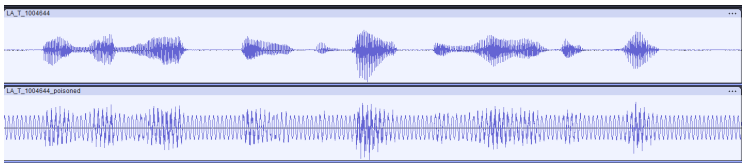

Fig. 14. Top: Unpoisoned Sppofed Audio File, Bottom: Same File But Poisoned

poison population was utilized, it is likely that every audio sample would be treated as accurate.

What makes this attack successful can be observed in Figure 14. Before the poisoning, there was not much of a Pop Noise presence, and there was not much of a pattern that could be picked up. However, with the bottom graph, we see that our synthetic PopNoises can be observed throughout the entire audio file. The idea is that if you make a very noticeable pattern that can be picked up as PopNoise during extraction, this would be enough to introduce a noise that will confuse the model and ultimately cause the results seen in Figures 12 and 13.

VI. Conclusion

To conclude, Voice Authentication models are critical forms of authentication, primarily used within banking and adopted within the world of IoT devices, such as Amazon's Alexa. With such a rise in Voice Authentication, the question of how well these models can withstand attacks from a training perspective and attackers attempting to gain access to individual accounts. This paper managed to recreate a proposed way to get around replay attacks, with an overall accuracy of 69.29%. Although the accuracy did not match the accuracy noted from VoicePop by Wang et al. [2], this can be attributed to the

dataset used, as this dataset included other types of attacks that VoicePop did not consider to defend against.

In addition to recreating this model, two poisoning attacks were made upon this recreated system: flipping of labels and synthetic VoicePop insertion. The findings of label flipping show that if the percentage of labels flipped increases, there is an unintended effect of system accuracy increasing. Within the test, this helped improve the accuracy of a system that was trained on a subset of the dataset, in this case, an even split of 5160 audio files. The accuracy went from 37.52% overall accuracy to an accuracy reaching 68%, which was much closer to the accuracy seen when training on an entire dataset. Though this attack did not intend to improve accuracy, it does provide a good improvement strategy for training these types of systems when working with limited data.

Lastly, an attack on the dataset by directly modifying 20% of randomly selected spoofed files was employed. This attack's findings show that inserting synthetic pop noises over the entirety of individual audio files labeled spoof provided enough noisy data to cause the model to degrade in classification and ultimately resulted in a system accuracy of 14.01%. This type of attack proved to be a determent to the overall system and could have possibly allowed other types of non-replay attacks to be classified as accurate due to this noisy data. The result of such an attack provides insight into how robust systems like VoicePop are to these types of data attacks. This is an important finding, as many Voice Recognition systems that can differentiate between human, bot, or replay attacks can be vulnerable to these issues depending on where they receive such data.

For further steps of refinement, additional testing of synthetic GFCC features being added to data can give more insight into how well these types of systems can handle multiple types of attacks from many different angles. In addition to this, further improving the VoicePop recreated model is a necessity due to the lack of accuracy matching. To confirm if the model recreated is a close 1:1 match, a dataset that is comprised solely of replay attacks and impersonation attacks should be tested to verify that the accuracy is close to the paper's findings. One last attack that can be developed is the idea of using diffusion models to recreate a person's pop noise sequence in order to refine further recreating any passphrase, such as a victim recreating their voice passphrase.

REFERENCES

[1] J. Zhou, T. Hai, D. N. A. Jawawi, D. Wang, E. Ibeke, and C. Biamba, "Voice spoofing countermeasure for voice replay attacks using deep learning," *Journal of Cloud Computing: Advances, Systems, and Applications*, vol. 11, no. 51, pp. 1–14, 2022. [Online]. Available: https://doi.org/10.1186/s13677-022-00306-5

[2] Q. Wang, X. Lin, M. Zhou, Y. Chen, C. Wang, Q. Li, and X. Luo, "VoicePop: A pop noise-based anti-spoofing system for voice authentication on smartphones," in *Proceedings of the IEEE International Conference on Communications (ICC)*, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2007.08199.

[3] X. Valero and F. Alías, "Gammatone cepstral coefficients: Biologically inspired features for non-speech audio classification," *IEEE Transactions on Multimedia*, vol. 14, no. 6, pp. 1684–1692, Dec. 2012. [Online]. Available: https://doi.org/10.1109/TMM.2012.2199972

[4] J. Yamagishi, M. Todisco, M. Sahidullah, H. Delgado, X. Wang, N. Evans, T. Kinnunen, K. A. Lee, V. Vestman, and A. Nautsch, "ASVspoof 2019: The 3rd Automatic Speaker Verification Spoofing and Countermeasures Challenge database," *University of Edinburgh, The Centre for Speech Technology Research (CSTR)*, 2019. [Online]. Available: https://doi.org/10.7488/ds/2555

[5] S. Mochizuki, S. Shiota, and H. Kiya, "Voice liveness detection based on pop-noise detector with phoneme information for speaker verification," *Journal of the Acoustical Society of America*, vol. 140, no. 4, pp. 3060, 2016. [Online]. Available: https://doi.org/10.1121/1.4969520

[6] M. Merchant, *Voice Authentication and Face Recognition*, GitHub repository, 2024. [Online]. Available: https://github.com/MohamadMerchant/Voice-Authentication-and-Face-Recognition.

[7] S. Sagar, C.-T. Li, S. W. Loke, and J. Choi, "Poisoning attacks and defenses in federated learning: A survey," *IEEE Security and Privacy*, Dec. 2022. [Online]. Available: https://arxiv.org/abs/2301.05795

[8] M. A. Ramirez, S.-K. Kim, H. Al Hamadi, E. Damiani, Y.-J. Byon, T.-Y. Kim, C.-S. Cho, and C. Y. Yeun, "Poisoning attacks and defenses on artificial intelligence: A survey," *arXiv preprint arXiv:2202.10276*, Feb. 2022. [Online]. Available: https://arxiv.org/abs/2202.10276

[9] H. Song, M. Kim, D. Park, J. Shin, and J.-G. Lee, "Learning from noisy labels with deep neural networks: A survey," *ResearchGate*, Mar. 2022. [Online]. Available: DOI:10.1109/TNNLS.2022.3152527