5-2023

RADIC Voice Authentication: Replay Attack Detection using Image Classification for Voice Authentication Systems

Hannah Taylor

RADIC Voice Authentication:

Replay Attack Detection using Image Classification for Voice Authentication Systems

by

Hannah Taylor

An Undergraduate Thesis Submitted in Partial Fulfillment

of the Requirements for the University Honors Program

East Tennessee State University

Hannah M. Taylor 4/14/23

Hannah Taylor Date

Ghaith Husari 4/14/23

Dr. Ghaith Husari, Thesis Advisor Date

Mathew Desjardins 4/14/23

Mr. Mathew Desjardins, Reader Date

4/14/23

[illegible]

Dr. Phillip Pfeiffer, Reader Date

Abstract

Systems like Google Home, Alexa, and Siri that use voice-based authentication to verify their users' identities are vulnerable to voice replay attacks. These attacks gain unauthorized access to voice-controlled devices or systems by replaying recordings of passphrases and voice commands. This shows the necessity to develop more resilient voice-based authentication systems that can detect voice replay attacks.

This thesis implements a system that detects voice-based replay attacks by using deep learning and image classification of voice spectrograms to differentiate between live and recorded speech. Tests of this system indicate that the approach represents a promising direction for detecting voice-based replay attacks.

Table of Contents

1 Introduction

Voice Authentication (VA) is a means of hands-free identity verification that uses voice recognition to unlock devices or accounts. It serves the same general purpose as a password or other biometric authentication methods but without requiring users to make touch or eye contact with a device. While this offers speed and ease of access to users, it also opens a door to attackers and can leave devices open to replay attacks. A replay attack is an attempt to gain access to devices or accounts by replaying a recorded passphrase to the authentication device.

The ramifications of successful replay attacks range from minor inconveniences to identity theft. For example, if an attacker were to gain access to a banking app that uses VA, the victim's information and finances would be completely compromised. Thus, detecting these attacks as they occur is essential in their prevention.

One promising approach for detecting such attacks involves the use of a trained convolutional neural network (CNN) to distinguish between spectrograms of live and recorded human speech through image classification. CNNs are a deep learning (DL) algorithm used in machine learning (ML) and are often applied to problems involving image classification and speech recognition. They offer fast classification of (e.g.) an image's contents extracted with less computational strain than a traditional neural network.

This work investigated the performance of using CNN for spectrogram image classification with 90 training cycles to accurately detect replay attacks. Overall, the system achieved an accuracy score of 69% in detecting an attack. This result shows the potential of using image classification techniques to help secure voice-based systems.

2 Related Work

2.1 Methods of Voice Authentication and Feature Extraction

In [8], Meng et al. propose the use of active authentication (AA) to improve the security of voice authentication (VA) systems. Common authentication methods test what a user knows—e.g., a password—or possesses—e.g., a key—to grant access to an application or device. Most authentication systems implement a one-time authentication check for the duration of the logged session. AA provides “continuous and real-time monitoring of the user’s identity” (ibid.) without disrupting a user’s workflow. VA’s contactless nature makes it a good choice for AA.

VA processes the most important attributes of an audio signal from a speaker such as pitch, range, and vocal tract length. Traditionally, methods for measuring these attributes have used large amounts of data ranging from several minutes to hours of recorded speech. Improvements in statistical modeling and analysis such as the hidden Markov model (HMM) and i-vector approaches have reduced the amount of test data needed for an accurate estimate of a speaker’s characteristic attributes. However, these methods still fall short of real-time requirements for constantly monitoring and verifying a speaker’s identity. For example, “many i-vector based systems exhibit sharp performance degradation when they are tested with short duration (below 5s) utterances” (ibid.). Meng et al. mention research that generated baselines with test data samples of two seconds using Gaussian-mixture-model-based universal background models; these systems, however, require large computations that may complicate their use for real-time active voice authentication (AVA).

The main challenge with real-time voice authentication “is to effectively train talkerspecific models, using as little enrollment speech as possible” (ibid.). Most speech analysis systems break test data into fixed-length segments of speech (speech frames) and test groups of frames, under the assumption that the sample is from one speaker. This assumption, however, is incompatible with AVA, which must also detect changes in speakers.

To detect changes in speakers in speech samples, Meng et al. use a window-based approach to analyzing speech. A single window “is expressed in number of frames N_w” and is a test segment of the speech signal. During authentication, as speech frames pass through the window, the analysis measures how well the current frames match the target speaker’s voice. These scores are calculated based on an adapted version of the HMM, which generates models for the target speaker and imposters and are log-likelihood ratios taken from each window.

Meng et al.’s system must be trained before use. To minimize its use of enrollment data, Meng et al. use a speaker-independent (SI) model in training. The SI “is trained on a sufficient pool of data from a general collection of speakers in the training set” (ibid.). When the system registers a user, it generates a speaker-adapted model by applying the SI’s parameters to the target speaker’s sample data. A minimum verification error (MVE) algorithm is used to reduce the number of errors output by the system. Typical errors include a “miss”, or wrongful authentication of an imposter is wrongfully authenticated, and a “false alarm”, or wrongful rejection of the target speaker. The authentication system’s performance, its windows-based equal error rate, is determined by how often the miss and false alarm rates are equal. Because AVA uses window-based tokens, “the sample tokens for training and enrollment must each correspond to a segment of speech signal within a test window” (ibid.) to ensure that the training aligns with AVA’s short-time testing requirements.

In [9], Lei and She propose a new method of identity vector extraction for VA systems. Their i-vector model of human speech converts speech signals into feature vectors that encode their key attributes while suppressing redundancies in human speech. Lei and She rejected Mel-frequency cepstral coefficients, one common method for obtaining feature vectors from signals, as unsatisfactory, due to the method's use of the short-time Fourier transform, which incorrectly assumes the signal is stationary. Their alternative method, wavelet analysis, uses a "multi-scale resolution which is suitable for analyzing non-stationary signal" (ibid.).

Wavelet analysis can be conducted using one of several transforms. One, the Wavelet Transform (WT), which only decomposes a signal's low frequencies, is insufficiently precise for human speech. A second, Wavelet Packet transform (WPT), decomposes the full frequency spectrum. WPT, however, is complex and implements a regular decomposition: this creates too much overhead for real-time analysis. A third, perceptual WPT, gives the same output quality as WPT but reduces computational cost by using an irregular decomposition. It also reduces speech noises, making it the best option for wavelet entropy feature extraction.

Feature extraction begins with a three-stage procedure that normalizes and frames speech and removes silence from frames. In the normalization stage, "the effect of volume is discarded and utterance becomes comparable" (ibid.). The framing stage divides the normalized signal into multiple short-term frames. Finally, periods of silence are removed from the sample, reducing its length. Lei and She give equations for Shannon entropy, Non-normalized Shannon entropy, Log-energy entropy, and Sure entropy as commonly used forms of wavelet entropy calculation.

After feature vectors are retrieved from the wavelet entropy calculations the i-vectors are extracted. Lei and She identify a feature vector's frame posteriors as "a key issue of i-vector extraction” (ibid.). The authors estimate the frame posteriors using a universal background model (UBM). The UBM is trained on background utterances: a collection of speech signals spoken by different people. Typically, the UBM would be trained on the Gaussian mixture model (GMM), but it only “considers the inner information within feature vectors and is trained in a generative way…” (ibid.). Deep neural networks (DNN) could be used to improve training by including correlational information between vectors and by being trained discriminatively. However, DNNs require large numbers of parameters, which increases the system’s complexity and computational costs. Instead, the authors use convolutional neural networks (CNN), which are like DNNs, but with fewer training parameters.

Applying this i-vector approach to VA systems requires the UBM to be trained before the VA’s enrollment and evaluation stages. Obtaining the i-vector allows the authentication system to differentiate between the user and the imposter from an unknown speaker sample by comparing the unknown speaker’s i-vector to i-vectors in a database and comparing the verification score to a defined threshold.

In [11], Aizat et al. discuss the use of deep neural networks (DNNs) and i-vector extraction for VA and identification. Voice identification “is the process of determining whose speaker provides the speech” (ibid.). VA determines a speaker’s identity to allow access to a service or device. Each case generally uses voice activity detection algorithms and automatic speech recognition to teach machines to independently detect and analyze speech. For voice recognition, Aizat et al. use Transformed Mel-Frequency Cepstral Coefficients (T-MFCC) built on i-vector algorithms to feed a hidden Markov model (HMM) based DNN.

Aizat et al. trained the DNN using phoneme labels together with elementary speech unit

(ESU) clusters. Phoneme labels, which are extracted from the transcript and audio files, are aligned using HMMs to provide a level of word isolation. ESU clusters are groups of similar speech (phonetic) indicators found with signal feature extraction techniques such as i-vector. They are extracted from sets of files containing speech recordings from speakers. ESU clusters are used to "improve computational indicators, reduce subjective decisions in biometric systems, and increase the security against attacks" (ibid.).

VA systems typically consist of an input device, a speech processing system, a storage system for templates (sample speech for verification), a comparison and decision system, and a user interface. Attacks against each component of a VA system – except the input device – are common. Some examples include replacing, adding, or deleting sample speech template files, as well as recording and repetition (replay) attacks. Most of these attacks can be thwarted "by applying digital coding, encrypting an open data transmission channel and using time stamps" (ibid.). However, an attacker can use a Dictaphone to recreate a recorded reference message. To combat this, Aizat et al. provide a multistage approach. Stage 1 records the user's reference speech material. Stage 2 compares the standard from stage 1 with what is spoken for authentication at a given moment. The author's proposed model only requires a verification phrase of 3-5 seconds where identification is not dependent on language.

Aizat et al.'s model "increased the overall classification accuracy by about 13%" (ibid.) with low probabilities of errors due to producing equal counts of false acceptance rates and false rejection rates. The use of phoneme labels also helped to reduce/remove background noise and silent frames from the speech audio files being analyzed.

In [7], Moffat et al. present a comparison of audio feature extraction toolboxes. “Audio features are contextual information that can be extracted from an audio signal” (ibid.) and are usually categorized as low-level – e.g., features computed from the signal frame-by-frame – or high-level – e.g., key of a note. Moffat et al. apply the Cranfield model to evaluate a set of toolboxes based on their coverage, effort, presentation, and time lag. Coverage depicts how many features are extracted by a specific toolkit as well as its other processing capabilities – as a measure of how well it covers all relevant information. Effort refers to how easily the interface can be manipulated and used. Presentation is how easily and in what format the features can be saved as output. Time lag refers to the efficiency of the tool.

Moffat et al. compared Aubio, Essentia, jAudio, Librosa, LibXtract, Marsyas, Meyda, MIR Toolbox, Timbre Toolbox, and YAAFE. The MPEG-7 standard list of seventeen low-level audio descriptors was used as a basis for comparison. For coverage, Essentia provides the “largest range of features, and is the only toolbox to produce 100% coverage of the MPEG-7 audio descriptors” (ibid.). MIR Toolbox and LibXtract covered 85% or more of the MPEG-7 descriptors. Librosa, jAudio, and YAAFE all came in under 40% of MPEG-7.

For effort, several toolboxes only provide command line interfaces, which can limit control over the system. Vamp plugins were found to be the most flexible options as different plugins offered different interface options. Moffat et al. also reviewed the amount and quality of existing documentation on the list of toolboxes, noting that LibXtract – along with a couple others – provide limited documentation.

Tool formats for presenting data included XML, YAML, and JSON. However, only a couple toolboxes provided users with options of how the data was to be displayed – i.e., structured, or tabular. Meyda and LibXtract are meant to be used for real-time computations and are thus designed to run faster. Toolboxes written in C based languages also run faster than those written in Python and Java.

2.2 Liveness Detection

In [2], Zhang et al. propose VoiceGesture, a liveness detection system for VA that measures articulatory gestures. VA enables users to authenticate without making direct contact with a device. Potential applications for VA range from authenticating access to a mobile device to forensics and banking.

Zhang et al. note that VA systems are vulnerable to spoofing attacks, playback (replay) and mimicry attacks being the two main forms of attack. In playback attacks, an attacker plays a recording of a user's passphrase to the VA system's microphone. In mimicry attacks, an attacker imitates a user's voice either with voice or the simultaneous use of voice and recording replay. These attacks are easy and practical due to the ready availability of low-cost, high-quality recording devices and speakers.

Many previous VA methods relied on the acoustic characterizations of speech for user identification. These methods, however, have high error rates and limited effectiveness. Other systems use the challenge-response technique, which may require a speaker to move the device along a pre-determined path or hold it a certain way during authentication.

VoiceGesture does not require such practices. Rather, it uses articulatory gestures (AGs), which are specific to how people pronounce and intone speech. This enables AGs to serve as a reliable authentication that resists spoofing attacks. AGs are combinations of movements by multiple muscles around the lower face (e.g., lips, tongue, throat, and jaw) known as articulators. Zhang et al. observe that AGs create speech, noting that “the coordination among multiple articulators produces gestures like lip protrusion, lip closure, tongue tip and tongue body constriction, and jaw angle” (ibid.). For example, p’s are uttered by closing a speaker’s lips during speech. While humans create sound through AGs, electronic speakers typically lack elaborate mechanics, relying on a diaphragm for sound production.

VoiceGesture checks for live speech by detecting a mobile device user’s AGs, using the device’s built-in speakers and microphones as a Doppler radar system. A device’s speaker emits a sound frequency of 20kHz, which reverberates off a user’s articulators as the articulators move. VoiceGesture picks up these reflections as a pre-defined passphrase is being captured by a device’s mic.

VoiceGesture processes inputs in five steps. It first extracts Doppler shifts from a voice sample by converting it to the frequency domain, then segmenting phonemes to remove non-audible segments or pauses. Phonemes are segmented by using the lowest two captured frequencies to identify vowels and other frequencies, along with hidden Markov models and other algorithms to identify consonants. The second step extracts two features from the Doppler shifts. The first, energy-band frequency, is the set of energy levels captured based on an articulator’s proximity to the microphone; this yields a frequency contour for each band. The other, frequency-band energy, refers to the energy contours produced by an articulator’s velocity during speech movement. The third step, wavelet-based denoising, “further remove[s] the noisy component mixed in the extracted features” (ibid.). Zhang et al. use the discrete wavelet transform to divide the signal into two parts, one of which, the detailed coefficient, contains the noise segments. A dynamic threshold is applied to extract and remove noise segments that can be caused by hardware or a user’s environment. A correlation coefficient is then used to assess the captured speech’s similarity to a reference recording, followed by a determination of the utterance’s authenticity.

Zhang et al.’s system was tested with three different phones, all running Android OS version 6 with different speaker and microphone models. Accuracy at sampling rates 48, 96, and 192kHz was tested, as well as accuracy with a user holding the device in two positions: against the ear as if talking on the phone normally and in front of the mouth. Tested passphrase lengths varied between 2 to 10 words. Overall, VoiceGesture produced a detection or accuracy rate of around 98% with a 1% false acceptance rate. It was noted that “participants who have smaller scale of articulatory movements generate higher false acceptance rate” (ibid.).

In [4], Wang et al. propose VoicePop, an anti-spoofing system for VA systems. Various methods for combating these attacks, including liveness detection and automatic speaker verification suffer from a variety of limitations. For example, “the extent of articulatory movements affects the effectiveness of [AG- based authentication]” (ibid.).

VoicePop uses pop noise in speech signals to differentiate between live speakers and attacks. Human speech, which is uttered on an exhale, produces different phonemes with varying bursts of air. These bursts of air are known as pop noise. If, during VA, a user’s mouth is close enough to the microphone, the microphone will capture pop noise along with the audible passphrase. Wang et al. note that “pop noise has a high energy in the low frequency” and “the duration of pop noise varies in the range 20~100 msec” (ibid.). Since different phonemes are produced via different means, using different movements and amounts of breath, the probability of pop noise being present during the pronunciation of each phoneme is different.

VoicePop operates in four phases. The first of these uses traditional voice authentication to extract and validate spoken words against a passphrase. If the words are correct, the system uses a second, data processing phase to segment phonemes and detect pop noise. The final two stages of VoicePop detect replay and impersonation attacks, respectively. Most replay attacks can be rejected immediately as recordings are often taken too far from a speaker's mouth to record pop noise. Impersonation attacks are detected by using a phoneme-pop sequence algorithm to identify how a speaker breathes while speaking and comparing it to the enrolled verification sample.

VoicePop detects pop noise in six steps. VoicePop removes non-speech segments from a signal, then puts the modified signal through the short-time Fourier transform. It locates the peak of a given pop noise by checking for a frame whose energy within the range 0-93Hz is greater than three times the standard deviation of the energy of all the frames in that range. The derivative is then taken to find the differential coefficient of the boundaries around a peak. Specific boundaries are determined when the energy at some nearby point (within 3 points of the peak) is less than or equal to .45 times the energy at the peak and the derivative of the nearby point is greater than or equal to .45 times the derivative of the peak.

To test VoicePop, the authors recruited 18 volunteers and evaluated the system's accuracy under replay and impersonation attacks. Overall, the system achieved over 93% true acceptance rate with just over 5% equal error rate. When the authentication device was more than 6 cm from the speaker the system's accuracy decreased rapidly. Accuracy was also reduced when the device is closer than 2 cm.

In [10], Anand et al. propose EchoVib, a vibration-based means of VA. Most VA algorithms analyze audible input, which leaves them open to voice synthesis attacks. By contrast, EchoVib uses vibration patterns from short segments of replayed speech to detect voice synthesis and modeling attacks. These patterns, which a smartphone's loudspeakers created as a passphrase is (re)played, can be detected by the phone's built-in accelerometers. As noted by Anand et al., these vibrations depend in part "on the source generating the vibrations in response to the speech signal" (ibid.). These vibrations are specific to individual phones due to minor differences in parts and defects, even between two speakers of the same make and model. The resulting combination of a user's speech signal and that person's device produces a signal that is extremely difficult for an attacker to recreate: an attacker would need to "generate a correct response to the EchoVib challenge-response authenticator, while synthesizing a vibration pattern response that could pass the verification" (ibid).

EchoVib's verification process requires users to register with the system and train it by providing input. After this, when users respond to an authentication challenge with a passphrase, the system replays the passphrase using the authentication device's loudspeakers and uses the accelerometer to record the device's vibrations. These vibrations are subsequently compared to the initial training data to approve or reject the authentication attempt.

As a basis for training their system and analyzing user input, Anand et al. implemented two algorithms for feature extraction and three ML algorithms. They compared Mel-frequency cepstrum coefficients with time and frequency domain feature sets as inputs and compared support vector machines, regression algorithms, and decision trees as bases for classification. They found that time-frequency domain feature sets and the random forest classifier, a decision tree ML algorithm, were best for authentication. Overall, EchoVib showed a true positive rate of over 90% against voice synthesis attacks and over 85% against voice modeling attacks.

2.3 ML in Cybersecurity

In [3], Xin et al. discuss ML and deep learning (DL) and their use in cybersecurity. Xin et al. define cybersecurity as “technologies and processes designed to protect computers, networks, programs and data from attacks and unauthorized access, alteration, or destruction” (ibid.). ML and DL are types of artificial intelligence applications; they imitate human thought processes and respond appropriately to various challenges. ML, according to Xin et al., “primarily focuses on classification and regression based on known features previously learned from the training data” (ibid.). DL, on the other hand, uses neural networks and tends to focus on data learning characteristics that enable predictions on largely unknown variables.

ML and DL models are constructed using a similar sequence of steps. Both typically begin with a determination of the features that will be used for prediction. This is followed by a choice of prediction algorithm, typically classification or regression; the training of candidate models; a determination of which of these models is best; and a final classification or prediction of the data. In DL, unlike ML, “feature extraction is automated rather than manual” and “model selection is a constant trial and error process…” (ibid.).

ML and DL models can exhibit what experts refer to as supervised, unsupervised, and semi-supervised learning based on the prediction and classification algorithms. Supervised learning, which assumes the use of labeled data, identifies relationships between labels to content. Unsupervised learning is commonly used for unlabeled data: e.g., datasets that are too large or costly to label. According to Xin et al. this approach “deduces the description of hidden structures from unlabeled data” (ibid.). It also suffers from a lack of procedures for measuring its accuracy. Semi-supervised learning models are used to bridge the gap between label maintenance and calculating accuracy.

Similar approaches are used to evaluate the effectiveness of ML and DL models. One, a confusion matrix, is a table that characterizes a classifier’s outputs. Binary classifiers, for example, can be characterized in terms of correct (true positive, true negative) and incorrect (false positive, false negative) classifications. Other common confusion matrix metrics include accuracy, precision, recall, and the F1-score. Accuracy is a percentage of correctly identified samples to the total. Precision is the percentage of correct identifications divided by the incorrect. Recall is the ratio of all correctly identified samples to all samples that should have been identified. The F1-score is the “harmonic mean of the precision and the recall” (ibid.).

Although ML and DL models are constructed and trained similarly, they differ in terms of their data and hardware dependencies, feature processing techniques, problem solving methods, execution time frames, and interpretability. DL algorithms need much larger data sets than ML algorithms for the same level of accuracy and precision. DL algorithms are typically used to solve more complex computations than ML algorithms and require more processing power to do this. ML often requires a more detailed and expert breakdown of a problem’s characteristics than DL, which can function with more abstract content. ML approaches tend to decompose problems via a set of subproblems, while DL typically avoids such decompositions. DL usually takes longer to train than ML—for example, weeks instead of minutes or hours—but less time to run tests. ML algorithms can provide rationales for their choices; current DL algorithms cannot.

ML- and DL-based applications are often deployed as elements of a layered approach to cybersecurity, as defensive technologies that check for possible attacks. They are often incorporated in intrusion detection systems (IDS), a class of applications that checks for events that are associated with known attacks and/or anomalous uses of a system. Anomaly-based detection allows for detection of zero-day attacks: unknown or new attacks that misuse detection cannot detect. Xin et al. observed that most modern ML and DL methods in cybersecurity are hybrid approaches of intrusion detection.

ML algorithms used in cybersecurity applications include support vector machines (SVM), k-nearest neighbor (kNN) algorithms, and decision trees (DT). A SVM classifies a dataset's items by partitioning them into clusters, using hyperplanes. kNN classifiers measure the distance between a dataset's items, grouping items within some k value of the current instance. The current instance is classified by the majority classification of its neighbors' classifications within the range of k. Decision trees use tree structures to depict the use of sequences of tests to characterize a data item's attributes; the tree's leaf nodes represent categories into which these items might be classified.

DL algorithms used in cybersecurity applications include deep belief networks (DBN), recurrent or recursive neural networks (RNN), and convolutional neural networks (CNN). Xin et al. define a DBN as "a probabilistic generative model consisting of multiple layers of stochastic and hidden variables" that represents the correlations between instances in the data with weights (ibid.). RNNs are used when the data is presented in a sequential format where a previous instance's output alters or relates to the output of the current instance. CNNs use a shared weight system that more closely resembles biological neural networks than other neural network designs while reducing the network complexity. It produces hierarchies of the features used for learning extracted from unlabeled data.

In [1], Aiyanyo et. al. review ML algorithms used in cybersecurity, datasets used to train those algorithms, and cyber threats that could be thwarted by ML algorithms. The authors focus primarily on defensive cybersecurity techniques, which are reactive and attempt to prevent known attacks based on previous knowledge.

ML algorithms used in defensive cybersecurity include SVMs, which are typically used in IDSes; decision tree classifiers; Naïve Bayes classifiers; random forest classifiers; logistic regression; neural networks; and hybrid algorithms. Naïve Bayes algorithms are simplified Bayesian classifiers that provide good accuracy and performance for their simplicity. Random forest algorithms are based on decision trees but use several trees. They average the predictions from all the trees to produce the final classification decision; this can improve upon decision trees by reducing the false alarm rate and time required for processing. According to Aiyanyo et al., “Logistic regression is a probabilistic linear classifier that involves the projection of input vectors onto hyperplanes” (ibid.). Logistic regression, according to the authors, is best used for removing noisy data from the network. Neural networks are also used in IDS. They can solve larger problems because they can extract patterns from data without needing supervision or much prior knowledge of the data. Hybrid systems typically combine signature and anomaly detection techniques, using a knowledge of known attacks and variances in the normal flow of network traffic to mitigate cyber-attacks.

Any of these ML algorithms can be trained with using supervised, unsupervised, or semi-supervised learning. Unlike supervised approaches, unsupervised techniques do not need to be trained on as much historical data pertaining to the network and can be more generally applied to different problems. As for semi-supervised ML, the authors state that this “is useful when the training data is insufficient for supervised ML, but the unsupervised alternative may not give the best results” (ibid.).

Offensive cybersecurity methods attempt to proactively detect attacks and remove them as they are detected before any data is lost. Aiyanyo et. al. noted that there were fewer offensive applications of ML in cybersecurity and that the most common algorithms for this approach were association rule, neural network, and SVM.

Of the data sets used to train ML algorithms in cybersecurity, Aiyanyo et. al. identified KDDCUP’99 database as the most common and the DARPA’99 data set as a common basis for testing IDS methods. Because of its age, the DARPA’99 set was often used with other data sets. Log files and data from honey pots were also used, with the latter primarily being a source for attack vectors.

Aiyanyo et. al. identified “denial of service, user to root, remote to local, and probe attacks” as attacks that have been mitigated with ML systems (ibid.). Denial of service attacks effectively shut down service by overloading a network or a host with traffic. In a user to root attack, an attacker attempts to elevate their privileges on the system to gain control. In a remote-to-local attack, an attacker “exploit[s] vulnerabilities which could involve the guessing of passwords to take control over a remote machine” (ibid.). In probe attacks, an attacker gathers information about a system and its weaknesses to use in a later attack.

Zero-day attacks can be also mitigated with ML, usually with anomaly detection-based systems. Zero-day attacks are previously unknown attacks; as such, they are difficult to identify.

Despite advances in the use of ML in cybersecurity, Aiyanyo et. al. observe that the classification problem continues to be a challenge as the complexity of network packets and processes grow with the emergence of more network-based communication technologies. Additionally, the use of more software to detect and mitigate security issues creates a greater attack surface for attackers to exploit.

In [5], Shaukat et. al. review the literature on ML algorithms in cybersecurity, comparing their performance and discussing challenges with using ML in cybersecurity. Their review includes assessments of SVM, decision tree, deep belief network (DBN), artificial neural network (ANN), random forest, and Naïve Bayes algorithms as reported in the research literature, focusing on defense against malware and spam attacks and ML-based IDSs. Researchers had evaluated these algorithms based on their precision, recall, accuracy, ROC curve, and error rate based on the confusion matrix, with accuracy being the most documented performance measurement.

The highest accuracy listed for SVM is 99.30% when used for IDSs. Decision trees had the highest reported accuracy when used with IDS at 99.96% with malware detection following extremely close behind. An anomaly-based approach to IDS using DBN returned a 99.45% accuracy with the highest accuracies for malware and spam detection approaches using DBN returning around 96 and 97%, respectively. For ANN, the highest accuracies for IDS, malware, and spam were 99.82%, 92.19%, and 93.71%, respectively. Random forest had highest accuracies for IDS, malware, and spam all above 98%. For Naïve Bayes IDSs, Shaukat et. al. identified accuracies ranging from 99.9% to 36%.

Overall, most algorithms' accuracies were reported to be at or above 90% with few falling below 85%. Several approaches were evaluated for IDS, malware, and spam relative to different data sets. The reported highest accuracies for IDS, malware, and spam were not consistently obtained from a specific approach. For example, the highest accuracies for IDS with SVM and decision tree were from hybrid and misuse-based approaches, respectively.

Current challenges of using ML in cybersecurity, as discussed by Shaukat et. al., are the lack of current data sets, lack of standardized methods of feature extraction, low attack detection speed, and high costs of hardware and data for some methods.

In [6], Priyam et. al. compared the accuracy of three serial decision tree algorithms, C4.5, ID3, and CART, for predicting a student's test performance. C4.5 and ID3 are based on Hunt's algorithm, have serial implementations, and use a splitting attribute found by sorting the tree with the information gain ratio. However, C4.5, unlike ID3, can handle incomplete training data and can be trained on continuous or discrete datasets. CART, which is not based on Hunt's algorithm, classifies data attributes by splitting them into two classes; it can also be used in regression techniques.

Pryiam et al. noted that CART had an accuracy of around 67% on "some data sets using 10-fold cross validation" (ibid.). One of these algorithms' limitations is a decrease in performance on larger datasets. To address this, the authors mention two other algorithms – i.e., SPRINT and SLIQ – that can be implemented serially or in parallel to handle larger sets of data.

2.4 Image Classification

Image classification is the practice of training DL algorithms to recognize patterns within images so that the images can be sorted into classes. Tripathi [16] mentions applications of image classification that include medical research, navigation, and biometric authentication.

Tripathi’s work focused on the use of image classification to detect and categorize types of fruit using filters and feature detectors. In order to achieve the best results, Tripathi first standardized the data by resizing the images and performed data augmentation to increase the size of the training dataset. The author then tested 4 types of CNN architectures in an attempt to maximize the accuracy of the system – DenseNet, VGG16, VGG19, and InceptionV3. The results of Tripathi’s study were reported using accuracy, recall, precision, confusion matrices, and f1-scores. By using CNNs and image classification, the four frameworks used achieved accuracies ranging from 87.81 to 100.

3 Methods and Experiment

3.1 The Proposed Approach

To determine the extent to which spectrograms of human voices could be used to detect replay attacks, a system was developed for analyzing spectrograms generated from audio files of human voices and voices replayed from mechanical speakers. The system, known as Replay Attack Detection using Image Classification (RADIC), consists of the following four main modules (Figure 1):

- *Audio sample preparation*. This module chooses a sample from the audio files dataset based on a preferred sample size. In this work, 2200 random files were chosen with 50:50 split ratio, where 1100 are Spoof files – i.e., recordings of replayed utterances – and 1100 are Bonafide, or recordings of human voices.
- *Voice to spectrogram (audio to image) conversion*. This module converts all the audio files in the dataset into colored spectrograms.
- *Image preprocessor*. This module applies standardization, normalization, and data augmentation techniques on the generated spectrograms.
- *Feature extraction and deep learning classifier*. This module uses a CNN machine learning algorithm to construct a detection model that classifies spectrograms as human (Bonafide) audio or machine-generated (Spoof) audio.

3.2 Dataset

This research's dataset was obtained from the open source Zenodo repository [12] released by the creators of the 2021 ASVspoof challenge. According to [12], the data from the

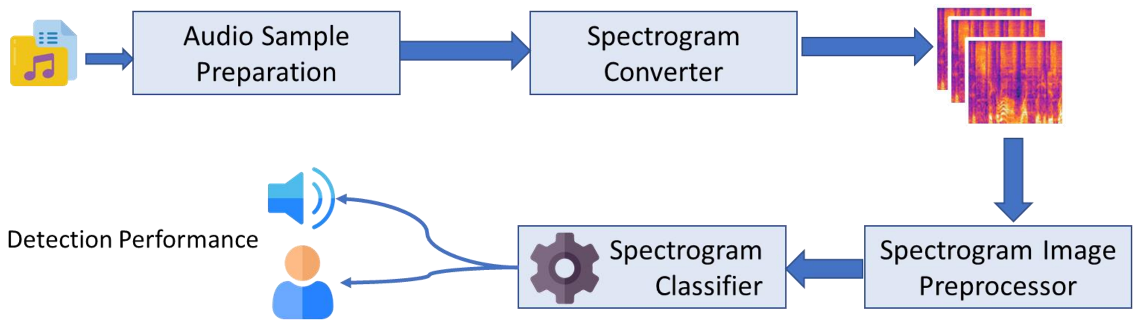

Figure 1: The Major Modules of RADIC

Physical Access (PA) set of this challenge "comprises bonafide speech and replayed recordings both collected in a variety of real physical spaces". This data collection consisted of various scenarios and combinations of different replay devices, microphones, and spacing between the attacker and human speaker. The resulting dataset consisted of segments of audio files generated from human speakers and replayed audio from an attacker's device. Using the posted metadata keys for the repository, the audio files were separated into Bonafide and Spoof datasets. The subset of audio files used from the PA ASVspoof dataset were filtered into separate folders based on filenames and corresponding categories found in the metadata file.

The audio files were then converted into spectrogram images for the CNN model to classify. The ffmpeg framework handled the conversions from audio to images. Ffmpeg can handle almost any format of multimedia files – including audio – and comes with multiple filters, libraries, and functions for file manipulation. Using ffmpeg's showspectrumpic filter, the original .flac audio files were converted to spectrogram images with the single following line of code:

```
for i in *.flac; do ffmpeg -i $i -lavfi showspectrumpic=s=960x540 ${i%.*}.png; done
```

The ffmpeg command was run from an Ubuntu terminal from within the directory of the original audio files. This command takes all the .flac audio files and generates an image using showspectrumpic setting the size of each image to 960x540 and replaces the file type with .png. After the images were created, they were separated from the audio files and placed in their own directories/folders, yielding the finalized dataset used in the CNN model creation and speaker verification testing. In total, the process generated 3,759 images converted from the ASVspoof audio files selected at random consisting of 1,188 bonafide images and 2,571 spoof images.

3.3 Image Preprocessing

Before the data was used by CNN for training and testing, it was preprocessed to standardize it, normalize it, and vary it, using data augmentation to create content that differs slightly from the original. This last step was intended to improve the classifier's performance and training stability: i.e., to avoid skewed results that the model might generate using inconsistent or excessively uniform data.

The standardization process ensured that all images had a common size and color format. This step used methods from the image processing python library cv2 (a.k.a. OpenCV). The resize() method was used to make sure each image was the same height and width, which in this case was a hard-coded variable called *img_size* equal to 224. The imread() method was used to ensure that all images are in the red-green-blue (RGB) color format.

The normalization process rescaled all data to a common range. This process entailed two steps. First, the RGB values produced by the standardization step set each pixel's color to a value between 0 and 255. Since large numbers can slow the computations of CNNs, this first step divided each pixel by 255. Then the method reshape() from the NumPy library was used to scale the images based on the pixel division operation. This ensures that the results of the CNN classification are returned as a percentile.

The final, data augmentation step was implemented in order to increase the CNN's accuracy by slightly altering the dataset to increase the amount of training data. This was done to minimize the likelihood of overfitting due to an excessive degree of uniformity in the training set's images. This step utilizes the keras ImageDataGenerator, which is a function that used to perform the following functions on the data:

- randomly rotate image within a selected range (30 degrees)
- randomly zoom image
- randomly shift image horizontally and vertically a fraction of the images dimensions
- randomly flip images

3.4 Constructing the Machine Learning Model

The CNN model (Figure 2) was built using TensorFlow Keras API because of its flexibility in image processing. The documentation pages for Keras describe it as being simple, flexible, and powerful [13]. Keras is widely used for many deep learning algorithms including image classification. The specific model class used was the Keras Sequential class which takes a linear stack of – typically – convolutional and pooling layers. According to the Keras documentation guides, "a Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor" [14]. This works for the dataset used in

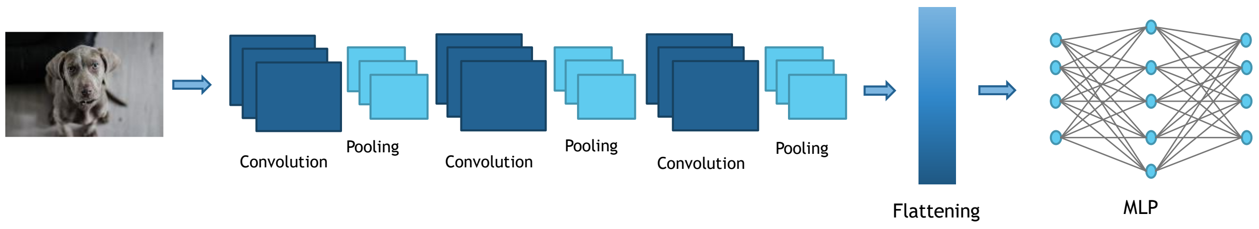

Figure 2: The CNN Model for Image Classification

this work because each layer accepts one type of input and produces one type of output – the image and the classification, respectively.

The model is a standard 3-layer CNN, with pooling layers after each convolutional layer. To avoid overfitting for small datasets – this occurs when the testing phase has an unfair advantage usually because the dataset was too small and training data is being used in testing – a dropout layer was added to the model after the third max pooling layer. The dropout layer sets random inputs back to zero while scaling non-zero inputs so that the sum of all inputs stays the same [15].

To train and test the model, the original dataset was reduced to 1,100 images for each classification (2200 total) during audio sample preparation. The training dataset received a random 80% of the images from each classification, and the testing dataset received the remaining 20% from each. This 80-20 split produces a training set that is large enough to avoid overfitting. The model was set to complete 90 epochs during training and is designed to take in 32 images at a time (i.e., the batch size for the CNN is 32). An epoch is one iteration of the dataset through the CNN training algorithm; the number of epochs can increase the model's accuracy depending on the size of the dataset being used. The batch size determines how many samples will be run through the network before the model's training parameters are updated. By training the model with 90 epochs, in the training process, the model went through the entire training dataset 90 times. With a batch size of 32 and a total training dataset size of 1760, it takes the model 55 iterations to finish one epoch (i.e., the number of iterations per epoch equals the dataset size divided by the batch size).

4 Evaluation

The Sequential CNN model's accuracy was determined by presenting it with images from the test dataset in order to determine how accurately it classified images from the Bonafide and Spoof datasets. The following three measures were used to characterize the model's accuracy:

- Precision: the percentage of data the classifier correctly identified as spoof
- Recall: the percentage of overall spoof data the classifier identified
- f1-score: the harmonic mean of precision and recall

4.1 Results

Running the CNN model with 90 epochs during the training stage resulted in an overall accuracy of 69%. The precision, recall, and f1-score are listed in Table 1.

Class	Precision	Recall	F1-score
Bonafide	0.70	0.65	0.67
Spoof	0.67	0.72	0.70

Table 1: Precision, Recall, and F1-score for Bonafide vs Spoof classification

Figure 3 shows the accuracy measures for detecting spoof attempts using the proposed system (RADIC) with 90 epochs.

Despite the limited amount of data that was used to train the model, the model's gradual improvement in accuracy over the course of training indicates that any overfitting was minimal.

4.2 Discussion

The model's ability to accurately classify 69% of the test data indicates that it did in fact recognize some patterns between Bonafide and Spoof data.

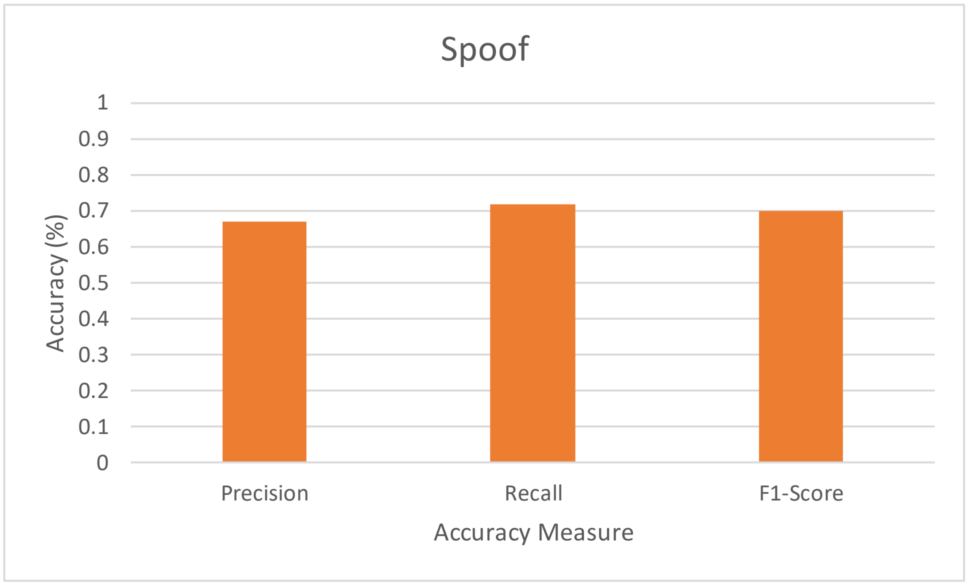

Figure 3: Spoof Detection Accuracy

The model labeled 67% of the data correctly as spoof, which indicates that the approach warrants further study. The recall rate for spoofed data was 72%, meaning that the CNN has a 28% chance of missing a replay attack. Deciding which measure to improve upon in the future warrants a discussion as to which business application it would be applied to and whether that business values customer satisfaction (i.e., limiting the number of users incorrectly identified as spoof) over better security (i.e., maximizing the number of replay attacks caught).

4.3 Future Work

One potential direction for further research would be to generalize this model to classify other types of attacks against VA systems such as deep fakes and text-to-speech spoofing attempts. The ASVspoof challenge repository can serve as a source of data for both. A second would be to determine how an attack's setting and means of delivery affects the model's performance: e.g., assessing the impact of the setting in which an attack was recorded and the quality of the recording and replay devices on the model's accuracy. A third would be to determine a limit of diminishing return for the number of epochs run and the accuracy achieved by running more epochs. Generally speaking, if the model continues to improve and does not reach a state of diminishing returns (where the improvement curve nearly flattens), then adding more epochs (300, 500, 800) could improve the model's accuracy so long as it does not begin overfitting the data.

5 Conclusion

This thesis presented RADIC, a voice-based authentication system. The proposed system constructed a deep learning (CNN) model that attempts to differentiate between live speech and recorded speech that was replayed in a simulated spoof attack against a VA system. The voice dataset was first converted to spectrogram images, then the CNN model was used to detect spoof attempts. An overall accuracy of 69% was achieved with a subset of the ASVspoof physical access dataset. The proposed system's evaluation shows a promising direction for intrusion detection of malicious voice-replay attacks to gain unauthorized access to voice-based authentication systems. A potential future direction for improving the proposed system is to experiment with different configurations for the CNN algorithm such as training the model on more images and using more advanced image feature extraction techniques.

References

[1] I. D. Aiyanyo, H. Samuel and H. Lim, "A Systematic Review of Defensive and Offensive Cybersecurity with Machine Learning," *Applied Sciences,* 2020.

[2] L. Zhang, S. Tan and J. Yang, "Hearing Your Voice is Not Enough: An Articulatory Gesture Based Liveness Detection for Voice Authentication," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, Dallas, TX, 2017.

[3] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou and C. Wang, "Machine Learning and Deep Learning Methods for Cybersecurity," *IEEE Access,* 2018.

[4] 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," *IEEE,* pp. 2062-2070, 2019.

[5] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, S. Chen, D. Liu and J. Li, "Performance Comparison and Current Challenges of Using Machine Learning Techniques in Cybersecurity," *energies,* 2020.

[6] A. Priyam, Abhijeet, R. Gupta, A. Rathee and S. Srivastava, "Comparative Analysis of Decision Tree Classification Algorithms," *International Journal of Current Engineering and Technology,* 2013.

[7] D. Moffat, D. Ronan and J. D. Reiss, "An Evaluation of Audio Feature Extraction Toolboxes," in *Int. Conference on Digital Audio Effects,* 2015.

[8] Z. Meng, M. U. B. Altaf and B.-H. Juang, "Active Voice Authentication," *Digital Signal Processing,* 2020.

[9] L. Lei and K. She, "Identity Vector Extraction by Perceptual Wavelet Packet Entropy and Convolutional Neural Network for Voice Authentication," *entropy*, 2018.

[10] S. A. Anand, J. Liu, C. Wang, M. Shirvanian, N. Saxena and Y. Chen, "EchoVib: Exploring Voice Authentication via Unique Non-Linear Vibrations of Short Replayed Speech," in *Cyber-physical Systems*, Hong Kong, 2021.

[11] K. Aizat, O. Mohamed, M. Orken, A. Ainur and B. Zhumaxhanov, "Identification and authentication of user voice using DNN features and i-vector," *Cogent Engineering,* 2020.

[12] X. Liu, X. Wang, M. Sahidullah, J. Patino, H. Delgado, T. Kinnunen, M. Todisco, J. Yamagishi, N. Evans, A. Nautsch and K. A. Lee, "ASVspoof 2021: Towards Spoofed and Deepfake Speech Detection in the Wild," *Latex Class Files,* vol. 14, no. 8, August 2021.

[13] "Keras," [Online]. Available: https://keras.io/about/. [Accessed 1 April 2023].

[14] "Keras," [Online]. Available: https://keras.io/guides/sequential_model/. [Accessed 1 April 2023].

[15] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, R. Jozefowicz, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, M. Schuster, R. Monga, S. Moore, D. Murray, C. Olah, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V.

Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu and X. Zheng, "TensorFlow: A System for Large-Scale Machine Learning," 21 January 2022. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout#:~:text=The%20Dropout%20layer%20randomly%20sets,over%20all%20inputs%20is%20unchanged.. [Accessed 1 April 2023].

[16] M. Tripathi, "Analysis of Convolutional Neural Network based Image Classification Techniques," *Journal of Innovative Image Processing*, vol. 3, no. 2, pp. 100-117, 2021.