NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

THE USE OF REGRESSION MODELS FOR DETECTING DIGITAL FINGERPRINTS IN SYNTHETIC AUDIO

by

William B. Brown

June 2022

Thesis Advisor:	Robert L. Bassett
Second Reader:	Ross J. Schuchard

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2022	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
THE USE OF REGRESSION MODELS FOR DETECTING DIGITAL FINGERPRINTS IN SYNTHETIC AUDIO

5. FUNDING NUMBERS

6. AUTHOR(S) William B. Brown

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

Modern advancements in text to speech and voice conversion techniques make it increasingly difficult to distinguish an authentic voice from a synthetically generated voice. These techniques, though complex, are relatively easy to use, even for non-technical users. It is important to develop mechanisms for detecting false content that easily scale to the size of the monitoring requirement. Current approaches for detecting spoofed audio are difficult to scale because of their processing requirements. Individually analyzing spectrograms for aberrations at higher frequencies relies too much on independent verification and is more resource intensive. Our method addresses the resource consideration by only looking at the residual differences between an audio file's smoothed signal and its actual signal. We conjecture that natural audio has greater variance than spoofed audio because spoofed audio's generation is conditioned on trying to mimic an existing pattern. To test this, we develop a classifier that distinguishes between spoofed and real audio by analyzing the differences in residual patterns between audio files.

14. SUBJECT TERMS spoofing, synthetic audio	**15. NUMBER OF PAGES** 101
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

THE USE OF REGRESSION MODELS FOR DETECTING DIGITAL FINGERPRINTS IN SYNTHETIC AUDIO

William B. Brown
Major, United States Army
BA, Grinnell College, 2007

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 2022

Approved by: Robert L. Bassett
Advisor

Ross J. Schuchard
Second Reader

W. Matthew Carlyle
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Modern advancements in text to speech and voice conversion techniques make it increasingly difficult to distinguish an authentic voice from a synthetically generated voice. These techniques, though complex, are relatively easy to use, even for non-technical users. It is important to develop mechanisms for detecting false content that easily scale to the size of the monitoring requirement. Current approaches for detecting spoofed audio are difficult to scale because of their processing requirements. Individually analyzing spectrograms for aberrations at higher frequencies relies too much on independent verification and is more resource intensive. Our method addresses the resource consideration by only looking at the residual differences between an audio file's smoothed signal and its actual signal. We conjecture that natural audio has greater variance than spoofed audio because spoofed audio's generation is conditioned on trying to mimic an existing pattern. To test this, we develop a classifier that distinguishes between spoofed and real audio by analyzing the differences in residual patterns between audio files.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

ASV	Automatic Speaker Verification
ASVSpoof	Automatic Speaker Verification and Spoofing Countermeasures Challenge
AUC	area under the relative operating characteristic (ROC) curve
AWS	Amazon Web Services
CART	classification and regression tree
CNN	convolutional neural networks
DNN	deep neural networks
DWT	discrete wavelet transforms
EER	Equal Error Rate
FoR	Fake-or-Real
GAN	generative adversarial networks
GBM	gradient-boosted tree model
GMM	Gaussian Mixture Models
HMM	Hidden Markov Models
LPC	Linear Predictive Coding
NPS	Naval Postgraduate School
RMSE	root mean square error
RNN	recurrent neural networks
ROC	relative operating characteristic

SAT speaker-adaptive training

SML supervised machine learning

STFT short time Fourier transform

SS speech synthesis

SSML Speech Synthesis Markup Language

TP/FP true positive to false positive

TTS text-to-speech

VC voice conversion

Executive Summary

In an increasingly interconnected world, the United States is becoming more vulnerable to attacks targeted at disrupting social cohesion as adversaries leverage the information space to discredit public figures. One type of attack is the use of deepfakes, where bad actors can digitally manipulate videos, images and audio to appear as if originating from target individuals. As deepfake technology becomes more accessible throughout the online community, it is important we develop mechanisms for detecting false content that easily scales to the monitoring requirement.

This thesis addresses the concern for scalability by reducing the question of false content to its audio component. We introduce a method that is simple to implement and therefore easy to scale. This method is based on our theory that real audio has more natural variation than synthetic audio and that this discrepancy can be captured using the distribution of an audio file's residuals. Using discrete wavelet transforms (DWT) of different wavelet families, we reconstruct smoothed approximations of the original audio signal to extract the residuals. The summary statistics from the audio file's resultant distribution of residuals are then analyzed by the gradient boosted-tree model (GBM) classifier to determine whether the analyzed audio is genuinely representative of a human voice or has been synthetically manipulated to sound like that human voice.

We applied this method to two distinct datasets, the Fake-or-Real (FoR) dataset, and the 2015 Automatic Speaker Verification and Spoofing Countermeasures Challenge (ASVSpoof) dataset. Both datasets challenged the classifier to correctly classify synthetic audio files that were generated by algorithms excluded from the train set. In our experiments, we observed that the wavelet used to train the best performing classifier changed depending on which spoofing algorithms were used in the validation set. The FoR validation set only used Google's *Cloud Text-to-Speech (TTS) with Wavenet* algorithm to create its synthetic files. Consequently, out of 105 wavelet families, the best classifiers for the FoR validation set was provided by the coiflet-14 DWT. In contrast, the 2015 ASVSpoof tested the classifier on five spoofing algorithms excluded from the train set. Results from these sets of experiments suggested that the selection of wavelets to build the classifier involves compromising between identifying voice conversion (VC) spoofing algorithms and TTS algorithms.

Lastly, we used the 2015 ASVSpoof dataset to construct a classifier to analyze the deepfake spoofing attack on the Ukrainian president, Volodmyr Zelensky, that was uploaded to social media on March 16, 2022. In this deepfake, President Zelensky reportedly calls on Ukrainian defenders to surrender to Russian forces during the first month of the 2022 Russia-Ukraine military conflict. By training the classifiers on individual spoofing algorithms, and by consolidating the scores of relevant classifier results, we produced evidence that the audio portion of the deepfake attack on President Zelensky was the product of a VC spoofing algorithm.

Acknowledgments

First and foremost, I would like to thank my lovely wife, Millyzzen, for her unwavering support through the last two years of this difficult program. I also would like to thank Dr. Robert Bassett for his patience, experience and mentorship that were critical in shepherding me throughout this entire process. I am grateful to my family for their support throughout my education and career. I am also grateful to the United States Army for providing me the opportunity to attend this institution and further my education.

Lastly, I would like to thank my mother who recently passed away at the start of this project. As a former college professor, she had a passion for academia and would have been very proud of how everything has turned out. This thesis is in memory of her.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

> Russia has operationalized the concept of perpetual adversarial competition in the information environment by encouraging the development of a disinformation and propaganda ecosystem . . . Disinformation is a quick and fairly cheap way to destabilize societies and set the stage for potential military action.
>
> – U.S. Department of State, 2022

We live in a world where voice-enabled technology is increasingly prevalent and increasingly vulnerable to spoofing attacks. Bad actors' false assumption of a person's vocal identity through spoofing creates consequences ranging from petty identity theft to the erosion of social trust in important public figures (Chesney and Citron 2019a). This furthers an environment of uncertainty that is only accelerated by the increased use of online communication platforms. We need to develop more effective ways for combating that uncertainty and in a manner capable of processing the large volume of digitally enabled interactions.

This thesis explores a new method for developing a classifier that can distinguish between spoofed and authentic digital audio recordings of human speakers. This method for classification primarily relies on analyzing the audio files' residuals between the actual audio and various compression techniques. In the paper, we apply this method to two separate sets of spoofed or real datasets and highlight differences between fully synthetic text-to-speech (TTS) methods and voice conversion (VC) methods. Lastly, we show how this method demonstrates the inauthenticity of a widely debunked deepfake video through audio-only analysis.

1.1 Motivation

When impersonators spoof a target speaker's voice, they electronically mimic a person's vocal characteristics such that artificially generated utterances are ascribed to a target speaker. In other words, not only can people modify their voice to mask individual identities,

but they can also assume the vocal identities of other people. Advancements in TTS and VC techniques leveraging deep neural networks (DNN) make it increasingly difficult to distinguish a natural sounding voice from a synthetic voice (Reimao and Tzerpos 2019). These techniques, though complex, have "the potential to proliferate widely. Commercial and even free deepfake services have already appeared in the open market...The capacity to create professional-grade forgeries will come within reach of nearly anyone with sufficient interest and the knowledge of where to go for help"(Chesney and Citron 2019b, p. 2). Therefore, we expect increased propagation of spoofed voices leveraged for either nefarious purposes or mainstreamed personal entertainment. Countering this falsified information will require more resources for early detection that can scale beyond operator-centric, case-by-case scrutiny.

Our method addresses the resource consideration by only looking at an audio file's residuals when determining if it is sourced to the targeted speaker. In this paper, residuals are the differences between an audio file's smoothed waveform and its actual waveform. We theorize that natural audio has greater variance than spoofed audio because spoofed audio's generation relies on mimicking an existing pattern. To test this, we develop a classifier based on various summary statistics of the residuals for both authentic and spoofed audio that assigns a binary probability to whether an unidentified audio file is believed to be fake. This thesis explores the potential of this technique to identify unnatural audio that was likely spoofed.

1.2 Relevance

In a world of modern convenience, the incorporation of synthetic audio has accelerated with recent advances in small form computing. Commonly recognized phrases, such as "Siri," "Alexa" and "Hey Google," can activate voice-enabled interactions with small devices that have large implications. Using just their voice, people turn on vacuum cleaners, manage security features in smart homes and even remotely authenticate online financial transactions. While this is convenient, it creates considerable vulnerabilities as each digital-enabled conversation provides opportunities for training an algorithm that fingerprints to a speaker's individual vocal identity. If bad actors successfully spoof that vocal identity, these household conveniences become liabilities. Additionally, propagation of fraudulently attributed voice clips threatens the public credibility of spoofed speakers.

Unfortunately, spoofing technology is becoming more available to the online community and is becoming better integrated with the growing phenomena of increasingly sophisticated deepfakes and its potentially far-reaching consequences. Historically, the label "deep fake" refers to the use of "deep learning" and neural networks to generate the artificial content. For example, one variation of deep learning uses iterative tuning from within generative adversarial networks (GAN) where a "generator" algorithm is pitted against a "discriminator" algorithm producing digital voices that sound highly realistic (Chesney and Citron 2019a). Although deepfakes are increasingly less cost prohibitive, they are not the only spoofed audio-visuals in public circulation. Our classifier looks at a range of spoofing techniques not limited to deep learning algorithms. For this reason, this thesis uses the deepfake label more broadly, "as shorthand for the full range of hyper-realistic digital falsification of images, video, and audio" (Chesney and Citron 2019a, p. 4).

While people normally associate deepfakes with swapping the faces of speakers in video media (Agarwal et al. 2019), it is important to note that people also respond profoundly to audio-only information. This influence is not limited to radio shows and podcasts but extends to simple sound files no longer than a minute in length. In 2014, a secretly recorded audio-only clip of a NBA team owner, Donald Sterling, speaking disparagingly about minorities resulted in the forced sale of the Los Angeles Clippers and a lifetime ban from the NBA. In 2016, a leaked Access Hollywood tape of a lewd 2005 conversation between then-presidential candidate Donald Trump and a television host was effectively used to polarize support between political parties. Most recently, when Russia invaded Ukraine in February 2022, the defiant audio-only response of Ukrainian soldiers on Snake Island to a Russian Warship generated a groundswell of support for Ukrainian resolve and was even immortalized in a Ukrainian national stamp (*Guardian,The* 2022). Clearly, in appropriate contexts, small audio clips are powerful purveyors of impactful information when authenticated as real.

Given the impact of both audio and video information in the public space, it is understandable that adversaries use the information space to discredit opponents. This contested information space contributes to an environment of informational uncertainty. In this public context, information uncertainty refers to "uncertainty regarding the truth of signals in a distorted media environment" and people are more likely to question the veracity of media presented to them due to the widespread multivariate nature of the information and the

author biases (Schiff et al. 2021, p. 7). The application of deepfakes, both successful and unsuccessful, leverages this information uncertainty to present unique social, political, and security challenges.

Socially, deepfakes provide opportunities for groups to strongly react to fake video or audio directly attributed to other figures. Such reactions can easily become hyperbolized as social media not only decentralizes the propagation of digital media, but also encourages large populations to respond emotionally and cumulatively to compressed packets of information. In many cases, online communities center around positions or perspectives of influential public figures. Not only can inflammatory deepfakes disrupt leadership hierarchies, scaled population reactions reinforce stereotyped, oppositional narratives. Target audiences for insurgent groups are already susceptible to messages furthering discontent (Chesney and Citron 2019b). Deepfakes that are not quickly and decisively debunked can accelerate biases and foment potential instability.

Deepfakes, even when debunked, exacerbate the public mistrust of media signals that is characteristic of information uncertainty. The consequences of this reluctance to accept information leads to vulnerabilities when political figures claim the "liars' dividend" to discredit any compromising information as fake news. Ideologically, the existence of this uncertainty incentivizes polarized groups to retreat further into information bubbles that reinforce their beliefs (Schiff et al. 2021). Ironically, these echo chambers foster the acceptance and propagation of inflammatory deepfakes to increase polarization and potential instability.

Recent real-world examples illustrate these emerging challenges and their potential impact on security. In June 2020, a faked audio message attributed to the Ghana's president, Nana Akufo-Addo, went viral on social media. The quickly debunked audio claimed that President Akufo-Addo warned Ghanans of a corporate driven attempt to use COVID-19 to depopulate the world (*Premium Times* 2020). Four months later, Ghana's National Cyber Security Center debunked a fake audio clip that spoofed an international news outlet's report on a falsified story of government money laundering (Appiah-Dolphyne 2020). Most recently on March 16, 2022, a deepfake of Ukrainian President Zelensky was boosted on Russian social media calling for Ukrainian soldiers to surrender to Russia. Although western social media platforms quickly debunked and de-platformed the deepfake, experts

still expressed concern about the impact of this this type of disinformation in the future (*National Public Radio* 2022).

Although in the above examples the adverse impacts of spoofing public figures were quickly and successfully minimized, continued success cannot rely on a system of individual scrutiny, predicated on the deepfake already going viral. The dynamics of the social media environment make rapid detection even more important. The expanse of digital space from which initial information can be generated drastically reduces the probability for rapid detection. The scale at which that information can be initially propagated is far too variable for humans to consistently process. If a false narrative remains unchecked, we cannot underestimate the exponential speed at which certain emotional elicitations accelerate at a viral spread. As current online communication platforms evolve to facilitate increased variety in user-generated content, it is important we develop mechanisms for detecting false content that easily scales to the size of the monitoring requirement.

The method explored in this thesis addresses this concern for scalability by reducing the question of false content to its audio component. Instead of conducting frame-by-frame video analysis to detect inconsistencies in the synchronization of lips, expressions and vocal patterns, our classifier seeks to raise flags on possible signatures for manipulated audio. If successful, the classifier can support the triage of digital media requiring further review, or make outright determinations of authenticity when comparing authentic audio files of the same speaker.

In the next two sections, we provide a general background on the current methods for spoofing audio. We also discuss contemporary methods for how audio experts are already attempting to distinguish between real and fake audio. This background information provides a strong foundation for the follow-on chapters where we outline our methodologies and describe the results of the follow-on experiments.

1.3 Literature Review

The upcoming sections provide a general overview of the spoofing methods that will be challenged by our classifier in upcoming chapters. The last sections of this chapter discuss the current methods for detecting spoofed audio and provides perspective on how these methods inform or contrast with the methodology outlined in this thesis.

1.3.1 Background on Spoofing Methods

This section outlines three methods for synthesizing speech that can be easily integrated into deepfake videos. The first two methods, which rely on speech synthesis (SS) and VC, arise directly out of efforts to bypass speech-based biometrics systems. The third method uses deep-learning-based speech algorithms for TTS audio files designed to be indistinguishable from human speakers (Reimao and Tzerpos 2019). This third method highlights an evolution in synthetic speech attacks because it departs from previous methods' reliance on the utterances of targeted human speakers. All these methods for synthetically generating natural sounding human speech contribute to the problem of spoofing and highlight the need to ground detection techniques in features common to all methods.

Public interest in spoofing has evolved since the advent of speech-based biometric systems and its identifying role in granting access to authenticated user. In their paper, Paul et al. (2017) summarizes the four types of spoofing efforts for circumventing these security measures: mimicry, replay, speaker-adapted SS and VC. While these methods vary in effectiveness, the individualized nature of secure access enables Automatic Speaker Verification (ASV) to simply be another layer integrated into a personalized system for verification. Additionally, specific training data sets using proprietary parameters built around an individual's unique utterances make it more difficult for spoofing software to capture a target's unique acoustic features (Poddar et al. 2018). Despite spoofing's declining ability to bypass security systems, SS and VC can easily be adapted to support the development of convincing deepfakes and merit discussion in the following paragraphs.

Voice Conversion

VC is generally understood as taking the speech signal from a source speaker and the modifying that signal to sound as if it is pronounced by a different speaker. In earlier research,

Stylianou et al. (1998) noted that speaker-specific information such as speaking rate, pitch contour, or other overall speech dynamics are individualized into respective spectral envelopes and acknowledged that analysis of these spectral envelopes at the segmental level led to efficient discrimination for distinct speakers. Their paper looked at ways to statistically map and thereby introduce mechanisms for control of the relations between distinct spectral envelopes. This control at the segmental level would constitute one of the earlier methods for conversion to reduce audible discrimination between speakers.

Subsequent researchers built on this earlier mapping of distinct spectral envelopes and generalized beyond the constraint of parallel utterances between speakers. Dutoit et al. (2007) used the Viterbi algorithm to select sequences of frames from a target speaker database that matches segments within the speech of the spoofing speaker. In this way, components of speech from a target speaker can be used to refine elements of another speaker's distinct sequence to be closer to the target speaker. Another method uses insights from Fukada et al. (1992) about an algorithm that can converge mel-cepstral coefficients such that the slopes of the source spectrums between speakers are more similar. While the previous two VC methods looked to leverage segmented similarities between two speakers, Saito et al. (2012) proposed a tensor method that adapts a spoofer's voice to the target using a vectorized speaker space constructed from many pre-stored speakers. Although more complicated VC techniques exist, these examples provide context for how VC algorithms map and mirror elements from source to target speakers.

Speech Synthesis

In contrast to voice conversion's reliance on previously spoken phrases, SS uses select target utterances as the building blocks for constructing entirely unique sentences still sounding like the target. The idea to break utterances into their most basic sounds and storing them as building blocks in a database of diphon waveforms is an old concept. In the late 20th century, Charpentier and Stella (1986) were already improving on existing Linear Predictive Coding (LPC) methods for concatenating these waveforms into intelligible syllables. As summarized in their paper, LPC-coded speech utilized the predictive prosodies (or rhythms) in speech elements to smooth transitions of disparate diphone waveforms at the excitation signal level. Their proposal, however, provided a better overlap of diphones by representing the waveforms through a short time Fourier transform (STFT). Representing

the waveform using STFT enabled researchers to match the pitch marks in the waveforms. In turn, this addressed LPC's difficulty with voiced fricatives (i.e., consonant transitions) and allowed the text-to-speech to sound more natural.

Despite the clear utility in TTS applications for assisting machine-communication, the resulting robotic quality of the voices meant TTS would not provide viable samples for effective spoofing until the early 21st century. However, computer assisted granularity in signal processing provides more opportunities for generalizing patterns across larger datasets. The Panda and Nayak (2017) paper on using vowel onset point based waveform concatenation techniques is a good example for how TTS methods develop in natural tonality. While the paper focuses on Indian languages, the concept of analyzing the duration of consonant units between vowel transitions shows how more rules can be applied to generalize across similarities in the speech patterns embedded in various languages.

Although researchers have demonstrated growing capability for developing algorithms to better capture the rhythms unique to various languages, the issue remains that modifying the timbre of the speech is necessary for capturing the emotional tenor for target speakers. To address this, Japanese researchers used contextual Hidden Markov Models (HMM) in 1999 to train on large datasets of acoustic features (Yoshimura et al. 1999). Along with other methods, this concept for incorporating the human acoustic range into synthetic speech has undergone continuous refinement and presentation during annual Blizzard Challenges that started in 2005 (Fraser and King 2007). For example, researchers in 2009 used a speaker adaptive approach that trained an average voice model from several speakers. Their method showed how a space speaker-adaptive training (SAT) algorithm reduced the negative influence of speaker differences in the data set. This meant the researchers could use that averaged speaker model to assist in training on the utterances of a target speaker. Therefore, while other methods required thirty minutes of target speaker utterances, this approach produced natural-sounding synthetic speech with only six minutes of target speaker utterances (Yamagishi et al. 2009). As researchers find more ways to generalize different speaker data to fill out the emotional spectrum for target speaker patterns, the gap between a voice actor's natural resonance and the tinniness commonly detected in TTS continues to narrow.

Deep Learning Text-to-Speech

The last generation method we discuss is similar to voice adapted speech synthesis but involves deep learning, characterized by its use of neural networks. This section looks at two types of deep learning TTS. The first type is designed to train on and sound like an individual speaker, and the second type generates entirely unique voices that sound virtually indistinguishable from generic human speakers. Due to the growing success of neural networks' utility for transforming text to dramatically better natural sounding speech, companies are commercializing the potential for engaging voice-enabled smart devices (Cambre and Kulkarni 2019). This means more synthetic voices, of a greater variety, are being evaluated less on the ability for spoofing target phrases, but on the ability to expand synthetic voices' range of natural human conversation. Consequently, priorities are shifting to rely less on side-by-side comparisons between a spoofed voice and its target. Rather, the popular challenges for effective speech-file testing evaluate the collection of unpaired human and synthetic voices from which the classifier must agnostically identify signature elements defining synthetic speech.

While deep learning TTS algorithms have been effectively commercialized to support user-integration with generic human-sounding voices, the parallel deep learning elements to support speech processing remain very effective at mimicking a target speaker. One good example is the /textitDeepVoice3 that was developed by Baidu Labs. In his book, Taylor (2009) provides a general breakdown for how the TTS process develops a multi-stage, hand-engineered pipeline. This systematic pipeline transforms text to an audio representation (usually a spectrogram) that is then converted to audio through an audio waveform synthesis vocoder method. Neural networks help manage the myriad of features through various hidden layer combinations and transformations that result in pipelines with fewer components and higher quality synthetic speech. While there is no consensus on an optimal neural network architecture for TTS, Baidu researchers noticed that sequence-to-sequence methods such as Tacotron demonstrated ways for incorporating random initialization of <text, audio> pairs to accelerate training the model (Wang et al. 2017). Although recurrent neural networks (RNN) are still used to translate a text-character representation to spectrograms representations, Baidu /textitDeepVoice3 researchers proposed ways for using convolutional neural networks (CNN) to scale the method across very large data sets (Ping et al. 2017). This ability to utilize and generalize different speaker data sets to accelerate

training on one speaker may seem similar to the techniques introduced in the 2008 Blizzard Challenge (Yamagishi et al. 2009) mentioned in the last section. The use of deep learning architecture, however, means that more datasets can be incorporated at a greater scale, and the generalizations across datasets are not as intuitive because the transformations occur within various hidden layers.

The utility for extracting generalizable and applicable acoustic features from massive amounts of agnostic speaker data is not limited to quickly training models for spoofing voices of target speakers. Large cloud computing companies like Google and Amazon Web Services (AWS) use these techniques to refine and expand the linguistic range for proprietary synthetic voices. The Speech Technology Research Department of Google stresses that it prioritizes making "speaking to devices around you, devices you wear, devices with you ubiquitous and seamless" (Google 2022). Google's declared passion for "computing scale and data" pairs extremely well with its ability to grow its datasets with "millions of users talking to Voice Search or the Android Input everyday" (Google 2022). Additionally, Google TTS services allow people to upload a Speech Synthesis Markup Language (SSML) that can support the individual tailoring of emphasis or breaktimes (Reimao and Tzerpos 2019). These improvements feed not only into algorithms that better process speaker-to-device input but also enable Google Home and other smart devices to communicate more fluidly in over 55 languages around the world. Similarly, AWS Polly, a service charging for every character converted to speech, has created two unique brand voices for Kentucky Fried Chicken Canada and National Australia Bank. As synthetic voices continue to improve their natural speaking tenors using generative neural networks, more companies will realize the commercial value in adopting brand voices that can "adopt different speaking styles in different contexts, improving customer experiences" (Gantenbein 2020).

This trend supports the value of research examining datasets that do not prioritize spoofing attacks on select speakers. This may appear counterintuitive considering the paper's emphasis on countering spoofed speech. However, the audible quality in computer-to-human interactions clearly show an accelerating convergence of TTS methods that approaches the resonant complexity of voice actor conversions. Using voice conversion to transform a human-indistinguishable TTS speech sample to target speakers is the logical next step. Therefore, it is important for classifiers to be effective in agnostically identifying synthetic speech without the predicated alert of an individualized targeted spoofing attack. The next

session provides a brief overview of the elements within synthetic audio that were sufficiently distinct from natural human speech to discriminate themselves using conventional classification methods.

1.3.2 Background on Detection Methods

The most widely known community-sourced effort to combat spoofing attacks is the bi-annual Automatic Speaker Verification and Spoofing Countermeasures Challenge (ASVSpoof) that started in 2015. This competition provides researchers an opportunity to test new approaches for distinguishing between synthetic and real audio. The most recent event concluded in September 2021 and included challenges for producing or detecting methods to capture and reproduce audio originating through a microphone or in a physical space and repurposed to bypass security authentication measures. Another challenge was to effectively distinguish between synthetic and natural audio files in a variety of conditions and without speaker verification (Wu et al. 2017a). In the following paragraphs we discuss two methods that have been successful in detecting synthetic audio files because they narrow in on distinct aspects within the generation of synthetic audio. We also highlight the advantage of neural networks for agnostic detection of audio not predicated on paired speaker comparisons. The last detection method has the most similarity to the methodology proposed in the following chapter because it exploits the predictive nature of audio and the retention of spectral features in signals across time.

One conventional approach for detecting synthetic audio examines the audio files that are translated onto visual spectrograms measuring amplitudes at varying frequencies across tiny snapshots in time. The most typical result from this type of analysis shows a tendency for synthetic audio to reveal inconsistencies at higher frequencies. This can be detected by understanding that spoofing algorithms tend towards the Mel scale that prioritizes distinctions in the lower human audible frequencies (Umesh et al. 1999). Therefore, gaps in the synthesis of audio have been more readily identified at higher frequencies after researchers adjusted the weights of frequencies in the Mel scale (Paul et al. 2017). This distinction at higher frequencies makes it easier for comparisons with target speaker examples and can also be generalized to binary classification of synthetic audio voices.

The second successful method uses Gaussian Mixture Models (GMM) to represent the emission densities from the HMM parameters extracted from audio files (Lu et al. 2011). The GMM-based classifier works as follows. The state of the underlying HMM captures both the phoneme being uttered and the authenticity of the audio clip. The state of HMM can also include which generator was used to create a synthetic piece of audio clip, if this information is produced. Given the HMM's state, which provides information on the phoneme and the veracity of the audio clip, the observed audio sample is assumed to be drawn from a mixture of Gaussian distributions with state dependent parameters. These state dependent parameter are estimated when training the model. Then, at test time, estimating the state of the HMM from samples of an audio clip provides an estimate of the audio clip's authenticity, or to which distribution hit belongs (Povey et al. 2011). In most cases, the distributions would be classified into two categories, either natural or synthetic. However, the classification of distributions can be extended to make it possible to distinguish between different spoofing methods. This is possible because HMM captures the dynamic features for both the excitation parameters and spectral parameters for targeted speech using a multi-space probability distribution (Yamagishi et al. 2009). The excitation signal is the source sound that takes on the qualities of the spectral filter (or vocal tract) through which it is processed (Narendra and Rao 2015). The HMM parameters capture a wide range of acoustic qualities such as magnitude-based features, phase-based features, pitch-pattern features, and modulation patterns (De Leon et al. 2012). These qualities extend beyond the discrimination at higher frequencies that was stressed in the last paragraph. Because the HMM parameters for these features often form the basis for creating a similar sounding synthetic audio, GMM's ability to incorporate the HMM parameters made it very successful in identifying spoofed audio.

Recently, neural networks have become very effective in classifying falsified speech (Yu et al. 2017). One of the shortcomings of the insights from ASVSpoof challenges was the tendency for detecting algorithms to rely on prior knowledge of spoofing types. The degradation in efficacy from known to unknown attacks was particularly noticeable using the GMM classifier despite using Deep Neural Networks (DNN) to extract the relevant features (Yu et al. 2017). An advantage of using neural networks to analyze files is the networks' use of hidden layers to automatically learn features from audio input signals. In other words, neural networks are reacting to and incorporating classifications of acoustic

features along with previous methods to form a more robust conclusion. For example, Zhang et al. (2017) showed how combinations of CNN and RNN to integrate feature extractions and modeled long term dependencies improved earlier abilities for agnostically identifying spoofed audio. Despite these considerably sophisticated advances in the granular analysis of audio, neural network analyses' success for detecting spoofed audio was comparable to conventional spectrogram analysis in its prioritization for discrimination between higher and lower frequencies (Reimao and Tzerpos 2019).

Given the use of neural networks to both generate and detect synthetic speech, the last detection method discussed here prioritizes tracing the audio features that are unique to the generative process in speech synthesis. In other words, the detection method looks for features that are present in synthetic audio but absent in natural audio. Motivated by the understanding that modern synthesized speech passes through different layers in a neural network, Singh and Singh (2021) used a combination of cepstral and bispectral statistics to explore whether autocorrelative features can be identified in synthetic audio. These insights led to the current method's focus on "studying the relationship between past and current audio samples" and the extent to which previous samples predict the next sample (Borrelli et al. 2021, p. 3). While this predictive element was popularized in earlier LPC compression methods, researchers were now able to simultaneously compare short-term (sequenced samples) and long-term (fundamental voice sounds) predictors within the same audio file. This means that researchers could identify the extensions of micropatterns in series of pairwise sequences to entire utterances that are consistent with the signature of a synthetic generation method.

This method's emphasis on temporal features has the most similarities to the methodology used in this thesis because it prioritizes the predictive value of how, in human speech, the resonant quality of previous utterances carries forward into future speech. However, predictive trace methods suffer the same issues as predecessors because they require breaking the audio signal into a plethora of complementary features within which the classifier hunts for specific aberrations. This remains an issue because spoofing advancements will continue to correct on the microscopic inconsistencies identified by current detection methods. In the next chapter, we provide the theoretical background to this thesis' proposed methodology

that addresses the concern that the growing complexity for synthetic audio detection may not scale well to the corresponding complexity and variety with which spoofed audio is created. We do this by redirecting the focus from “What in the audio is spoofed” to “Is the audio spoofed?” This distinction relies primarily on using a machine to evaluate a piece of audio as “too good” and not on “what is wrong.”

CHAPTER 2:
Technical Background and Methodology

In the previous chapter, we saw that the creation and detection of synthetic audio is increasingly nuanced in its detection of minute acoustic features within the audio. This leads to potential problems of scale as the variety of detection methods, variety in generation methods, and sheer volume of digital voices threaten to outpace manual attempts to distinguish between synthetic and real human audio.

This chapter introduces a method for analyzing audio that will be simple to implement and therefore easy to scale. This method is based on our theory that real audio has more natural variation than synthetic audio and that this discrepancy can be captured using the distribution of an audio file's residuals. Residuals are the differences between an audio file's actual amplitude and the amplitude of its compressed form. This analysis of differences between compressed signals and the actual audio is the distinguishing feature of the methods presented in this thesis.

The first section describes the theory's motivation in terms of basic audio features. The second section describes our use of wavelet smoothing methods to extract the audio residuals on which we test our theory. The third section introduces the classifier and the analyzed features supporting the identification of fake audio. By the end of this chapter, the reader will understand the underlying principles of the theory, our approach to extracting residuals from an audio file, and our use of a gradient-boosted tree model (GBM) to analyze the residuals. This thesis uses the GBM as the classifier to determine if an audio file originated from either a synthetic generation method or a human speaker. The next chapter covers our specific implementation and experimental results.

2.1 Technical Problem Description

We propose that natural variation in spoofed audio is less than in real audio of a human speaker. This variation can be measured by analyzing the differences in amplitude between an original audio and its compressed signal, which we call the residual. The compressed signal is a simplified representation f the actual signal from which the residual is calculated.

Given the appropriate compression algorithm, this thesis conjectures that the resulting distribution of residuals for spoofed audio will have less variance than the distribution of residuals for real audio. Consequently, our goal is to show that distinctions in residuals are useful in distinguishing real and fake audio when used as inputs to a feature-based classifier.

In this thesis, compression of digital audio refers to interpolation of the discrete points of amplitude representing signals stored as a function of time in a digital audio file. Because of the natural oscillations in human speech, this interpolation will have a natural curve that approximates the general waveform representation of an audio file. Additionally, this curve will tend to be smoother than the pattern of originally sampled points. For this reason, we will treat smoothed signals and compressed signals as synonyms when referring to the interpolating curve that represents an approximation of the original signal. Points that lie along the smoothed signal are referred to as $x_{\text{smooth}}(t)$ and exist as a function of time (t). In contrast the actual sample is represented by $x(t)$ and the residual error is captured as $e(t)$. This gives us the following equation:

$$e(t) = x(t) - x_{\text{smooth}}(t).$$

Given this expression, we anticipate the residuals to have less variation for synthetic audio. Figure 2.1 depicts this theory with a representation of two hypothetical audio files composed of audible samples represented by pressure amplitude across time. The figure shows a representation of synthetic audio and of natural audio. The actual audio signals are the blue points, and the compressed signal is the green line. For this hypothetical example, we can easily differentiate between the synthetic audio and the natural audio by comparing the absolute values of their residuals. This data-driven approach to classifying files as real or synthetic provides a level of reproducibility and statistical rigor that manual classification by subject matter experts does not. The example in Figure 2.1 is artificial because the samples in both audio signals are randomly generated from different probability distributions with the mean centered along the sine curve. However, it highlights a key characteristic of the compression algorithm that is central to our theory: For the residuals to inform our assessment of an audio file's variation, the compression algorithm must smoothly interpolate the audio samples. In other words, the residuals should capture the difference between the original audio and a smoothed version of that audio.

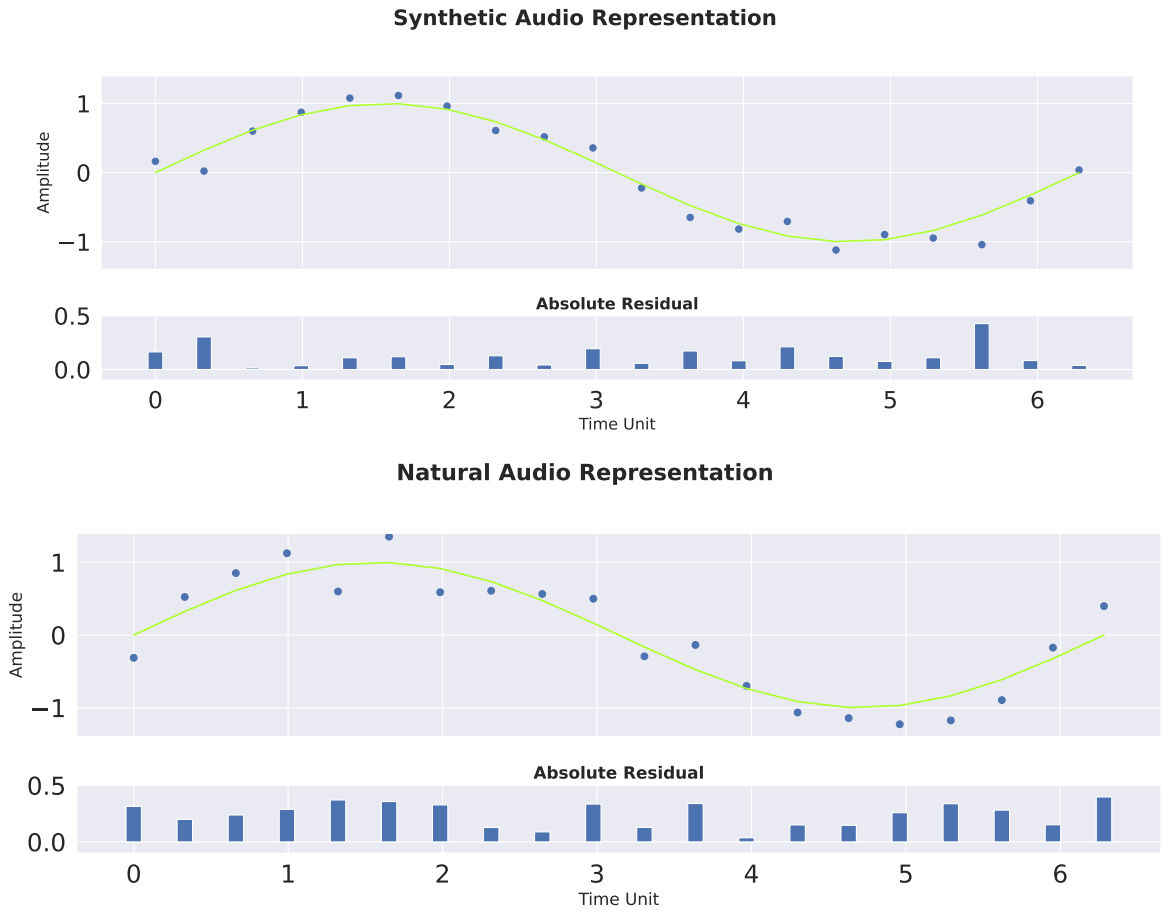

Figure 2.1. Residual Values in Synthetic and Natural Audio. Comparing the differences between a compression algorithm's predictions and actual values (i.e., the residuals) will help distinguish between fake and real.

The theory that synthetic audio has less variation than real audio is grounded in the following mathematical understanding of optimal prediction. The task of generating synthetic audio can be cast as one of predicting a sequence of audio samples using various inputs. In this way, generating synthetic audio can be viewed as a sequence of regression problems. Let Y be a random variable which represents an audio sample to predict and X a random variable which represents the coding of contextual information relevant to that prediction, such as the words spoken or the acoustic environment. The optimal predictor of Y given X has probably less variance than Y. Indeed, if X and Y are random variables, and we seek to predict Y using an observation x of X, the optimal predictor of Y with respect to mean squared error

is $\mathbb{E}[Y|X = x]$ (Shalizi 2019). Moreover, from the law of total variance

$$\text{Var}(Y) = \mathbb{E}[\text{Var}(Y \mid X)] + \text{Var}\left(\mathbb{E}[Y \mid X]\right) \geq \text{Var}(\mathbb{E}[Y \mid X]),$$

where the inequality follows from the fact that $\text{Var}(Y|X)$ is nonnegative. This shows that the optimal predictor $\mathbb{E}[Y|X]$ has less variance than the original random variable Y. Thus we expect any data driven estimate of $E[Y|X]$ to follow a similar trend. In summary, generation of synthetic audio can be cast as one of optimal prediction of a numeric quantity, and the theory of optimal prediction demonstrates that the predicted values will have less variance than authentic audio.

In practice, to test the theory that audio spoofing of a human speaker has less variation than a recording of a natural speaker, we extract residuals derived by comparing the actual samples with a smoothed representation. The algorithms for creating these smoothed representation are what we refer to as methods for audio compression. An audio compression method creates a simplified representation of the audio signal that interpolates the original samples.

If we can identify compression methods that best represent the outputs of the algorithms generating synthetic audio, the residuals for synthetic audio should have less variance. This is because the compression method's prediction of the audio will more consistently represent the pattern of the actual audio samples across time. By identifying the right compression method, we narrow the scope for what features are processed by a classifier to discriminate between synthetic and real audio.

In our research, we explore the use of wavelets as the compression method to create a smoothed representation of audio from which we extract an audio file's residuals to conduct our analysis. Our use of wavelets and its relevance to our testing for natural signal variation within audio files are explained in the next section.

2.2 Wavelet Transforms

In this section, we introduce wavelets and show how they can be used as a smoother for a sequence of audio samples. We also elaborate on the properties of wavelets that make them ideal building blocks for representing audio both locally and across the entire file. We explain that while these building blocks are limited to one type of wavelet, we can vary how closely

the wavelet reconstruction matches the pattern of the original audio by thresholding some of the wavelet coefficients within different frequency sub-bands. Although this smoothing, also referred to as denoising, alters the original audio, it reduces the requirements for digital storage since the wavelet coefficients are sparse. The last part of this section shows how different smoothing parameters create different reconstructions of an audio file, each of which can be used to derive residuals for follow-on analysis.

2.2.1 Introducing Wavelets

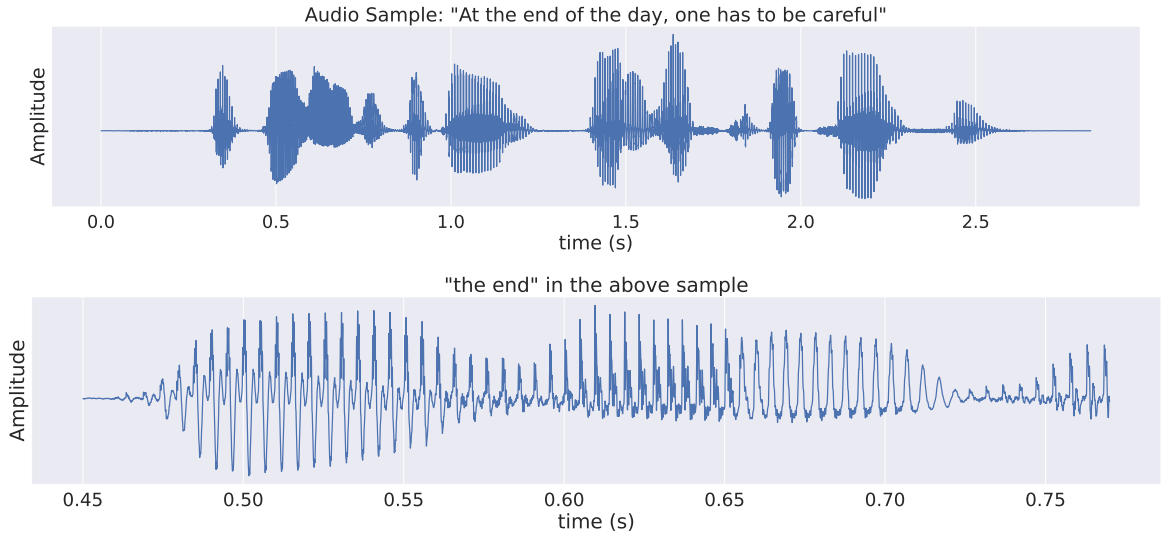

Figure 2.2. Periodic Amplitudes in Audio. As a function of time, amplitude appears wave-like, especially when viewed on a small time scale.

Wavelets are representations of short oscillations, each with different shape. They are used to match specific wave patterns along a single axis of measurement (Hellemans 2021). For audio files, this single axis is represented by time, and the variation along this axis is measured by pressure amplitudes. When an audio file is played, the sounds we hear are the oscillating pressure of air particles that stimulate the eardrum (Schremmer et al. 2000). Figure 2.2 provides a close-up of this oscillation in a waveform representation of a sound file, and highlights a wavelike nature to audio files when stored as measurements of amplitude over time.

To approximate this wavelike nature of audio, we can fit a specific type of wavelet to the overall patterns that are occurring throughout the file. There are many different types of wavelets which are distinguished by unique combinations of the frequency, the wavelength, and a specific shape. Figure 2.3 illustrates how the distinct shapes and frequency structures within wavelets are categorized into different families.

Figure 2.3. An Example of Different Wavelet Families. Source: Hellemans (2021).

The oscillatory pattern in an audio file of human and the variety of different waveforms

that can be represented using wavelets make wavelet transforms a strong candidate for both locally and generally approximating digital speech. These approximations are possible because wavelets are compact functions that vanish outside a certain interval and are well-suited to capture local variations. (Schremmer et al. 2000). In other words, wavelets can be treated like finite and sequential building blocks and pieced together to represent both high and low frequency ranges in an audio signal. This piecing together of wavelets to represent an audio signal is referred to as a wavelet transform. In the next subsection, we elaborate further on the wavelet properties that enable these transforms.

2.2.2 Discrete Wavelet Transforms

A wavelet transform decomposes a function into a set of coefficients, each of which is associated with a wavelet. These wavelets have been scaled (expanded or compacted), shifted laterally or both (Talebi 2020). Wavelets can be used in this way to uniquely decompose a function into coefficients because they are orthonormal (Cohen and Kovacevic 1996). Additionally, wavelets capture local variations in a function because each wavelet vanishes outside of a closed interval. In this way, wavelets contrast with trigonometric decompositions of a function; Figure 2.4 illustrates a wavelet that is zero outsides of a closed interval whereas a sine wave is not. Finally, a wavelet function's integral over time is zero, meaning it must be oscillatory (all positive values offset by negative values), which gives it that wave-like form (Valens 1999).

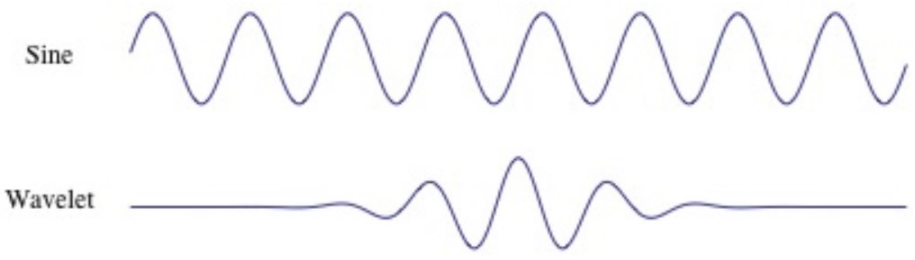

Figure 2.4. Sine Wave vs Wavelet. While sine waves are infinitely long, wavelets are localized in time. Source: Taspinar (2018).

Figure 2.5, using a symlet wavelet, shows that the time-frequency locality condition also

enables the wavelet form to be scaled such that the wavelet runs out of energy at shorter or longer durations.

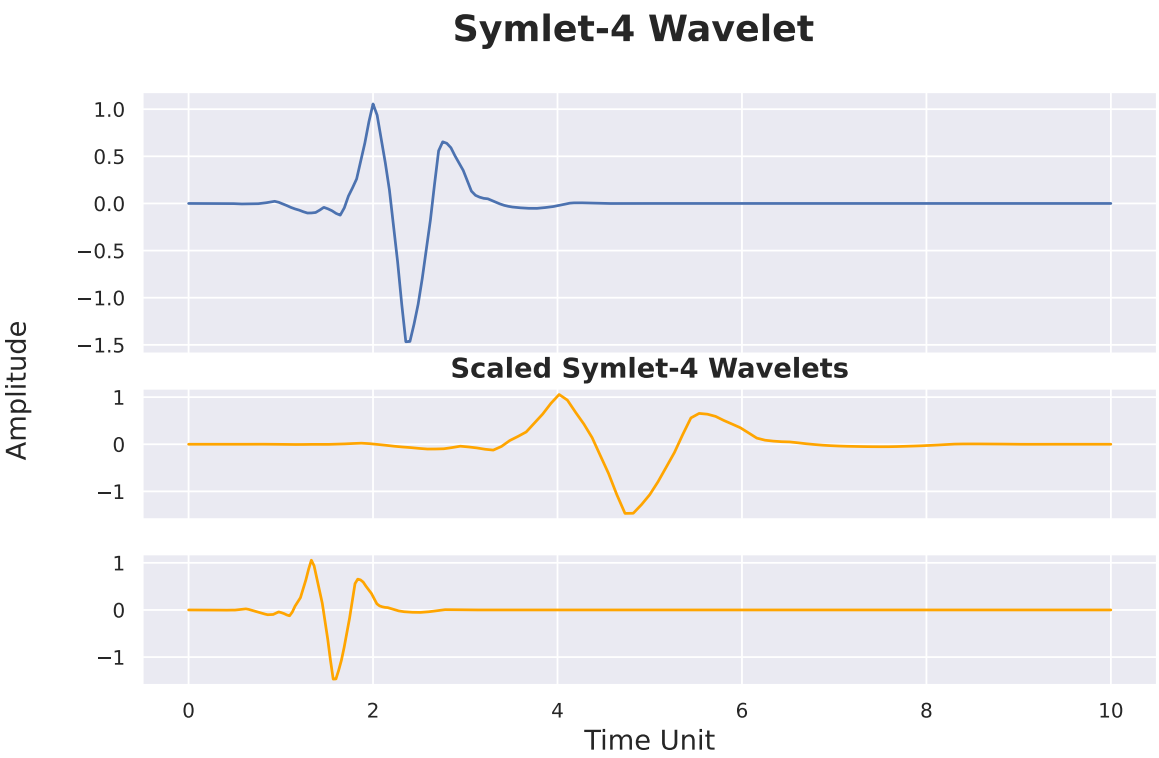

Figure 2.5. Scaling a Symlet Wavelet. Top: The wavelet has a beginning and end centered on zero. Bottom: Scaling the wavelet stretches the form to cover different time periods.

In the case of an audio file, the function to be decomposed into wavelet coefficients is the audio signal's amplitude sampled across fixed time intervals. However, this brings us to the main reason why approximating digital audio is limited to discrete wavelet transforms (DWT), the Discrete Wavelet Transform. Digital audio signals are stored as discrete samples of the waveform representation of an audio clip. In contrast to continuous wavelets, discrete wavelets are represented as a series of wavelet coefficients that are matched to specific point along the wavelet and "can only be scaled and transformed in discrete steps" (Valens 1999, p. 8). The coefficients in this type of wavelet decomposition allow one to decompose a discretely sampled function (pressure as a function of time, in our setting) into a set of discrete wavelet coefficients. By selectively thresholding these discrete coefficients, we can adjust the extent to which an audio file reconstructed from the corresponding DWT

coefficients matches the sampled points in the original audio file. This is called wavelet smoothing and is explained in the next subsection.

2.2.3 Wavelet Smoothing

Although we can use the DWT to fit a sequence of audio samples, the wavelet coefficients can only incorporate specific wavelets. For example, we cannot create a decomposition that uses coefficients from both the symlet and daubechies wavelet families. This restriction extends to wavelets belonging to the same family. While wavelets have been categorized into different families due to their similarities in shape and frequency structure, wavelets within each family are still considered distinct. Figure 2.6 provides an example of nine distinct wavelets which cannot be united in a single wavelet decomposition despite being introduced by the same researcher (Ingrid Daubechies).

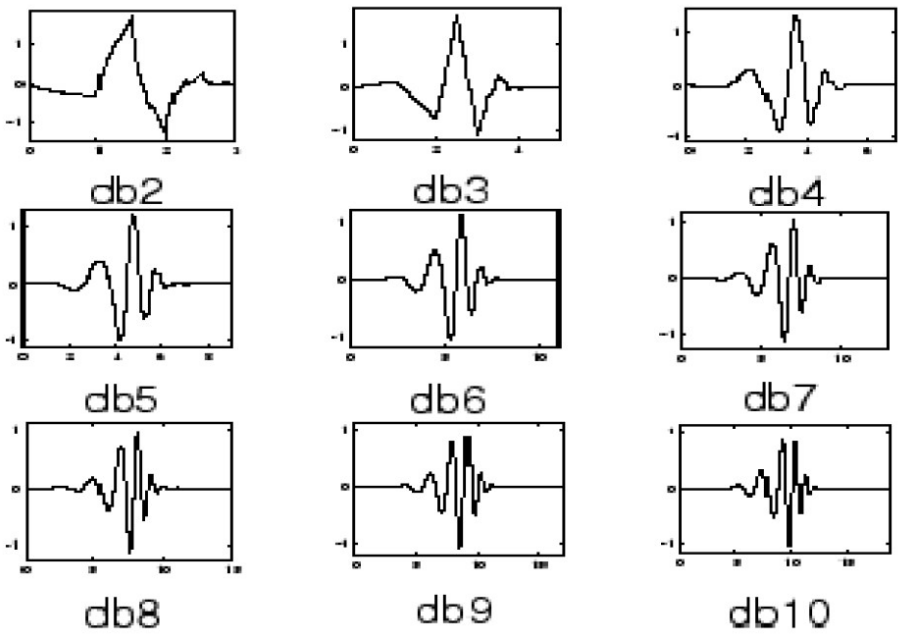

Figure 2.6. Daubechies Family. Despite their similar names, these wavelets belong to different families and therefore cannot be united in a singled wavelet decomposition. Source: Singh et al. (2011).

One key to the success of the wavelet transform is its ability to capture local oscillations of a signal at varying frequencies. This can be achieved by scaling a fixed wavelet function so that the local oscillation of the scaled wavelet occurs at the frequency of interest. This fixed wavelet, from which other wavelets are derived, is called the mother wavelet. When we scale a wavelet, we adjust the time interval on which the wavelet is nonzero. This alters the frequency at which the wavelet detects oscillations, which in turn allows us to decompose the original signal into a set of coefficients corresponding to the various scaled wavelets. Because wavelets are non-zero in a closed interval, by translating a scaled wavelet we can detect local frequency content in a signal. These translated and scaled wavelets are also incorporated into the set of coefficients representing the DWT. However, in a signal there are

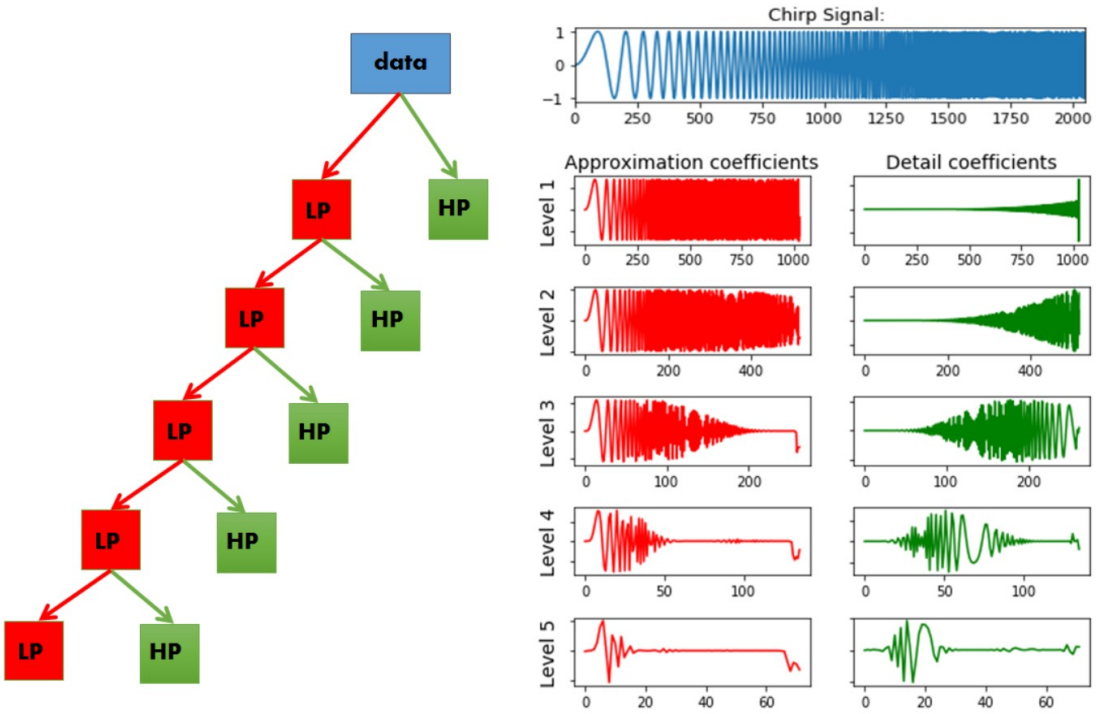

Figure 2.7. Symlet-5 Wavelet Decomposition (level 1-5) of Chirp Signal. Left: Schematic Representation of High Pass and Low Pass Filters. Right: Approximation and Detail Coefficients applied for level of the chirp signal. After reconstruction, the final wavelet is represented by the detail coefficients for all levels and the approximation coefficients for the last level. Source: Taspinar (2018).

potentially many frequencies which a scaled wavelet may need to fit. Fortunately, the DWT, where the original signal is discretely sampled in time, limits the range of discrete wavelet scalings/translations by passing the signal through a sequence of bandpass filters. After passing through the filter, we apply the DWT to the filtered signal at a reduced frequency band. For each frequency band, we extract a new array of coefficients, all of which contribute to the overall DWT's filter bank of coefficients. In this way, a discrete wavelet can be stored as a filter bank with arrays of coefficients corresponding to different frequency bands. This process is illustrated in Figure 2.7.

To better understand scaled wavelets' role in frequency bands, we use Figure 2.7 to illustrate applying a symlet-5 discrete wavelet transform to a signal whose frequency increases with time, also known as a chirp signal. This illustration will also help explain our use of two parameters, decomposition levels and thresholding, that provide the flexibility for smoothing a wavelet that was introduced in the last paragraph. Because the chirp signal increases frequency over time, it helps to illustrate how the wavelet decomposition level for a frequency band corresponds to a sequence of applications of low pass and high pass filters.

We note that every level increase results in halving the number of coefficients, both for the approximation and detail coefficients. The number of coefficients decrease because when you halve the frequency band, you only need to sample every other point along the array to analyze the frequency behavior in that band. At each level, a high pass and low pass filter are applied to split the frequency band in half between high and low frequencies after the wavelet is fit to the original signal.

The filter ensures that the lower band of frequency (low pass) still represents the wavelet transform but only at that lower band of frequency. The *detail coefficients* (high pass) correspond to the values that were extracted from the *approximation coefficients* representing the DWT deconstruction at an earlier level. The remaining approximation coefficients at the low pass then represent the DWT within the lower frequency band. If we increase the level of decomposition, another DWT is applied to the current low pass approximation coefficients and a new series of low pass (approximation coefficients) and high pass (detail coefficients) filters will be represented at the next level.

Applying a wavelet transform to a chirp signal in Figure 2.7 helps us visualize the partitioning of frequency activity because we notice the detail coefficients shifting further to the left as we increase the number of decomposition levels. This is explained by the fact that higher levels of decomposition mean that wavelet transform activity is represented at a lower frequency band. Since the chirp signal increases frequency over time, the chirp signal's high frequency values in later time periods are not being analyzed because they exceed the low pass frequency band at the higher levels of decomposition. In fact, the decreasing activity in detail coefficients show how there is less and less detail at higher levels because there is less frequency information to remove. Therefore, we can see that at higher levels of decomposition, more information about the DWT is captured at the lower levels of frequency.

This frequency information, stored as coefficients in the DWT's filter bank for frequency bands, to is used to reconstruct the audio signal. We can change the reconstructed signal by selectively adjusting the coefficients prior to reconstruction in a process called *thresholding*. During reconstruction, the detail coefficients provide additional information about the wavelet that is not represented by approximation coefficients from the lowest frequency band. This thresholding of detail coefficient values removes frequency information about the wavelet and the resultant smoothing of the wavelet peaks is also referred to as denoising.

This thresholding of wavelet coefficients at different decomposition levels can be determined by setting a thresholding parameter. If the thresholding parameter value is greater than the absolute value of the detail coefficients, then the detail coefficient is set to zero. If the thresholding value is less than the absolute value of the detail coefficient, then the detail coefficient is either left unaltered (hard thresholding) or shrunk towards zero (soft thresholding). The more detail coefficients that are set to zero, the more aggressively we smooth the original sample. This can happen with both large values of the thresholding parameter and large levels of decomposition because at large levels of decomposition, the approximation coefficients (untouched by the threshold parameter adjustments) are using a lower frequency band corresponding to less coefficients in the decomposition. This is illustrated in Figure 2.8 and shows how different levels and threshold parameters impact a DWT's reconstructed signal relative to the original audio samples.

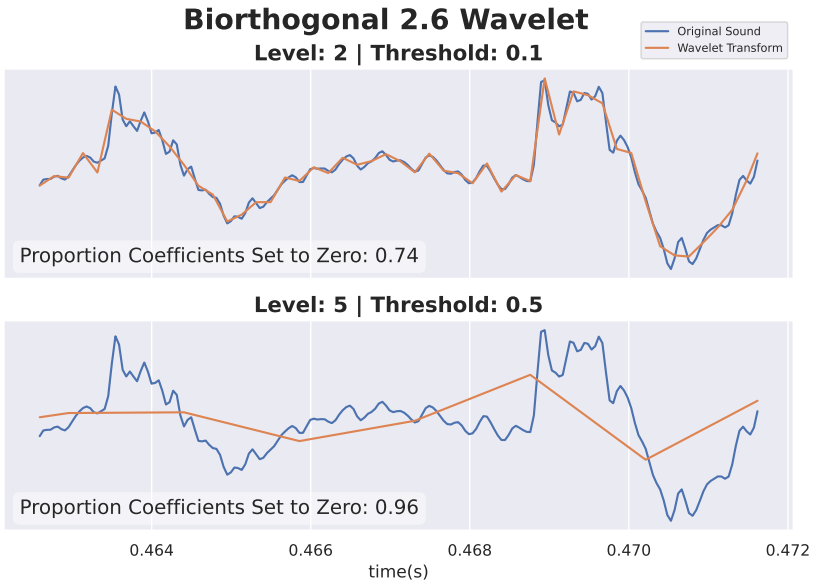

Figure 2.8. Thresholding/Level Comparisons. Higher decomposition levels and larger threshold parameters result in more detail coefficients being set to zero. This diminishes how closely the DWT fits the original audio sample.

As Figure 2.8 illustrates, our choice of smoothing parameters gives us a lot of flexibility for how to reconstruct a DWT that can provide different approximations for patterned activity within an audio file. This extends naturally to how every audio file allows us to extract and consolidate residuals for further analysis. Figure 2.9 shows this by contrasting the differences between an audio clip's original signal and the reconstructed signal from a Haar DWT. For every sampled point in the audio file, we extract a residual that, when plotted on a histogram, forms a unimodal distribution.

While Figure 2.8 demonstrated how different smoothing parameters led to different approximations of an audio file using the same mother wavelet, other distinct wavelets can be used to approximate audio, but with a different pattern of residuals. For example, the symlet-2 wavelet in Figure 2.9 reduces the observed residuals in Figure 2.10 despite using the same thresholding and level parameters used in the Haar DWT.

As we can see, different wavelets can be used to create different approximations of the sequence of sampled amplitudes contained in a digital audio file. Additionally, Figures

2.9 and 2.10 both show how an audio file's residuals can be aggregated into unimodal distributions with distinct statistical properties despite their similarities in appearance.

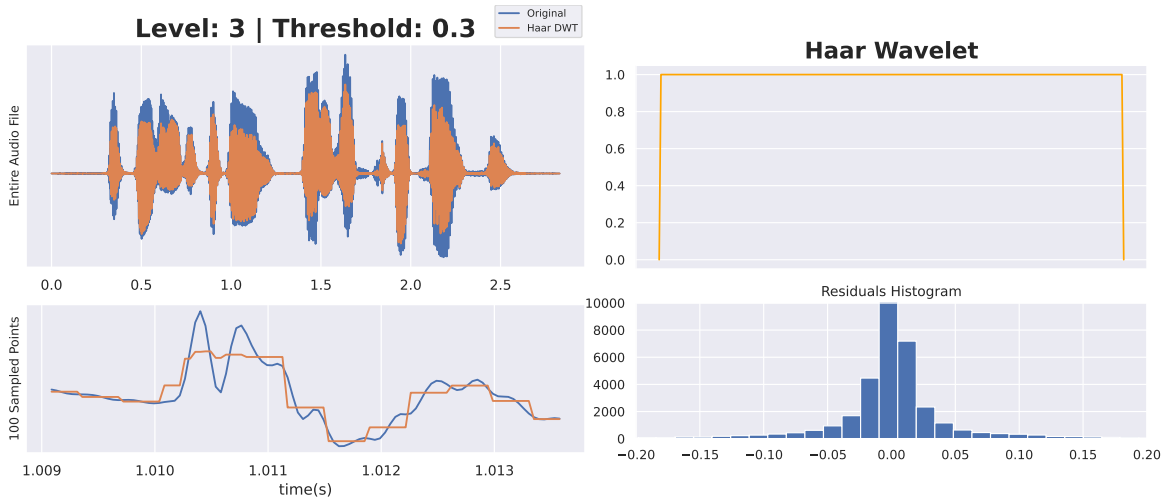

Figure 2.9. Haar DWT to Audio Sample. Aggregated residuals between the Haar DWT reconstruction and the actual audio signal form a unimodal distribution centered around zero.

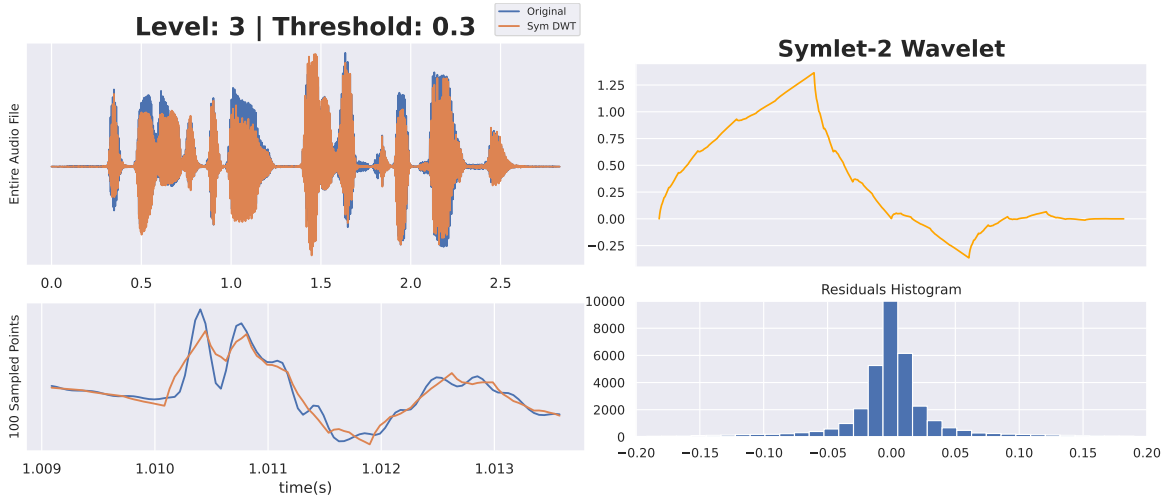

Figure 2.10. A Closer Approximation Using the Symlet-2 DWT. Fitting a wavelet with greater variation in its peaks reduced the overall residuals despite keeping the same smoothing parameters.

2.2.4 Extracting Residuals

In this section, we provided a brief background on wavelets and their utility in approximating the pattern of sampled amplitudes in a digital audio file. We also showed how we can adjust parameters in the wavelets to create unique approximations of the audio file. As a result, we now have many ways for collecting unique patterns of residuals from a sampled audio file. This variability is important because each approximation represented by a specific wavelet transform may be capturing different predictive elements within the audio file. In the next section we introduce and explain the classifier that we use to analyze these residuals.

2.3 Gradient Boosted Trees

As Figures 2.9 and 2.10 show, every audio file can be reduced to a distribution of residual values. To use this information, we need to train a classifier using supervised machine learning (SML). Prior to classifying unknown data, SML learns how to map inputs x to outputs y by using algorithms that analyze previous samples with known values for x and y. For this research, y is the probability that the given audio sample is fake. During training, every y_i is assigned either a probability value of 0 or 1 according to its known classification. The goal of the classifier is to approximate a function f with x inputs such that $y = f(x)$. While y is restricted to a scalar value between 0 and 1, x is a vector of components $[x_1, x_2, ..x_n]$ called features and in our setting captures statistical properties of the residuals' sample distribution. We extract the mean, variance, skew, kurtosis, quartiles and min/max values from the residuals. In this paper, we will use GBM to learn how these features x can be mapped onto a probability y that measures to what extent a given audio file is synthetically generated.

GBM, "as the de facto choice of ensemble method," has been widely popularized in data-science competitions "to capture the complex data dependencies and scalable learning systems that learn the model of interest from large datasets" (Chen and Guestrin 2016, p. 785). As an ensemble method, GBM uses classification and regression tree (CART) models to combine the predictions from weak classifiers into a stronger prediction (Opitz and Maclin 1999). However, GBM is distinct from another popular ensemble CART random forest model because the overall prediction is not aggregated from simultaneous tree-structure prediction models grown from the same data distribution (Breiman 2001). Rather, the GBM

sequentially trains, or "boosts," a new CART model that will incrementally improve the model's prediction result because it is trained on the previous iteration's prediction error (Freund and Schapire 1997).

Algorithmically, this means throughout the GBM model, there are consistent rules for governing how the GBM selects CART trees prior to each training iteration (Torlay et al. 2017). These rules are generalized into three tuning parameters that determine how many CART models to grow, the level of complexity within each CART model, and the learning rate (or loss function) of the overall model (i.e., how quickly the residuals are reduced for every iteration). In their paper titled, *XGBoost: A Scalable Tree Boosting System*, Chen and Guestrin (2016) formalize the relationship between learning rate and complexity using the following function L representing the "regularized learning objective."

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

On the right hand side of the equation, the following expression

$$\sum_i l(\hat{y}_i, y_i)$$

refers to the loss function which measures the difference between prediction $\hat{y}_i$ and the actual value y_i . Contained within the function l are "learning measures" for governing how quickly to reduce the error every iteration. These learning measures determine to what extent each decision tree remains a "weak hypothesis learner that is slightly better than random guessing" (Freund and Schapire 1997, p. 120). In statistical learning, models that learn slowly tend to perform better because they are resistant to overtraining.

Within the equation,

$$\sum_k \Omega(f_k)$$

addresses the two parameters for k-number of CART models and the subsequent regular-

ization of tree complexity, $\Omega(f_k)$. $\Omega(f_k)$ is broken up into

$$\gamma T + \frac{1}{2}\lambda \|w\|^2$$

where the penalty parameter γ is assigned to the number decision leaves T in the CART and the penalty parameter λ is assigned to the w vector of scores for every leaf in CART k.

The penalties in the regularization portion of the equation "help to smooth the final learned weights to avoid over-fitting" and support selecting a model "employing simple and predictive functions" (Chen and Guestrin 2016). In their article, Chen and Guestrin (2016, p. 786) explain how the XGBoost model uses algorithms for selecting tree-splits, handling incomplete data within features per observation, and how their algorithms enable "parallel and distributed computing. . . which enables quicker model exploration." The last feature, along with historical success in previous data competitions, is a strong motivator for our use of the XGBoost GBM classifier for our research.

To summarize Chapter 2, we have explained our theory for why we expect algorithmically generated synthetic audio to have less natural variation than a real human voice. Additionally, we described our use of DWT to reconstruct audio signals that closely approximate the original audio. By comparing the reconstructed signal and original signal we extract the residuals from which we can assess the audio file's natural variation. Lastly, we provided a theoretical background for the GBM classifier that we use to extract patterns from audio samples with known fake/real classifications in a train set. This trained classifier then analyzes a test audio file's summary statistics from its distribution of residuals to assign a probability declaring to what extent the test file is assessed to be synthetically generated. In Chapter 3, we will explain how we executed these processes using select packages within the python programming language and how they informed our experimental results.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Experimentation and Results

Chapter 2 described how we use the discrete wavelet transforms (DWT) and its smoothed reconstruction of a digital audio file to extract that audio file's residuals. We then explained our use of the gradient-boosted tree model (GBM) classifier to analyze the residuals and thereby determine the probability that audio file has been synthetically generated. In this chapter, we apply this methodology to two datasets and assess our ability to distinguish between synthetically manipulated audio and audio files capturing the recorded voice of authentic human speakers.

Section 3.1 describes what packages we use in the Python programming language for applying DWT to digital audio samples and for implementing the GBM to analyze a residual distribution's summary statistics across thousands of samples. Section 3.2 analyzes the 2019 Fake-or-Real Dataset introduced by Reimao and Tzerpos (2019), and Section 3.3 analyzes the data from the 2015 ASVSpoof Competition. Lastly, Section3.4 uses the third section's dataset to build a classifier that decisively demonstrates that the recorded voice in the March 16, 2022 deepfake video calling for Ukrainian troops to surrender did not originate from the Ukrainian president Volodmyr Zelensky.

3.1 Python Implementation

This section explains the three primary python programming packages that we used to conduct our experimentation: **librosa**, **pywavelets**, and **xgboost**.

3.1.1 Librosa

We used the **librosa** package to convert the various digital audio formats, such as *.flac* or *.wav*, into an array of sampled points corresponding to amplitude values (McFee et al. 2015). The **librosa** python package not only has the advantage of converting multiple audio formats to a single-channel array, but also normalizes the information in the array and therefore allows us to downsample all digital files to a consistent sampling rate. For example,

both the Fake-or-Real (FoR) and 2015 Automatic Speaker Verification and Spoofing Countermeasures Challenge (ASVSpoof) datasets provide *.wav* files sampled at a rate of 16000 measurements per second. Audio for the deepfake analysis, however, was downloaded from YouTube and had a sample rate of 44100 measurements per second. The **librosa** package allows us to ensure that our experimentation methods are applied to audio files with a uniformly consistent sampling rate.

3.1.2 Pywavelets

To create a smoothed wavelet representation of an audio file's original array of sampled amplitudes, we used the **pywavelets** package (Lee et al. 2019). This open source python package enables us to convert an array of amplitude values into DWT coefficients. We then apply the soft thresholding operator (described in Section 2.2.2) to the wavelet coefficients. Afterwards, **pywavelets** processes those adjusted coefficients to reconstruct an approximation of the original audio file. **Pywavelets** provides 105 distinct wavelet families which can be used to perform the DWT. We provide the names for these wavelet famileis in Table 3.1.

Table 3.1. Pywavelets Discrete Wavelet Families. Pywavelet provides the capability to apply 105 different DWTs to one digital audio file. Each wavelet family's reconstruction of audio can be adjusted by thresholding the coefficients of the DWT.

Wavelets Families	
Biorthogonal 1.{1,3,5}	Reverse Biorthogonal 1.{1,3,5}
Biorthogonal 2.{2,4,8}	Reverse Biorthogonal 2.{2,4,8}
Biorthogonal 3.{1,3,5,7,9}	Reverse Biorthogonal 3.{1,3,5,7,9}
Biorthogonal 4.4	Reverse Biorthogonal 4.4
Biorthogonal 5.5	Reverse Biorthogonal 5.5
Biorthogonal 6.8	Reverse Biorthogonal 6.8
Coiflet {1,2,...,17}	Haar
Daubechies {1,2,...,38}	Discrete Meyer
Symlet {2,3,...,20}	

3.1.3 XGBoost

After extracting a residual for every sampled point in an audio file, we examine the distribution of residuals and acquire the following statistical features: mean, standard deviation, min, quartiles, max, skew and kurtosis. From this information, we build a dataframe where every row corresponds to an audio file and each column gives ones of these statistical features. This dataframe is processed into the gradient-boosted tree model (GBM) using the **xgboost** python package (Brownlee 2019). To train the classifier, we created an 80/20-split of train files consisting of audio files with known fake/real labels. This 80/20-split enables **xgboost** to iteratively improve on the errors in the 20% of train files set aside as a test set. After training the model, xgboost examines a new dataframe of residuals for audio files with unknown labels and assigns a scalar value between 0 and 1 to each audio file in the validation set. A larger value indicates the classifier is increasingly confident that the audio file has been synthetically generated.

The next three sections explain how we use these packages to analyze different sets of data. The general pattern for analysis begins by using the audio files with known synthetic/authentic classifications to identify a set of wavelet parameters that perform well on a test set. However, each dataset has different information available about the audio files. For example, one dataset indicates the spoofing method used to synthetically generate the audio sample and identifies the target speaker. Other datasets may contain speech files from different speakers but provides no information on which speaker is associated with each audio file. The next section looks at a dataset that, while using the latest synthetic audio generation methods, does not provide these speaker level details.

3.2 Fake-or-Real

Of the two datasets we assess in this thesis, only the FoR dataset introduced by Reimao and Tzerpos (2019) uses synthetic audio generated from deep-learning algorithms. Reimao and Tzerpos (2019) originally created this dataset because they determined that the current datasets for training complex classifiers had several deficiencies. Previous datasets were primarily focused on detecting spoofed utterances targeted at bypassing ASV systems. Using only speaker/attacker datasets when training the classifier potentially biases the classifier to rely on speaker-specific characteristics as opposed to general patterns of speech.

Consequently, the classifier may have more difficulty generalizing to datasets with unknown speakers. Additionally, the researchers assessed that the current sets of utterances were "typically not sufficient to train complex neural network models" because they did not cover a broad enough spectrum of speech (Reimao and Tzerpos 2019, p. 3). Lastly, Reimao and Tzerpos (2019) wanted to provide a dataset using state-of-the-art speech algorithms not necessarily tied to a spoofing attack. In summary, the researchers wanted to create a dataset where speaker data facilitates building classifiers which are agnostic to both speaker identity and generation mechanism. Given this background, our experiments for this dataset will not attempt to distinguish between methods or speakers but focus on assessing whether our classifiers identify which files are fake or real in their respective train, test, and validation sets. We divide our exploration of this dataset into three parts. First, we explain the primary features of these data and how they are structured. Second, we articulate our process for choosing which wavelet family and corresponding set of parameters will be used to assess the validation set. Lastly, we show the results of those experiments on the files in the validation set.

3.2.1 The Dataset

When building the FoR dataset, Reimao and Tzerpos (2019) wanted to ensure that classification models were not being trained on specific words/phrases but on the generalized differences between synthetic and real speech. Therefore, despite the overall dataset containing over 150,000 English phrases, there are no repeated utterances or phrases in the dataset. The audio files corresponding to real speakers that provided voice samples for the following well-known open source datasets: Arctic Dataset (Anumanchipalli et al. 2011), LJSpeech Dataset (Ito and Johnson 2017), and the VoxForge Dataset (*VoxForge* 2019). The researchers selected phrases from a variety of speakers from both genders and various accents. The quality of speakers ranged from professional voice actors to online volunteers independently recording and submitting utterances to a dataset.

Because the FoR dataset creators were not focused on classifiers targeting specific spoofing attacks, the synthetically generated audio files in the FoR dataset only originated from TTS methods where spoken phrases are generated from manual keyboard entries. The TTS methods represented a total of seven synthetic generative voice algorithms representing 33 voices. All synthetic audio samples were acquired from commercial computing companies

such as Baidu, Google, AWS and Microsoft. To provide multiple data points of contrast between a variety of generative speech algorithms at varying stages of advancement, the researchers ensured the synthetic samples in their data widely ranged in natural sounding quality. For example, while Baidu Labs allows for open source distribution of its DeepVoice3 Wavenet algorithm, it also offers a commercial Baidu Cloud TTS service using proprietary software. While Google did not release an open source code for its TTS service, people can easily create synthetic audio clips using the Google Translate website. However, Google also offers a premium Google *Cloud TTS with Wavenet* service for creating more natural sounding audio. Microsoft and AWS offer similar services and marketing strategies as well. The FoR Dataset contains synthetically generated audio from all these methods.

In addition to the sampled audio phrases being acquired from a wide variety of synthetic and real digital voices, the authors standardized each audio file to a sample rate of 16 kHz (16000 measurements per second), stored as a single channel *.wav* file. The authors also cropped each file to ensure no silences at either the beginning or end of the audio file. While both synthetic and real audio samples consisted of complete utterances, the synthetic audio samples were consistently shorter in length. To address this, researchers created another version of the dataset using only 2-second truncations for both synthetic and real audio files. Lastly, each version of the dataset was balanced so that genders, speakers and generation methods were evenly distributed. When conducting our experiments, we trained the classifier on the train folder and validated our classifier on the test folder for both versions of the FoR dataset. Table 3.2 summarizes this data below:

Table 3.2. Number of Files in the Normalized and 2-Second Versions of the FoR Dataset. Audio files in all folders have an even distribution of speakers, and genders. We use the train folder to identify suitable wavelet families and parameters.

		Normalized	**2-Sec Truncated**
Train Folder	Synthetic	26927	6978
	Real	26939	6978
Test Folder	Synthetic	2370	544
	Real	2264	544

For this dataset's series of experiments, the train folder contained synthetic audio files using only 6 of the 7 TTS generation methods. While the test folder (on which we validate our classifier) contained utterances from real speakers who were not in the train folder, all the synthetic audio test files were created using the remaining, and arguably most advanced, TTS method, which is Google's *Cloud TTS with Wavenet*. The challenge for this dataset, therefore, was to build a classifier capable of detecting and generalizing to unseen TTS algorithms set apart from the train folder.

3.2.2 Selecting Best Parameters

Prior to applying trained classifiers to the validation set of audio files in the test folder, we needed to identify wavelet families, decomposition levels, and thresholding parameters with success in the train set that generalizes to the test folder. To do this, we conducted a test on 800 randomly sampled files within the 2-sec Truncated Train folder. This set had an equal number of fake and real files. For each of these tests, we trained a classifier using 400 files and subsequently validated the classifier on the remaining 400 files in the subset. We scored the results of the mini validation by taking the proportion of files that were correctly classified.

We individually applied this test to the 2-sec Truncated Train Folder subset of 800 files using 105 wavelet families. For every wavelet, we tested levels for decomposition from three through six. Within every level, we examined the 45 different thresholding parameters. The first ten thresholding parameters ranged from 0.01 and 0.10 and were evenly spaced by .01. The remaining 35 parameters ranged from 0.15 to 0.5 and were separated by 0.1. In summary, for 400 files in the 2-sec Truncated Train Folder, we conducted 14,700 experiments that individually returned scaler values between 0 and 1. These scalar values informed us to the extent to which a classifier trained using the given parameters and wavelet family generalizes to a different set of 400 files within the same folder.

From these 14,700 experiments, we needed to identify a subset of wavelet/parameter candidates that had the best potential for detecting the unseen Google's *Cloud TTS with Wavenet* speech algorithm used in the FoR test set. To do this, we first grouped the wavelets in the following categories based on their names and specific shapes.

- Group 1: Biorthogonal
- Group 2: Coiflets
- Group 3: Daubechies
- Group 4: Reverse Biorthogonal | Haar | Discrete Meyer
- Group 5: Symlets

Next, for every level and for every wavelet group, we graphed the wavelet's performance on the 400 files as a function of all the thresholding parameters in that wavelet. To keep this graph from being too busy, we have only included wavelet families which correctly classified at least 320 files of the 400 considered for at least one value of threshold parameters. Figure 3.1 is an example of the Reverse Biorthogonal grouping using a Level 6 decomposition.

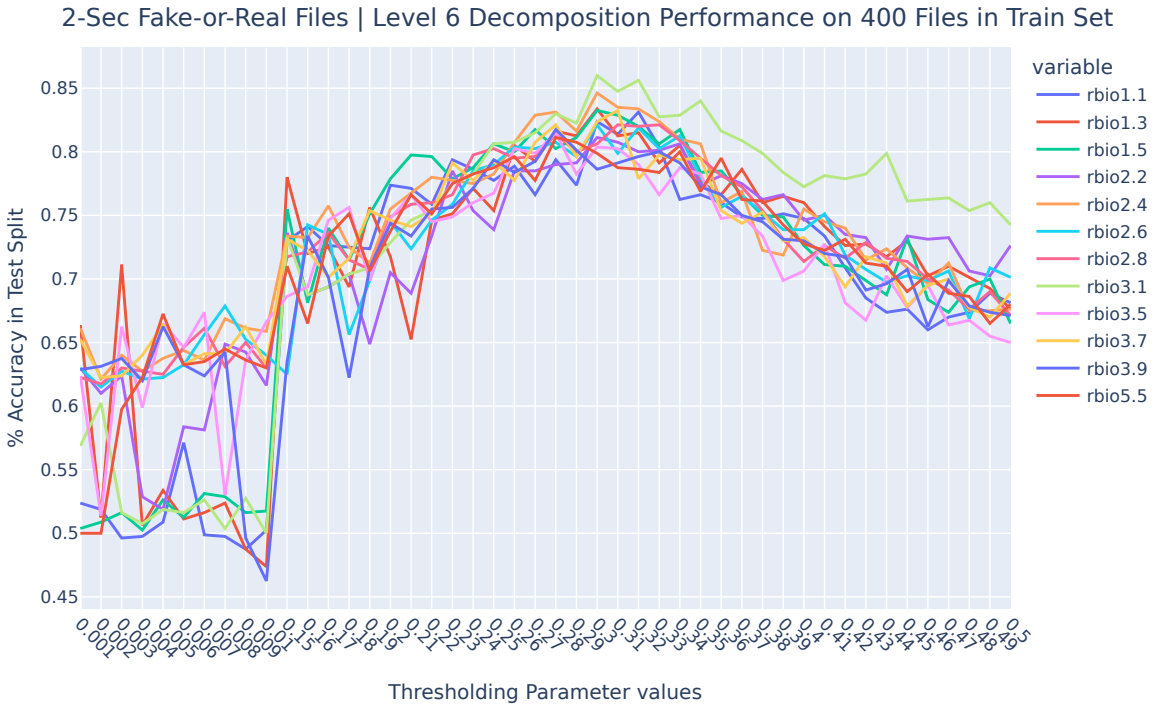

Figure 3.1. FoR Wavelet Candidates for Validation Set. Of the 17 wavelets in this Reverse Biorthogonal Grouping, 12 wavelets correctly classified more than 80% of the audio files in the train set for at least one of the thresholding parameters at Level 6 decomposition after being trained on the 400 files in the same folder.

Figure 3.1 represents one of potentially 20 different graphs (5 wavelet groupings across 4 Levels of decomposition) that we could examine to identify candidates for building a classifier to challenge the validation set of files in FoR's test folder. We note, however, an important trend where the relative success for each wavelet/parameter peaks around a thresholding value near or around 0.3. This trend is consistent for other wavelet groupings and across other levels. In Table 3.3, we show how many wavelet/parameter candidates achieve a success rate of at least 80% for the 400 audio files in the test set.

Table 3.3. Number of Wavelet Candidates. At Level 5 decomposition, there appears to be the most wavelet/parameter combinations that successfully classified at least 80% of the 400 test files in the train folder.

	Group 1	**Group 2**	**Group 3**	**Group 4**	**Group 5**
Level 3	12	1	0	3	0
Level 4	25	0	23	37	1
Level 5	82	180	321	149	148
Level 6	40	25	43	75	31

Table 3.3 shows that if we only consider wavelet/parameter combinations with at least an 80% success rate, there are still 1196 candidates. We also notice that most candidates among all the wavelet groupings are at decomposition level 5. However, we do not yet know how well these wavelets generalize to the unknown TTS method used to generate synthetic audio files in FoR's validation set. Therefore, instead of focusing on the decomposition level with the greatest number of wavelet/parameter candidates, we build classifiers using the top performing wavelet/parameter combination for each subgrouping. Our first round of classifiers to assess the audio files in FoR's validation set will be built using the following wavelets with their corresponding parameters and are displayed in Table 3.4.

3.2.3 Validation Set Results

In contrast to the classifiers that were built in Section 3.2.2 for the sake of choosing training parameters, the classifiers assessing the validation set of files in FoR's test folder will be trained on all the audio files in the Normalized and 2-Sec Truncated train folders. For every

Table 3.4. Initial Wavelet/Parameters for Classifier. In this table, each wavelet and its corresponding combination of parameters will be used to build a classifier that assesses the synthetic nature of audio files in the FoR's test folder. Of the wavelet categories defined by Level/Group intersection in Table 3.3, the wavelet/combinations in this table had the highest percentage accuracy scores.

Wavelet	**Level**	**Thresholding**	**% Correct**
Biorthogonal 1.3	L3	0.31	0.8175
Coiflet 14	L3	0.32	0.80125
Reverse Biorthogonal 3.1	L3	0.26	0.8125
Biorthogonal 1.1	L4	0.28	0.8225
Daubechies 1	L4	0.28	0.8225
Reverser Biorthogonal 1.1	L4	0.28	0.8225
Symlet 2	L4	0.3	0.81875
Biorthogonal 2.8	L5	0.29	0.84
Coiflet 2	L5	0.25	0.85625
Daubeches 19	L5	0.27	0.85625
Reverse Biorthogonal 3.1	L5	0.31	0.85
Symlet 15	L5	0.27	0.8475
Biorthogonal 1.1	L6	0.32	0.83125
Coiflet 1	L6	0.3	0.82125
Daubechies 1	L6	0.32	0.83125
Reverse Biorthogonal 3.1	L6	0.3	0.86
Symlet 9	L6	0.29	0.81875

wavelet and its corresponding set of parameters in Table 3.4, we train two classifiers. The classifier trained using the 53,866 audio files in the Normalized Train Folder will use the 4,634 audio files in the Normalized Test Folder as the validation set. Correspondingly, the other classifier, trained on the 13,956 audio files in the 2-Sec Truncated Train Folder, will use as a validation set the 1088 audio files in the 2-Sec Truncated Test Folder.

However, even though we train two classifiers using two variations of the dataset, our determination on the best wavelet/parameters will be based on the combined area under the relative operating characteristic (ROC) curve (AUC) scores from the classifiers' performance on their respective datasets. We justify combining the AUC scores because the audio files within their respective dataset variations are only distinguished by their differing lengths.

In fact, because the normalized variation of the dataset uses synthetic audio utterances that are consistently shorter in length than the real speaker audio samples, we anticipate the classifiers assessing the normalized dataset to perform better on average. The results for Table 3.4's initial selections of wavelet/parameter combinations for classifiers are displayed in Figure 3.2 and Table 3.5.

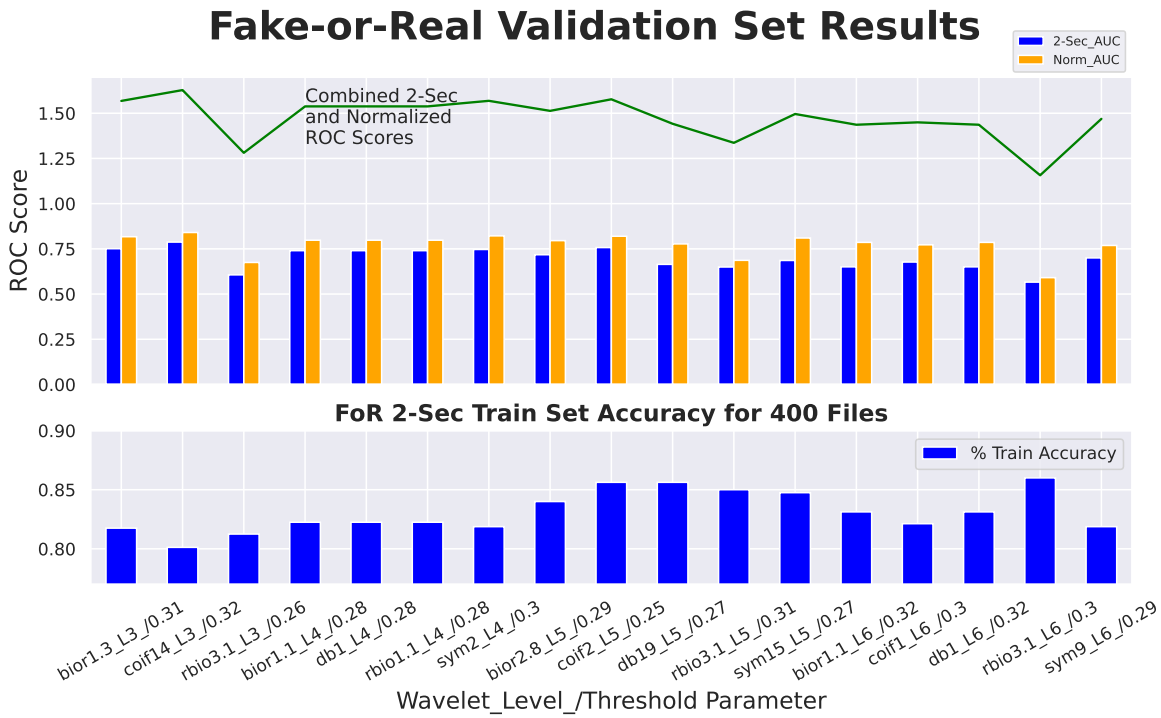

Figure 3.2. Initial Results on Fake-or-Real Validation Set.Despite the coiflet-14 at Level 3 decomposition having the lowest accuracy in the 2-Sec Train Set, it has the highest combined AUC score of the 2-Sec and Normalized Validation Sets.

Both Figure 3.2 and Table 3.5 confirm our intuition that the 2-Sec Validation AUC scores consistently perform worse than the Normalized AUC scores. The on-average shorter lengths of the synthetic files in the normalized FoR version are being used by the classifier to more accurately identify fake files. We also notice that while the reverse biorthogonal-3.1 wavelet at Decomposition Level 6 performed the strongest in the train set, its AUC scores in the validation sets were by far the worst of all the wavelet combinations. This suggests that the

Table 3.5. Initial Results on Fake-or-Real Validation Set. The green and red cells correspond to the highest and lowest scores in their respective columns. The wavelet/parameter combination with the strongest performance in the train set scored the lowest in the validation set and vice-versa.

Wavelet	Level	Thresholding	% Train Accuracy	Norm_ROC	2-Sec_ROC
Biorthogonal 1.3	L3	0.31	0.8175	0.81721	0.750846
Coiflet 14	L3	0.32	0.80125	0.840549	0.787449
Reverse Biorthogonal 3.1	L3	0.26	0.8125	0.674878	0.606251
Biorthogonal 1.1	L4	0.28	0.8225	0.797794	0.739986
Daubechies 1	L4	0.28	0.8225	0.797794	0.739986
Reverser Biorthogonal 1.1	L4	0.28	0.8225	0.797794	0.739986
Symlet 2	L4	0.3	0.81875	0.821933	0.746607
Biorthogonal 2.8	L5	0.29	0.84	0.795525	0.717518
Coiflet 2	L5	0.25	0.85625	0.819873	0.75715
Daubeches 19	L5	0.27	0.85625	0.777416	0.664269
Reverse Biorthogonal 3.1	L5	0.31	0.85	0.68658	0.649664
Symlet 15	L5	0.27	0.8475	0.809961	0.685829
Biorthogonal 1.1	L6	0.32	0.83125	0.785834	0.650507
Coiflet 1	L6	0.3	0.82125	0.772395	0.677102
Daubechies 1	L6	0.32	0.83125	0.785834	0.650507
Reverse Biorthogonal 3.1	L6	0.3	0.86	0.590846	0.566043
Symlet 9	L6	0.29	0.81875	0.768839	0.700011

combination was overtrained to known TTS methods in the train folder and was unable to generalize to the unknown TTS method in the test folder.

The most interesting observation, however, was the strong performance of the coiflet-14 wavelet (Group 2) at Level 3 decomposition on the validation files in the test folder. If we look back at Table 3.3, the only wavelet/parameter combination to make the cutoff in Group 2 at Level 3 decomposition was the coiflet-14 wavelet. When discarding the Norm-AUC scores below 0.7, we also note that the drop off in performance from the Norm AUC score to the 2-Sec AUC score is also the least when using the coiflet-14 combination.

With these observations in mind, we lowered the cutoff value for Group 2 at Level 3 decomposition to 75% accuracy and took the top 25 train set coiflet/parameter combinations that correctly classified at least 300 files in the train folder using the tests described in Section 3.3.2. Additionally, we expanded the window of wavelet/parameter candidates to include the top 40 train-set performers across all 4 Levels of decomposition and did not limit it to the 5 wavelet groupings. Using this expanded set of wavelet/parameter combinations, we individually trained new classifiers on the both the 2-Sec Truncated and Normalized Train folders and tested the classifiers on the validation set in the FoR test folders. To visualize

the comparison between wavelet/parameter combinations we sorted the performance by combined 2-sec/normalized AUC values. Figure 3.3 shows the resulting stacked barplot of combined AUC scores and wavelet/parameter combinations.

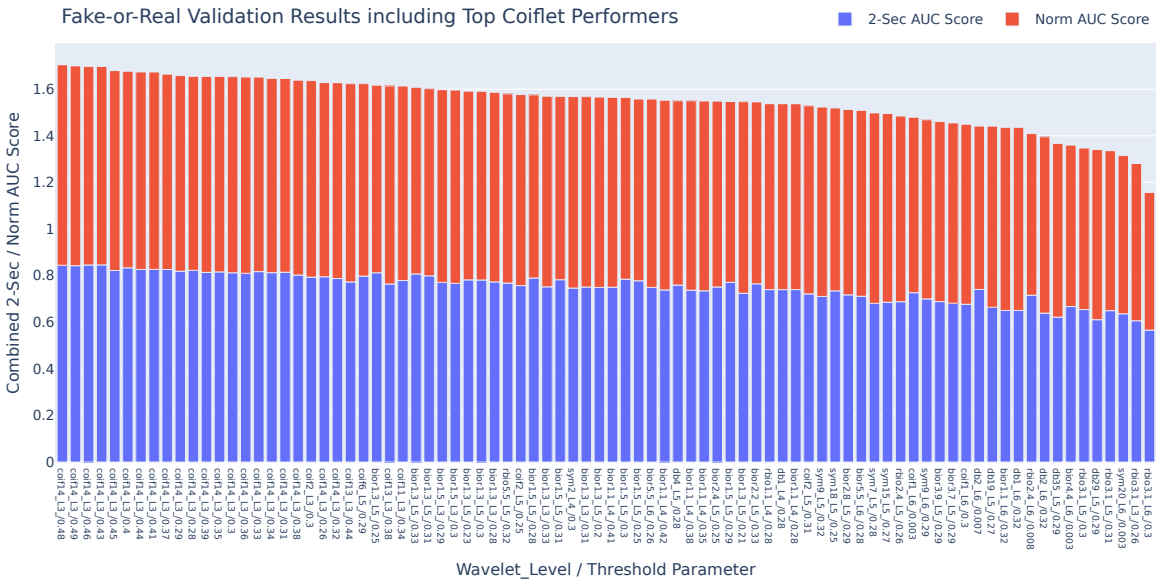

Figure 3.3. Fake-or-Real Validation using Coiflets. This figure shows that the top performing wavelet/parameters combinations are variations on the coiflet-14 wavelet.

At the conclusion of our experiments with the Fake-or-Real dataset, the wavelet/parameters combination with the best results on the validation set of audio files in the FoR test folder was the coiflet-14 wavelet at a Level 3 decomposition level with thresholding at 0.48. The AUC scores for the Normalized and 2-Sec Test Folders were 0.861 and 0.844 respectively. Additionally, 22 of the top 25 combined 2-sec/Normalized Validation AUC scores used a classifier trained on a coiflet-14 wavelet at Level 3 decomposition. Except for the biorthogonal 1.3 wavelet at Level 5 decomposition, the only wavelet/parameter combinations with AUC scores greater than 0.8 for audio files in both the 2-Sec Truncated and Normalized Test folder were coiflet wavelets at Level 3 decomposition.

In summary, we notice multiple variations on one wavelet family (albeit using different parameters) that consistently has the highest AUC score in both the 2-Sec and Normalized

Validation sets. When we pair these observations with the understanding that the validation set only uses one spoofing algorithm, we have evidence that different wavelet types may be more appropriate to different synthetic generation methods. We explore this further in the next section where the dataset consists of audio samples that can be partitioned by either its speaker label or its ten generation techniques.

3.3 ASVSpoof 2015 Dataset

In this section, we examine a dataset that supported the first bi-annual Automatic Speaker Verification and Spoofing Countermeasures Challenge (ASVSpoof) created in 2015. While the dataset in Section 3.2 focused on the speaker and generation method agnostic detection of synthetically generated human speech, the 2015 ASVSpoof dataset is specifically predicated on its synthetic speech files being spoofing attacks on identified human speakers. The ASVSpoof competition challenges researchers to develop methods for combating attempts to bypass audio-based security authentication measures. To support this challenge, the dataset allows us to separate audio files by human speakers and by the spoofing methods used to synthetically generate the audio.

Our exploration of this dataset is divided into three subsections: First, we explain how the dataset is organized with respect to train/develop/evaluate partitions for speakers and spoofing methods. Second, we develop classifiers trained on individual speakers and/or spoofing algorithms and identify the extent to which our classifier detects similar spoofing attacks on different speakers. The last part of this section tackles the actual 2015 ASVSpoof Challenge to see if a classifier trained on a subset of five spoofing algorithms and 60 speakers can generalize to detecting spoofing attacks on an entirely different subset of 46 speakers using unknown methods.

3.3.1 The Dataset

Table 3.6. Synthetic / Real Files in 2015 ASVSpoof Dataset. For every human speaker, there are many more spoofing attacks than samples of authentic human speech.

	Train	**Develop**	**Evaluate**
Fake	12625	49875	184000
Real	3750	3497	9404

The 2015 ASVSpoof dataset divides it audio files into three sets of folders: Train, Develop and Evaluate. Although the Train and Develop folders share the same algorithms for generating spoofing attacks, they do not share the same speakers. The Evaluate folder, in addition to having the same methods for spoofing audio as in the Train/Develop folders, also has audio files created using five spoofing methods that are unique to that folder. Like the dataset in Section 3.2, all audio files were stored as *.wav* files and sampled at a rate of 16000 Hz. However, as Table 3.6 shows, the distributions of synthetic and real files are heavily skewed toward synthetic files for all folders.

Although there are many more fake files than real files, Figure 3.4 shows that for every speaker there is an even number of spoofing algorithms per speaker. The ten spoofing algorithms in the 2015 ASVSpoof dataset are labelled from *S1* through *S10*.

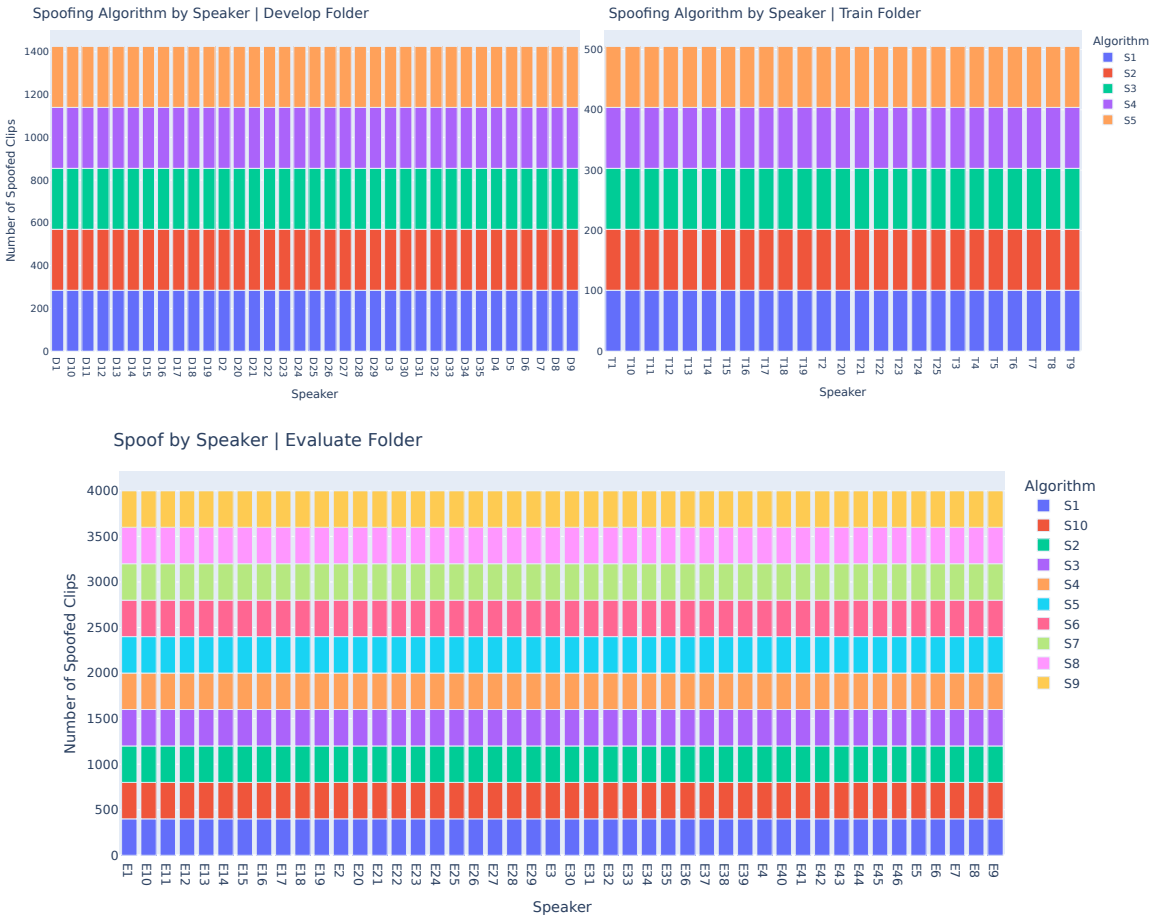

Figure 3.4. Spoofing Method Breakdown by Speaker. For the 106 different speakers (45 males and 61 females), there was an even distribution of spoofing algorithms relative to the respective Train/Develop/Evaluate folders.

As Figure 3.4 shows there are clearly more spoofing algorithms in the Evaluate folder than the Train/Develop folders. The Train and Develop folders used five spoofing algorithms that fell under the voice conversion (VC) or speech synthesis (SS) category and are labeled and described below:

- S1. A simplified frame-selection based VC algorithm.
- S2. A VC algorithm adjusting the spectrum of frequencies towards its target.
- S3. A SS algorithm using HMM-based speech synthesis and adaptation techniques using only 20 utterances.

- S4. The same SS algorithm as S3 but using 40 adaptation utterances.
- S5. A VC algorithm implemented using the publicly available Festvox Toolkit.

In addition to using spoofing algorithms, S1-S5, the evaluate folder had synthetic files generated from five other spoofing algorithms (S6-S10). A brief description of these algorithms is below:

- S6. A VC algorithm using Gaussian Mixture Models
- S7. A VC algorithm similar to S6 that uses line-spectrum pairs as opposed to mel spectrum coefficints
- S8. A tensor based VC algorithm trained on a Japanese speaking database.
- S9. A VC algorithm using a kernel-based partial least squares approach.
- S10. A SS algorithm using the open source MARY TTS system that concatenates diphone waveforms.

In their paper detailing the results of the 2015 ASVSpoof Challenge, Wu et al. (2017a) provide more background on these spoofing methods. Because the dataset allows us to organize files by either speakers {T1-T25, D1-D35, E1-46} and/or spoofing algorithms {*S1-S10*}, we have a lot of flexibility on experiments to conduct. In Section 3.3.2, we examine our classifier's ability to authenticate the speakers from which a classifier has been trained, or to generalize detection of known spoofing algorithms targeting unknown speakers.

3.3.2 Experiments with Known Spoofing Algorithms

For this set of experiments, we used a female speaker labelled in the dataset as "T1." The goal for this experiment was to identify which wavelet/parameters combination is most successful in correctly classifying files for the T1 speaker in a randomized environment consisting of other speakers and their respective spoofing attacks. These experiments train and test on synthetic files that have only been generated from the spoofing algorithms *S1-S5*. When a spoofing algorithm creates a synthetic audio file that is subsequently trained on by the classifier, that spoofing algorithm is referred to as a "known spoofing algorithm."

Since we are only training a classifier on speaker T1, however, we are limited to how many files that can be used to both train and validate the classifier. For speaker T1, we have

500 spoofed audio samples and 155 authentic recordings. If we build a classifier without correcting for this imbalance between fake and real files, the classifier may be biased toward classifying audio files as fake. To address this, we reduce the train set to 200 files that have been evenly split between fake and real. However, now we may not have enough audio files to sufficiently train the classifier. This concern is compounded by the classifier only being able to incorporate the 20 spoofed examples from 5 synthetic algorithms (100 fake files to balance 100 real files). This approach also would mean we are discarding the 400 other spoofed examples for speaker T1.

To address the concerns of either having an imbalanced train set or having too few files on which to train the classifier, we added authentic speaker recordings from the Fake-or-Real normalized folder. We justify adding real speech audio files from a different data source because we want to the classifier to maximize training on spoofed audio files. Our original theory suggested that naturally generated audio had a greater variance irrespective of individual speakers. Synthetic audio, however, was hypothesized to have more of its variation dependent on its generative algorithm. Therefore, in a fake/real balance, the classifier would more effectively incorporate the relevant information from synthetic examples that used the same spoofing algorithm than from random speakers not targeted by the spoofing attacks. With this in mind, we conducted a series of tests where files were distributed as follows:

- The train set had 800 total files with 400 spoofing attacks on the T1 speaker, 100 authentic T1 speaker files, and 300 authentic speaker files from the FoR Normalized training folder.
- The test set had 200 total files with 100 spoofing attacks on the T1 speaker, 50 authentic T1 speaker files and 50 authentic speaker files from the FoR Normalized training folder.

When running this classifier using all the wavelets and a range of thresholding parameters at Level 3 decomposition, we observed the following accuracy scores on the test set using different coiflet families within the wavelet grouping for coiflets (Group 2).

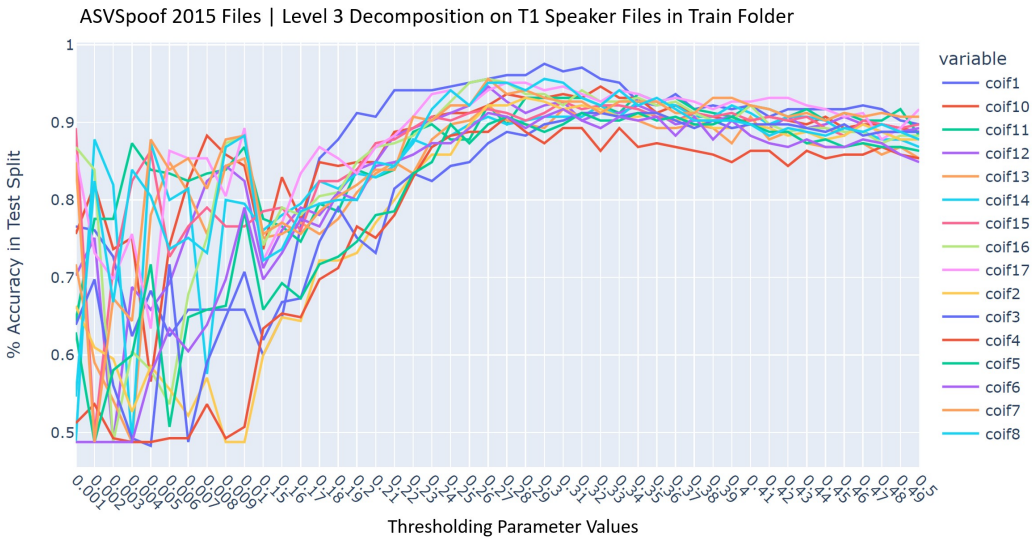

Figure 3.5. T1 Speaker Test Results. A large number of wavelet/parameter combinations for the T1-specific classifier successfully identified over 90% of the audio files in the test set.

Similar to the tests in Section 3.2.2, we tested the classifier's ability to accurately identify files that were in the same folders as the audio files used to train the classifier. However, the classifiers on the 2015 ASVSpoof train had more consistent success in identifying synthetic audio files than the mini-tests in the FoR dataset. We also noticed that in the 0.01 to 0.10 (separated by .01) band of thresholding parameters, there was much greater variance in accuracy results. These strong accuracy scores at Level 3 Decomposition were consistent across the other wavelet groupings.

For the next series of tests, we assessed the extent to which the classifier overtrains to the speaker and the extent to which the speaker-trained classifier generalizes to other speakers. To support this assessment, we increased the number of files to train the classifier from 800 to 895. We withheld 5 authentic T1-speaker files and 55 spoofing attacks on that speaker for the validation set. To test the classifier's ability to generalize to new speakers we added 200 files from the Develop Folder to the validation set–giving us a total of 260 files on which to test the classifier. While the new files use the same spoofing algorithms, the classifier has no prior information on the new files' speakers or spoofing attacks. After running the classifier on the validation set, we separated the T1 speaker files into their own subset and

considered the audio samples from the Develop Folder as a separate group. For additional analysis, we evaluated which spoofing algorithms were successfully identified as synthetic regardless of the targeted speaker.

For our first round of tests using this method, we used a wavelet/parameter combination that had one of the highest accuracy scores (0.97) during the mini-tests at Level 3 decomposition. The results of using this coiflet-1 wavelet with a thresholding parameter of 0.3 are illustrated in Figure 3.6 below:

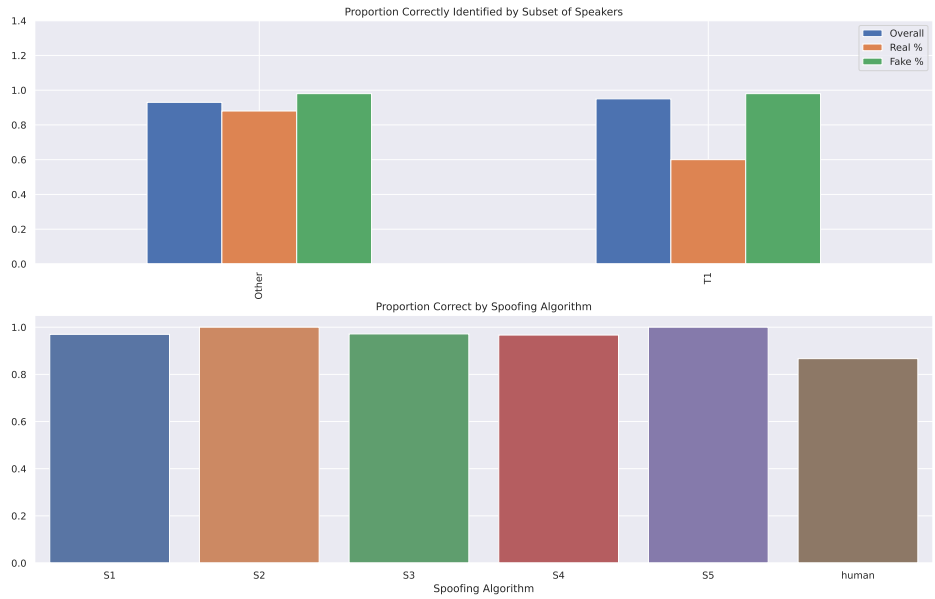

Figure 3.6. Speaker T1 Trained Classifier Results using the Coiflet-1 Wavelet. Top: Out of the 200 files from the Develop Folder, only 14 files were misclassified. For the T1 speaker, 3/60 files were misclassified. Bottom: The *S1*, *S3* and *S4* algorithms had misclassification errors. Most of the errors in the methods breakdown are due to the 14 false positives in the subset of 105 authentic human speaker files.

In the top chart, the T1 speaker and the other speakers are separated into different clusters. The barplot columns show that the percentages within each real/fake/overall category that were correctly identified. For example, the classifier incorrectly attributed one of the spoof attacks on T1 to be a real file and claimed two authentic T1 speaker audio files to be a

spoofing attack. The lower bar chart shows that the S5 spoofing algorithm generated the most synthetic speaker audio files that fooled the classifier. Additionally, the largest number of errors when looking at results by methods were false positives of spoofing attacks (i.e. 14/105 authentic human speaker files were classified as spoofed files).

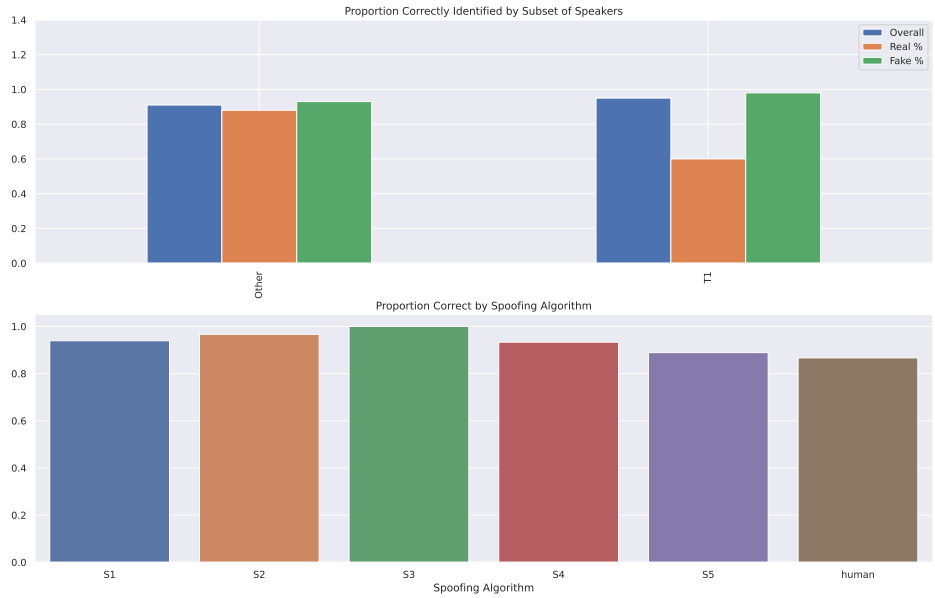

Figure 3.7. Speaker T1 Trained Classifier Results using Low Performer. Of the 200 Develop folder and 55 T1 speaker files, the classifier respectively misclassified 19 and 3 audio files. Additionally most of the spoofing methods had at least one misclassification error and the false positive rate for authentic human speaker files remained constant.

While these results showed great promise in generalizing the identification of known spoofing algorithms to other speaker (all categories in the high 80s/ low 90s), we conducted an additional test to check if other high performing wavelets in the tests improve the results for the T1 speaker. However, the other wavelet/parameter combinations at Level 3 decomposition that had similarly promising results to Figure 3.5 were unable to train a classifier that perfectly classified the T1 speaker files.

As an additional experiment, we tried a wavelet/parameter combination (biorthogonal-5.5, Level 3 decomposition, 0.15 threshhold) that performed the most worst on the test with 0.58 accuracy. In Figure 3.5, we saw a wide range of accuracies within the tests and wanted to

see if this range is reflected in our validation set. The results are displayed in Figure 3.7.

As expected, the overall results for the classifier in Figure 3.7 were lower than the results in Figure 3.6. However, the drop-off in results were less than anticipated. The test's accuracy of .58 did not extend to the validation set in the develop folder. The drop in results primarily were spoofing attacks on other speakers being labelled as authentic. While it still scored lower than the top performing wavelet combinations, the overall rate around 91-95% accuracy is still much higher than the earlier observed 58%.

The comparatively low variation in results from Figures 3.6 and 3.7 gives us low confidence that we can find an optimal wavelet/parameter combination at only Level 3 decomposition to correctly classify the speaker on which it was trained. To see if we could observe differences across different levels of parameters we tested Decomposition Levels 1-5 and reduced the range of thresholding coefficients to {0.00005, 0.0001, 0.0002, 0.001, 0.002, 0.01, 0.011, 0.015, 0.1, 0.2, 0.23, 0.3, 0.32}. Table 3.7 below shows the how the top scoring tests improved at higher levels of decomposition across the five wavelet grouping categories.

Table 3.7. T1 Speaker Mini-Tests Across Levels. For all wavelet category groupings, the accuracy for the top performing wavelet/parameter combination increased significantly at higher levels of decomposition.

	Group 1		**Group 2**		**Group 3**	
	Wavelet	**Accuracy**	**Wavelet**	**Accuracy**	**Wavelet**	**Accuracy**
Level 1	Biorthogonal 3.9	0.8478	Coiflet-14	0.8195	No Wavelet Made Cutoff	NA
Level 2	Biorthogonal 2.2	0.8878	Coiflet-15	0.8976	Daubechies-2	0.9366
Level 3	Biorthogonal 4.4	0.9561	Coiflet-11	0.961	Daubechies-31	0.9659
Level 4	Biorthogonal 5.5	0.9805	Coiflet-1	0.9805	Daubechies-2	0.9854
Level 5	Biorthogonal 3.5	0.9805	Coiflet-2	0.9902	Daubechies-2	0.9951

	Group 4		**Group 5**	
	Wavelet	**Accuracy**	**Wavelet**	**Accuracy**
Level 1	No Wavelet Made Cutoff	NA	No Wavelet Made Cutoff	NA
Level 2	Reverse-Biort 3.5	0.9122	Symlet-2	0.9366
Level 3	Reverse-Biort 2.2	0.9512	Symlet-14	0.9561
Level 4	Reverse-Biort 2.4	0.9854	Symlet-2	0.9854
Level 5	Reverse-Biort 2.8	0.9902	Symlet-2	0.9951

Although previous tests and subsequent validation tests suggested that differences in the test may not extend to the validation tests in the Develop folder, the rates of improvement at higher levels of decompression were striking. Therefore, we took the top performing wavelet combination (daubechies-2 with thresholding of 0.3 at decomposition level 5), tested the validation set from the Develop Folder, and displayed the results in Figure 3.8.

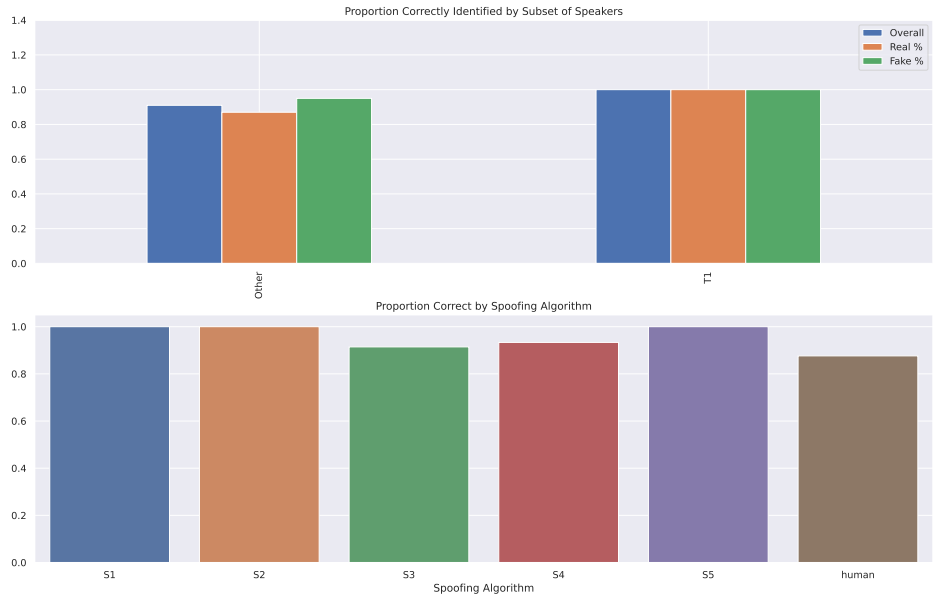

Figure 3.8. Speaker T1 Trained Classifier Results using Level 5 Top Performer. In addition to correctly classifying all the T1 Speaker audio files, it correctly classified all the VC generated spoofing files without increasing the false positive rate for authentic human speaking files.

Because of the high performance (over 90% accuracy) of the wavelet combinations at Level 3 decomposition, any improvements we observe from the Level 5 classifier will be from 1-2 more files being correctly classified. For these sets of experiments, our goal was for a speaker-trained classifier to balance authenticating the speaker files with generalizing to other speaker clips that were excluded from the training set. After using the daubechies-2 DWT at Level 5 decomposition, we noticed the classifier scored perfectly in detecting spoofed audio for the VC spoofing algorithms, *S1,S2* and *S5*. Additionally, these improvements decreased the number of false-positive errors with human audio files from 14 to 13. Lastly, the T1 speaker files in the validation set were all classified correctly. Another interesting thing to note is that while the difference is minimal, the classifier performed the most poorly on the *S3* and *S4* spoofing algorithms, the two non-VC methods.

The strong performances of the wavelet combinations at Level 5 decomposition were sufficiently encouraging for us to apply these wavelet/parameters to the partition of audio files used for validation in the actual 2015 ASVSpoof Competition. The results of our experiments in the corresponding Evaluate folder for the 2015 ASVSpoof dataset are discussed in the next subsection.

3.3.3 The 2015 ASVSpoof Challenge

For this series of experiments, we limited ourselves to the ASVSpoof 2015 Train Folder and the ASVSpoof 2015 Evaluate Folder to be consistent with the methods of previous researchers who also submitted their classifiers to the same challenge. Correspondingly, we also did not supplement the ASVSpoof 2015 Train real speaker files with the additional real speaker files to correct the the 12675:3750 fake-to-real imbalance. The primary differences between the train folder and evaluate folder are summarized in Table 3.8.

Table 3.8. ASVSpoof 2015 Train Set and Validation Set. Not only does the validation set of audio files in the Evaluate Folder have a greater imbalance between real and fake files, it also contains five spoofing algorithms that were not used to train the classifier.

	Fake Files	**Authentic Files**	**Spoofing Algorithms**	**Speakers**
Train Folder	12625	3750	5	25
Evaluate Folder	184000	9404	10	46

To assess the results of our classifier on the audio files in the Evaluate folder we used two metrics. First, we calculated the Equal Error Rate (EER) to classify the audio files in the overall dataset, the partition of audio files created by spoofing algorithms from the Train Folder (S1-S5), and the audio files generated using the unknown spoofing algorithms S6-S10. To calculate the EER we used the PYLLR package in Python (Brümmer and De Villiers 2013). The EER is similar to the ROC score because it incorporates the weighting of likelihood for either false positive or false negatives into its overall metric. In general terms, the EER calculates the empirical false alarm, $P_{fa}(\theta)$, and the miss rate, $P_{miss}(\theta)$ at threshold θ. These values are computed using the following formula.

$$P_{fa}(\theta) = \frac{\#\{\text{spoof trials with score} > \theta\}}{\#\{\text{total spoof trials}\}}$$

$$P_{miss}(\theta) = \frac{\#\{\text{genuine trials with score} \leq \theta\}}{\#\{\text{total genuine trials}\}}$$

The EER is the the threshold θ_{EER} where EER= $P_{fa}(\theta_{EER}) = P_{miss}(\theta_{EER})$ In their paper on the ASVSpoof 2015 competition, Wu et al. (2017a) provide more background on EER and their justification for using it to evaluate the competitors' submissions.

Our second assessment metric is the same as in Section 3.3.2 where we calculate the proportion of files that were classified correctly in their respective subsets. The three larger subsets are partitioned into authentic human speech files, audio files generated by known spoofing algorithms learned by the classifier (*S1-S5*) and audio files created by the spoofing algorithms excluded from the training set (*S6-S10*). The ten smaller subsets correspond to the audio files created from spoofing algorithms *S1* through *S10*.

System ID	Equal Error Rates (EERs)		
	Known attacks	Unknown attacks	Average
A	0.408	**2.013**	**1.211**
B	0.008	3.922	1.965
C	0.058	4.998	2.528
D	**0.003**	5.231	2.617
E	0.041	5.347	2.694
F	0.358	6.078	3.218
G	0.405	6.247	3.326
H	0.670	6.041	3.355
I	0.005	7.447	3.726
J	0.025	8.168	4.097
K	0.210	8.883	4.547
L	0.412	13.026	6.719
M	8.528	20.253	14.391
N	7.874	21.262	14.568
O	17.723	19.929	18.826
P	21.206	21.831	21.518
Average	3.337 (STD: 6.782)	9.294 (STD: 6.861)	6.316 (STD: 6.558)

Figure 3.9. Summary of Primary Submission Results in ASVSpoof 2015 Challenge. The blue line indicates where our Equal Error Rate (EER) score would have placed among the top 16 submissions for that year. Source: Wu et al. (2017b).

For our first set of results, we used the wavelet/parameter combination that had the highest accuracy (0.9951) in the series of mini-tests in Section 3.3.2. Although there were two wavelet/parameter combinations with that accuracy, the symlet-2 wavelet at Level 5 Decomposition with a thresholding value of 0.32 had the strongest performance when assessing the files in the Evaluate Folder. For the overall results, the EER score was 10.391. For the known algorithms and algorithms excluded from the training set, the EER scores were 3.728 and 17.260 respectively. As we show in Figure 3.9, if our classifier had been entered into the 2015 ASVSpoof Competition, our scores would have scored 13th out of the top 16 submissions for that year.

Table 3.9 shows the results of our experiments with the scoring methods that we used in Section 3.3.2.

Table 3.9. Symlet-2 Level 5 Classifier Results on Evaluate Folder. The 71% accuracy for the unknown algorithms can be traced to the low accuracy scores for both the *S8* and *S10* spoofing algorithms.

Overall	0.854
Known Algorithms	0.99
Algorithms Excluded from Train Set	0.709

S1	0.998	**S6**	0.996
S2	0.99	**S7**	0.996
S3	0.978	**S8**	0.49
S4	0.985	**S9**	0.999
S5	0.997	**S10**	0.063

In Table 3.9, we highlight the two accuracy scores in the spoofing algorithms excluded from the train set that significantly degrade the overall averages. Coincidentally, these two spoofing algorithms, *S8* and *S10*, are uniquely distinct from the other unknown algorithms because *S10* is the only synthetic speech method and *S8* was was trained on a Japanese speaking dataset. If we look at the performance of the known algorithms, we notice that the the two lowest performers were also the synthetic speech algorithms corresponding to *S3* and *S4*.

This interesting distinction in results between the synthetic speech spoofing algorithms

(*S3,S4,S10*) and the remaining VC algorithms suggests that the classifier trained on all the algorithms may be experiencing a trade off between the spoofing categories. To test this, we reduced the level of smoothing on the symlet-2 wavelet by lowering the Decomposition Level to 2. As a result, the symlet-2's DWT reconstruction more closely matched the pattern for the original audio file's signal. Figure 3.10 illustrates the impact of this parameter adjustment on the smoothed symlet-2 wavelet to refine the range of residuals extracted from each audio file.

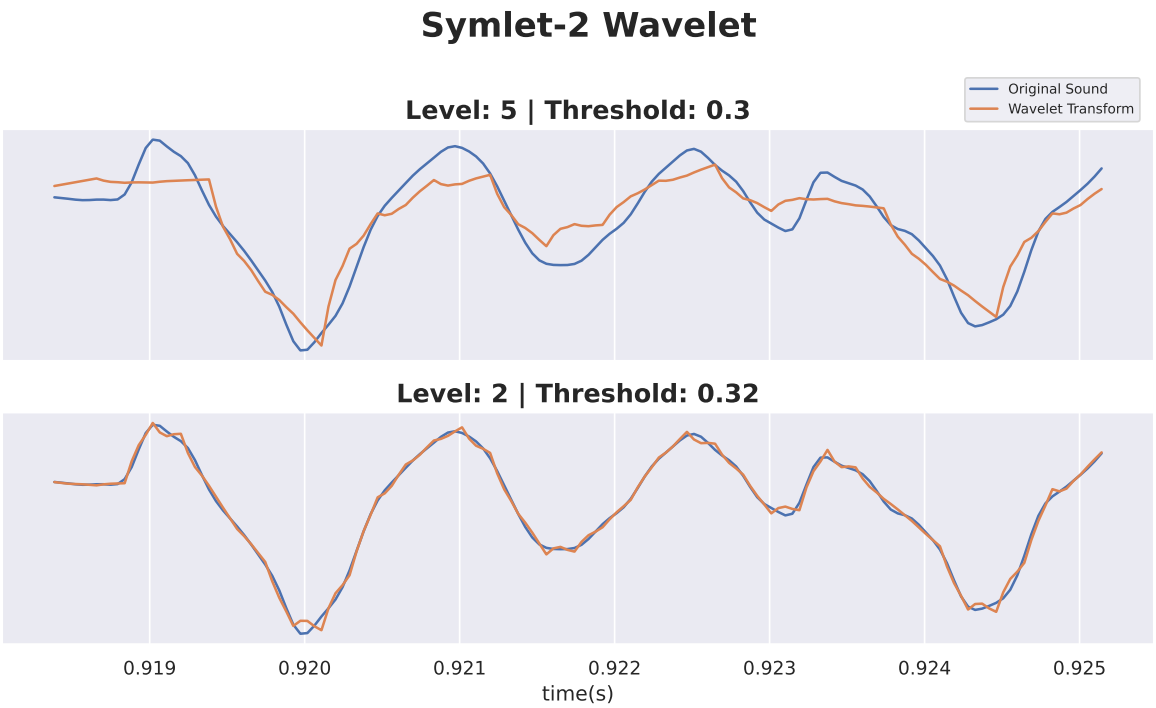

Figure 3.10. Symlet-2 Level 5 vs Level 2 Decomposition Level.

Table 3.10 shows our results on the 2015 ASVSpoof's validation set in its Evaluate folder after we applied the closer approximation of the symlet-2 wavelet at Level 2 decompression.

After testing the new classifier, we notice that the accuracy scores for *S8* and *S10*, both spoofing algorithms were excluded from the train set, improved dramatically. While *S10* still scored poorly, the accuracy of *S8* increased from 49% to 99%. While the newly calculated overall EER increased to 11.14, the overall accuracy also increased to 0.894.

Table 3.10. Symlet-2 Level 2 Classifier Results on Evaluate Folder. The red cells indicate how the VC methods decreased in accuracy among the known spoofing algorithms in conjunction with improving reults for TTS methods. These improvements extended to both known spoofing algorithms and spoofing algorithms excluded from the train set.

Overall	0.894
Known Algorithms	0.965
Algorithms Excluded from Train Set	0.878

S1	0.949	**S6**	0.938
S2	0.941	**S7**	0.967
S3	0.998	**S8**	0.993
S4	0.998	**S9**	0.984
S5	0.937	**S10**	0.245

More interestingly, we notice the consequences of the classifier with Level 3 (as opposed to Level 5) decomposition on the known methods. The synthetic speech methods (S3 and S4) increased from 9% to 99% at the expense of the VC methods (*S1,S2,* and *S5*).

These rounds of experiments provide more evidence that our selection of wavelets to build the classifier involves compromising between identifying VC spoofing algorithms and TTS algorithms. Therefore, when building a wavelet-based classifier, it may be better to train the classifier separately on different spoofing algorithms to better isolate and identify an individual spoofing method. We apply this idea in Chapter 3's final section, where we attempt to build a classifier that can confirm or deny the presence of a spoofing attack on the Ukrainian president, Volodmyr Zelensky, in the first months of the 2022 Russia-Ukraine military conflict.

3.4 Debunking a Deepfake

This section covers our use of discrete wavelet transforms (DWT) to build a classifier that analyzes the audio from a Zelensky deepfake video published to social media on March 16, 2022. In this video, during the ongoing Russia-Ukraine conflict, the president of Ukraine calls on Ukrainian defenders to surrender to Russian forces. First, we provide

a brief background on the Russia-Ukraine conflict to provide the context of the video. Then, we describe our initial analysis of the video's audio portion and how it led to our development of a classifier that directly compares audio clips reportedly belonging to a certain speaker. In the last part of this section, we summarize our classifier's results after using it to compare the Zelensky deepfake to an authentic Ukrainian president Zelensky speech uploaded to Youtube on April 22, 2022. The findings provide strong evidence for the case that audio portion of the Zelensky deepfake was generated using a voice actor and a voice conversion (VC) spoofing algorithm.

3.4.1 Background

After several weeks of Russian military buildup on the Ukraine border, Russia launched a full-scale attack into the territory of Ukraine on February 24, 2022. Despite initial media assessments projecting Kyiv to fall within 96 hours of the invasion (*Newsweek* 2022), the Ukrainian president, Volodmyr Zelensky, refused offers to evacuate, declaring famously, "The fight is here; I need ammunition, not a ride" (Network 2022). Zelensky elected to remain in Kiev and issued daily broadcasts to Ukraine's citizens to bolster their fighting spirit (*Hollywood Reporter, The* 2022). During these broadcasts, Zelensky lauded Ukrainian officials for thwarting multiple attacks on his life (*New York Post* 2022b) and also had to repeatedly debunk Russian media's efforts to fuel speculation on whether he actually fled the city despite claims to the contrary (*Observers* 2022).

Finally, on March 16, 2022, the national television station, Ukraine 24, was briefly hacked to display a fake Zelensky message on its scrolling-text news crawl and to upload the Zelensky deepfake video on to the Ukraine 24 website (*National Public Radio* 2022). Loosely translated, the video called on Ukrainian defenders to "lay down your arms and return to your families. You shouldn't die in this war. I advise you to live, and I'm going to do the same" (*New York Post* 2022a). The video also briefly circulated on popular social media platforms like Facebook, YouTube and Twitter before it was quickly removed. Despite being rapidly de-platformed, the video had been amplified on VKontakte, a social media network similar to Facebook, popular in Russia, and claimed to be controlled by allies of the Kremlin (*Atlantic Council* 2022).

While multiple articles quickly pointed out abnormalities in various head poses, facial

expressions, and odd-sounding accents, the analysis is generally subjective and centers on visual examinations of frozen frames throughout the video (*Sky News* 2022). By contrast, our method focuses only on the audio portion of the deepfake. In the next section, we show how our method of detecting synthetic audio supports an observer's ability to make visually compare the differences between a deepfake and an authentic recording.

3.4.2 Scatterplot Comparison

When initially examining the Zelensky deepfake, we wanted to create points of comparison between the Zelensky deepfake video and authentic videos of Zelensky's public addresses to Ukraine in similar settings. Therefore, we downloaded the YouTube videos of the Zelensky deepfake and three full-length Zelensky broadcasts. Two broadcasts (March 2 and March 9) were recorded prior to the deepfake and one was recorded after the deep fake (March 21). We extracted the audio portion of these videos and converted them to *.wav* files sampled at 44.1 kHz and broken up into equal 4-second segments. As a result, we had 291 audio samples from authentic Zelensky broadcasts and 17 audio samples from the Zelensky deepfake.

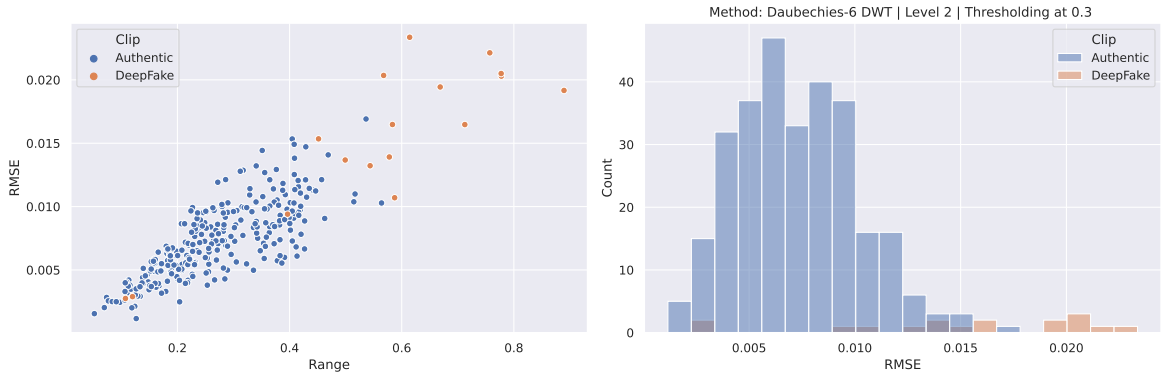

Figure 3.11. A Visual Comparison of the Zelensky deepfake to Authentic Broadcasts. Visualizing the separation in residual features between the Zelensky deepfake and authentic broadcast suggests that the voices in the recordings belong to different speakers.

To analyze these files, we applied the DWT using the daubechies-6 wavelet at Level 2 Decomposition with a soft threshold value of 0.3 to every audio file and extracted its

residuals. We used this combination because the daubechies family of wavelets appeared to have the most general success during the mini-tests in the 2015 ASVSpoof dataset. From the resultant distribution of residuals, we collected the root mean square error (RMSE) and the residual range as each audio file's individual features. For visual analysis, we label the Zelensky deepfake features in orange, the authentic file features in blue, and compared their graphical representations in both a scatter plot and a histogram. The results are displayed in Figure 3.11.

For both charts in Figure 3.11, we clearly see that the RMSE values for the deepfake samples are clustered separately from the RMSE values representing the authentic broadcasts. This pattern, however, is not consistent for all wavelet/parameter combinations. For example, Figure 3.12 reveals no discernible differences between the audio files after applying the symlet-2 DWT at Level 5 decomposition.

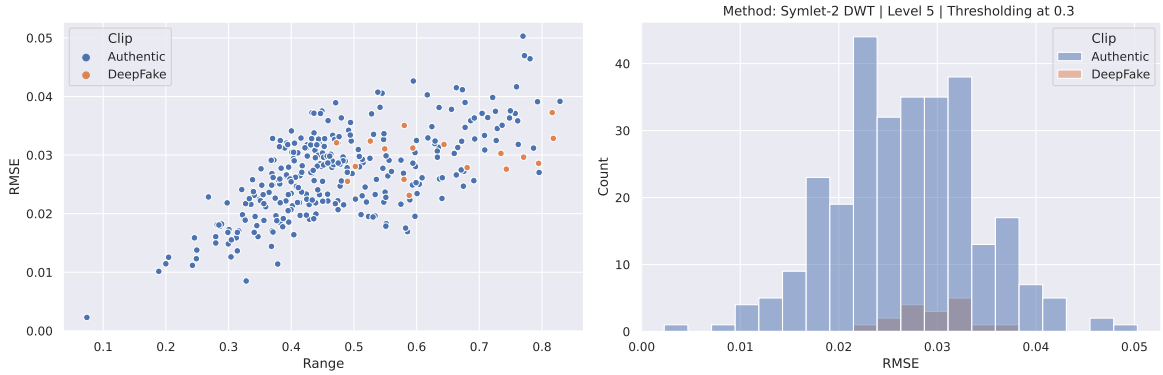

Figure 3.12. Symlet-2 DWT for Comparing Zelensky deepfake to Authentic Files. Not all wavelet/parameter combinations are effective at distinguishing between the Zelensky deepfake and Authentic Zelensky broadcasts.

These examples show that while some wavelet/parameter combinations support the differentiation of some deepfake samples from some authentic audio samples, other wavelet/parameter combinations cannot. However, this inability to generalize visual discriminations across all wavelet/parameter combinations does not challenge this paper's conjecture that we can use residuals to distinguish between fake and real audio. Rather, it supports our theory that different curve approximations capture different acoustic elements

within the audio file. Consequently, the more closely a curve approximation captures the method for creating the synthetic audio, the more the residuals help us distinguish between real and fake audio. However, there are different generative algorithms for spoofing audio, and we have yet to identify a wavelet/parameter combination that can universally apply to all spoofing algorithms.

The lack of a "one-size-fits-all" solution, however, should not be a limiting factor when building a classifier that seeks to verify the vocal authenticity of specific speakers. In the next two subsections, we show how a range of wavelets and parameters can be jointly utilized to successfully and agnostically provide significant evidence that the Zelensky deepfake was not only a synthetically manipulated audio clip, but also a product of a VC spoofing algorithm.

3.4.3 Building the Classifier

To train the classifier, we used the audio files from the 2015 ASVSpoof Dataset—only files related to spoofing attacks generated from 2015 ASVSpoof spoofing algorithms *S1-S5* in the Train and Develop folders. Correspondingly, the authentic human speaker files in this 2015 ASVSpoof subset belong only to those speakers whose vocal identities are being attacked by spoofing algorithms *S1-S5*. Additionally, rather than simultaneously training the classifier on all the spoofing algorithms, we train each of the the five classifiers individually on one of the spoofing algorithms. Lastly, we take a random sample of 257 files from the 274 four-second clips of the authentic Zelensky broadcasts. Because we are only training on one spoofing algorithm, the classifier trains on an even distribution between real and spoofed files prior to incorporating the additional 257 files from the authentic Zelensky broadcasts. Table 3.11 displays the breakdown of real and fake files on which a spoofing algorithm specific classifier is trained.

Table 3.11. Breakdown of Train Files Used in Algorithm-Based Zelensky Classifier. Although there are 36,235 spoofed files in both the Train/Develop folders, only 7,247 spoofed files correspond to one spoofing algorithm.

	Authentic	**Spoofed**	
ASVSpoof	7247	7247	
Zelensky	257		
	7504	7247	**Total**

As we see in Table 3.11, there is a minor imbalance between spoofed and authentic files on which we train the classifier. However, it is important to add these authentic Zelensky files to the train set to provide the classifier data points of comparison for assessing if a test Zelensky file is authentic or spoofed. Since the 257 additional authentic Zelensky audio samples constitute less than 5% of the authentic files, we anticipate that the fake/real imbalance will not bias the classifier towards classifying all files to be authentic.

In summary, each of the five classifiers uses 14,751 files and is trained on one of the spoofing algorithms *S1-S5*. However, not only do we have one trained classifier for every spoofing algorithm, we have 1060 wavelet/parameter combinations per every classifier: This represents 105 different wavelets, using either Level 3 or Level 5 decomposition, with five thresholding values $\{0.001, 0.01, 0.2, 0.3, 05\}$ per level. In this way, we can analyze a broad range of DWT approximations for each audio file and for each spoofing algorithm.

The next step is to filter though the 1060 wavelet/parameter combinations and incorporate only those combinations with relevant classifer results. After training on 14,768 audio samples, each classifer looks at a validation set consisting of 34 audio files. Seventeen of the audio files are the four-second clips extracted from the audio recording that claims to represent Volodmyr Zelensky. We refer to these clips as deepfake clips. The other seventeen audio files are the remaining audio samples from the 291 four-second clips extracted from authentic Zelensky broadcasts. We refer to these clips as authentic clips. For every clip in the validation set, the classifier assigns a scalar value representing the probability of that clip being fake. If the scalar value is greater than 0.5 for an authentic Zelensky clip, then that clip is a false positive because a known authentic speech file was misclassified as a

synthetic file. If the scalar value is greater than 0.5 for a deepfake file, then the clip is a true positive.

After the classifier has assigned a scalar value to each of the 34 audio clips in the validation set, we add up the number of false positives and true positives. If the difference between the false positives and true positives is at least 3, then we say that the classifier has made a comparative determination of the authenticity of both files (i.e., from which the clips derive). For example, if the classifier has 10 true positives and 7 false positives, then the classifier has determined the test files to more likely have a greater synthetic origin than the authentic set. Conversely, if there are 5 or more false positives and 2 true positives, the classifier has determined the authentic files to have more synthetic origin than the test set. However, if there were 14 false positives and 16 true positives, the classifier would not make a determination on either set even though both sets have a clear majority of files being individually classified as fake.

Using this criterion of comparative determination, we only count the classifier instances where the classifier clearly indicates that one set of audio files has been classified as distinctly different from the other set. This distinction allows us to filter through the wavelet/parameter combinations and incorporate only those combinations with relevant classifier results (i.e., results indicating one set of files is considered more synthetic than the other) into an overall score for the classifiers trained on one algorithm. For the purposes of this paper, we refer to this final scoring across the relevant wavelet/parameter combinations distinguishing between the authentic and test clips as the *overall classifier*. Table 3.12 provides an example of how an overall classifier incorporates 10 hypothetical mini-classifier tests representing 10 unique wavelet/parameter combinations.

Using this method for testing all 1060 wavelet/parameter combinations per spoofing algorithm, we aggregate relevant results across a range of audio file approximations. In this case, we measure relevancy by how strongly a classifier has determined a set of files to be more synthetic than the other set (i.e., the difference between true positives and false positives is at least 3). After completing all 1060 classifier tests, we aggregate the values of true positives and false positives from the relevant test results and create an overall true positive to false positive (TP/FP) ratio. This overall TP/FP ratio provides us insight into what extent all the classifiers, each trained on a specific spoofing algorithm using a unique wavelet/parameter

Table 3.12. Combining Results into the Overall Classifier. The green rows represent the wavelet/parameter combination providing enough of a true positive to false positive differential to be aggregated into the total ratio for the Overall Classifier.

Wavelet	Level	Threshold	True Positive	False Positives	Difference	Relevant	Score (TP,FP)
Symlet-2	3	0.3	15	14	1	No	NA
Symlet-2	3	0.01	0	1	1	No	NA
Symlet-2	5	0.3	3	1	2	No	NA
Haar	5	0.5	7	0	7	Yes	(7,0)
Coiflet-2	5	0.3	4	1	3	Yes	(4,1)
Coiflet-3	3	0.001	4	2	2	No	NA
Coiflet-3	3	0.3	0	3	3	Yes	(0,3)
Daubechies-3	5	0.2	10	2	8	Yes	(10,2)
Daubechies-3	5	0.001	5	1	4	Yes	(5,1)
Biorthogonal	3	0.01	2	4	2	No	NA
						Total Ratio	26 to 6

combination, had determined a set of suspected deepfake files to either be more fake or less fake. These suspected deepfake files are contrasted with a set of audio files that have been pre-identified as authentic samples of speech belonging to a specific speaker. The greater a TP/FP value than 1 that overall classifier has, the more confidence the classifier has that the deepfake set is composed of spoofed audio files. In contrast, the closer the TP/FP value is to zero, the more the classifier believes the deepfake set to be less fake than the set of files that have previously been identified as authentic.

In this chapter's final subsection, we apply this overall classifier approach to compare the Zelensky deepfake audio to audio from an authentic Zelensky broadcast that was recorded on April 21, 2022.

3.4.4 Testing the Classifier on the Zelensky Deepfake

As the previous section describes, we built five overall classifiers that were individually trained on five distinct spoofing algorithms. Each classifier is labelled *S1* through *S5* with each label corresponding to the classifier's use of one of the 2015 ASVSpoof spoofing algorithms. For our test sets, we used the 17 clips that were created from the Zelensky deepfake and 17 sequential four-second segments from an authentic Zelensky broadcast that was uploaded to YouTube on April 21, 2022. For the test set representing the authentic Zelensky recording, we ensured that the 17 sequential clips corresponded to a section of the broadcast that did not contain significant pauses in speech. In this way, the test set with authentic Zelensky speech samples had similar structure to the test set comprising the Zelensky deepfake samples.

For each classifier, and for each test set, we calculated the logarithm of the overall TP/FP ratio. Our reasoning for taking the logarithm of the TP/FP ratio was to visually approximate the extent to which the classifier believes the test set of files to be less fake than known authentic files. If we visually compared a 78:1 ratio to a 0.02 ratio on a barchart, it would have diminished the relevant information of the 1:50 (i.e., 0.02 ratio) imbalance indicating the strength for which a test set of files was determined to be more authentic than known real files. Because of the monotonic property in logarithmic functions, if we apply a logarithmic transform to a sequence of numbers (in this case the different TP/FP ratios for the overall classifiers), we preserve the given order of numbers in that sequence. Additionally, the logarithmic transform enables us to visualize more uniformly the differences between ratios with values greater than one and ratios with values between zero and one. Quickly identifying lopsided ratios is especially important when comparing TP/FP scores for overall classifiers. These comparisons between overall classifiers can be easily seen in Figure 3.13's results of the five algorithm specific classifiers on the test sets representing the authentic April 21, 2022, Zelensky clips and the March 16, 2022, Zelensky deepfake clips.

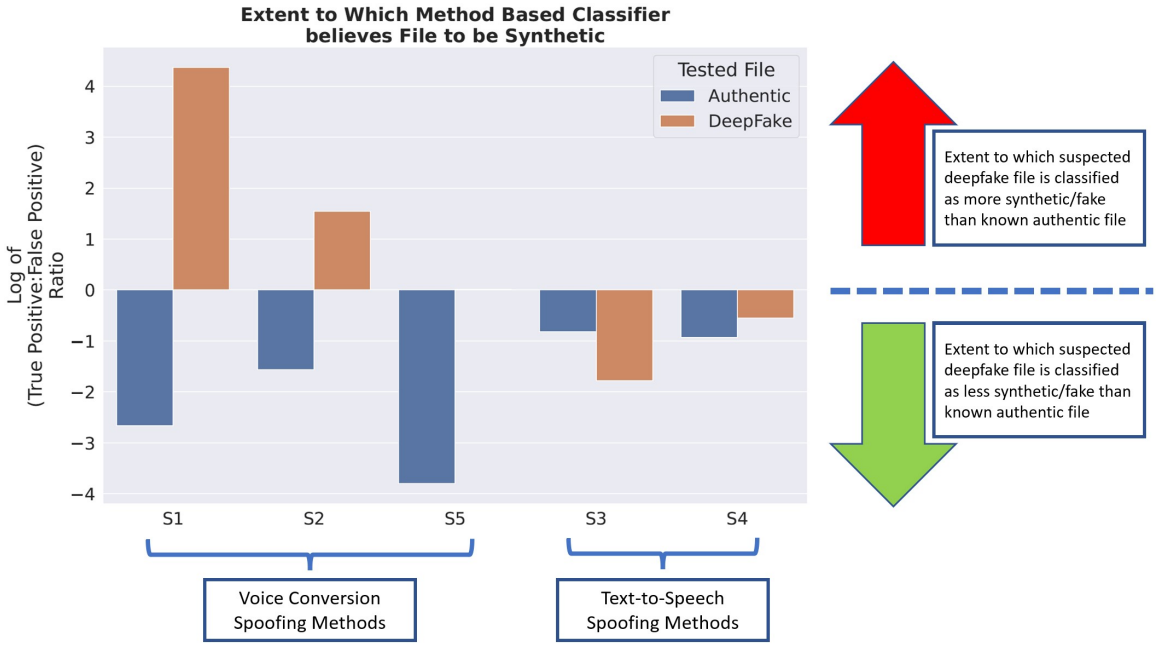

Figure 3.13. Classifier Comparisons of Zelensky deepfake to Authentic Broadcast. Across the five algorithm-trained classifiers, the negative values for the blue columns affirm the classifiers' ability to correctly confirm the authentic clips' authenticity. In contrast, the positive orange deepfake columns clearly suggests that VC-based classifiers provide strong indicators for the clips in the deepfake test set to be classified as fake.

Figure 3.13 shows significant differences in how the spoofing-based classifiers assess audio clips originating from the Zelensky deepfake and the April 21, 2022, authentic broadcasts. For the text-to-speech (TTS) spoofing algorithms (*S3* and *S4*), there appears to be little difference in the degree to which the classifiers assess both sets of files to be authentic. However, the VC-based classifiers (*S1,S2 and S5*) show a clear separation in results between the test sets. All three VC-based classifiers strongly support that the the test set of authentic files is less fake than the set of files known to real. In contrast, the S1-based classifier clearly assesses the test set of deepfake files to be much more fake than the authentic clips in classifier.

Table 3.13's articulation of the actual TP/FP ratios for each algorithm-based classifier reinforces the strength with which the S1-based classifier argues that the deepfake test set ought to be classified as synthetic.

Table 3.13. Numerical Breakdown of Zelensky Classifier Results by Spoofing Algorithm. The green cells highlight the spoofing algorithm where the classifer noticed the strongest imbalance between authentic and test clips in the validation set. The red cells indicate the algorithm-based classifiers where at least 90% of the mini-tests did not find a relevant difference between the clips in the validation set.

		Deepfake (March 16, 2022)		Authentic (April 21, 2022)	
	Algorithm	**TP/FP Ratio**	**% No Diff**	**TP/FP Ratio**	**% No Diff**
Voice Conversion	S1	78.76	59.1	0.07	91.4
	S2	4.7	83.4	0.21	94.9
	S5	1.01	90	0.02	85.1
Text- to-Speech	S3	0.17	72.2	0.44	82
	S4	0.57	82.9	0.39	79.4

Table 3.13 reveals two important details not readily apparent in Figure 3.13. First, it shows that the test set with the Zelensky deepfake using the S1-based overall classifier had over a 78:1 TP/FP ratio which numerically dwarfs the other TP/FP observations. For example, Figure 3.13 visually suggests that the S2-based overall classifer for the deepfake test set is comparable to the S1-based overall classifier, albeit only at a quarter strength. However, the two scores are heavily imbalanced with the *S2* TP/FP score of 4.7 being over 15 times smaller than the *S1* TP/FP score of 78.8.

The second important detail in Table 3.13 is the "% No Diff" column, which indicates the percentage of tests in the spoofing algorithm-based classifier whose TP/FP scores were not included in the overall TP/FP ratio. Only classifiers where the difference between true positives and false positives greater than two were included in this overall ratio. Understanding the proportion of wavelet/parameter combinations contributing to the overall TP/FP score adds important context to the overall tests. For example, a strongly lopsided TP/FP ratio using a small proportion of classifier results suggests that certain wavelet/parameter combinations were really effective in training a classifier to distinguish between sets of test files. Conversely, a high proportion of wavelet/parameter contributions to the overall TP/FP ratio indicates that the spoofing algorithm-based classifier is successfully generalizing across multiple wavelets.

This value provides additional context to the overall TP/FP ratios, both for the S1-based

classifier's scoring of 78:1 for the deepfake and the S5-based classifier's scoring of 1:50 for the authentic 21APR22 Broadcast. Table 3.13's "% No Diff" value of the 59.1 for the S1-based classifier's assessment of the deepfake tells us that over 400 wavelet/parameter combinations identified a significant difference between the test deepfake clips and the authentic clips. However, approximately only 160 wavelet/parameter tests for the S5-based classifier contributed to the assessment that the authentic broadcast from April 21, 2022 was less synthetic than the authentic clips in the validation set.

With these details in mind, we return to the results of the classifiers that were built to assess the Zelensky deepfake. Of all the algorithm-based classifiers, the S1-based classifier's assessment of the deepfake had the most lopsided TP/FP ratio. Additionally, the S1-based classifier's overall TP/FP ratio incorporated the most results from the 1060 wavelet/parameter combination-based mini-tests. Both factors suggest that the audio portion of the Zelensky deepfake used a spoofing algorithm with the most similarities to the S1-spoofing algorithm.

As explained in Section 3.3.1, the 2015 ASVSpoof *S1* spoofing algorithm uses the simplest form of Voice Conversion compared to spoofing algorithms *S2* and *S5*. Additionally, the *S1* spoofing algorithm produces sound files that are audibly discordant to the human ear. Nevertheless, while the Zelensky deepfake has audible characteristics that are clearly more natural sounding than *S1* generated files, the S1-trained classifiers provide the strongest evidence that the Zelensky deepfake is not authentic human speech. This suggests that the S1-based classifier is identifying inconsistencies in the residual structure that are not audibly distinguishable. In fact, the coarseness of the *S1* algorithm's approach toward spoofing audio may be working to the advantage of the S1-based classifier. This is because we believe the S1-based classifier is weighting more heavily the generalized indicators of VC methods when differentiating between synthetic and authentic audio samples. Additionally, since the overall classifier runs the gamut of filtering through the residual analyses of over 1000 different wavelet smoothing combinations, different wavelets may be triggering different acoustic inconsistencies within the test files for the classifier. Therefore, while we cannot point to which VC spoofing algorithm generated the Zelensky deepfake audio, we can confidently say the audio in the deepfake most likely originated from a recording of voice actor whose acoustic features were synthetically manipulated to sound like the Ukrainian president, Volodmyr Zelensky.

CHAPTER 4:
Conclusions

In this thesis, we introduced the theory that the natural variation in synthetically generated human speech is less than that of digital audio originating from the recording of a human speaker. More specifically, we hypothesized that analyzing the variation in a digital audio file enables us to effectively discriminate between synthetic (fake) audio files of a human speaker and audio files representing authentic (real) human speech. To analyze this variation in audio, we used discrete wavelet transforms (DWT) to reconstruct close approximations of that audio and extract its residuals to form a distribution. Summary statistics from these distributions were then analyzed by the gradient-boosted tree model (GBM) classifier to make the determination of whether the analyzed audio is fake or real. In this chapter, we describe our conclusions after applying this method to two distinct datasets and discuss additional opportunities related to this thesis for further research.

Findings

The first dataset, Fake-or-Real (FoR), trained the classifier using fake speech samples individually generated from multiple deep learning text-to-speech (TTS) algorithms. The fake audio files in the Fake-or-Real (FoR) validation set, however, were all created using the TTS algorithm, Google's *Cloud TTS with Wavenet*, that was not used to create audio files in the train set. While we were unable to acquire a AUC score greater than 0.86 using different wavelet/parameter combinations, the classifier's top scores converged on the use of the coiflet-14 wavelet at level 3 decomposition and using threshold values greater than 0.25. Therefore, we conclude that of the 105 wavelets used for the DWT, using the coiflet-14 wavelet to construct the smoothing from which residuals are derived provides the best classifier for identifying audio files generated in 2018 using the Google's *Cloud TTS with Wavenet*.

The second dataset from the 2015 Automatic Speaker Verification and Spoofing Countermeasures Challenge (ASVSpoof) Competition focused on spoofing attacks targeted at specific speakers and used a wider variety of spoofing algorithms to create fake audio files. Experimentation on this dataset revealed that classifiers using a single wavelet/parameter

combination for analyzing audio files face a trade-off in accuracy between two types of spoofing algorithms, voice conversion (VC) and text-to-speech (TTS). In other words, certain wavelet/parameter combinations in the classifier will score more accurately on audio files generated by VC algorithms than on audio files generated by TTS algorithms. Conversely, if we adjust the level of decomposition to favor more accurate scoring of the TTS algorithms, the classifier's ability to effectively identify fake files generated by VC algorithms decreases.

Both datasets suggest that analyzing the natural variation in digital audio using residuals supports the successful discrimination between synthetic and real audio. Residuals' effectiveness in supporting that discrimination, however, depends on the algorithms used to approximate the original signal. In this paper, we tested multiple reconstructions using DWT derived from various combinations of wavelet and thresholding parameters. Our experiments showed that different wavelet/parameter combinations led to different levels of success in identifying synthetic files depending on the spoofing algorithm. We were unable to find a perfect wavelet/parameter combination with universal success in identifying synthetic files created by spoofing algorithms that had not been used to train the classifier.

These findings, however, did not discourage us from building a classifier that can assess suspected spoofed audio files originating from unknown spoofing algorithms. Rather than relying on one classifier using a specific wavelet/parameter combination, we built an overall classifier that consolidates the results of multiple "mini" classifiers. These mini-classifiers were distinguished by their use of a specific variation of the wavelet/parameter combinations and by their being trained on only one spoofing algorithm. In this case, we trained five overall classifiers using the 2015 ASVSpoof spoofing algorithms—*S1-S5*. When we tested these classifiers to compare the March 16, 2022 Zelensky deepfake to the April 21, 2022 authentic Zelensky broadcast, we produced strong evidence that the audio portion of the Zelensky deepfake was the product of a voice conversion (VC) method.

In summary, our thesis demonstrated the promising potential of incorporating the analysis of digital audio residuals into determining the natural authenticity of human speech files. We acknowledge that the classifier is limited in that it can only use one wavelet/parameter combination to create the residuals that are analyzed to support the successful identification of signal audio. However, we also constructed a technique for filtering through multiple

wavelet/parameter combinations and consolidating the scores for relevant classifiers. The relevancy of the classifier depended on the extent to which it made a clear distinction between audio clips extracted from two sets of recordings—a recording authentically attributed to a certain speaker, and a recording suspected of being a spoofing attack on that speaker. Overall scoring from the consolidated classifiers in this technique was effectively used to compare the extent to which a suspected spoofing attack has more synthetic qualities than an audio recording of an authentic human speaker.

Further Topics for Research

The work in this thesis can be expanded upon in a variety of ways and at multiple levels of specificity. From a macroscopic perspective, researchers may identify additional methods for approximating a digital audio's signal and extracting residuals that provide more information about its synthetic qualities. Additionally, there may be merit in isolating which wavelet types generalize best to which generative synthetic speech algorithms as we briefly showed when analyzing the FoR dataset. If researchers want to continue exploring the use of wavelets and their reconstructions of the DWT, our uniform application of soft thresholding a decomposition level's detail coefficient can be conditionally refined as appropriate.

When analyzing the residuals, there may be alternatives to using gradient-boosted tree model (GBM) classifiers. Although the gradient-boosted tree model (GBM), by itself, is an ensemble method for classification, it does not limit its use in conjunction with other classification models. Also, we only use the summary statistics from the distribution of residuals for training the classifier. The GBM framework can seamlessly incorporate additional features associated with each audio sample—either additional feature observations from the residual or entirely different features extracted from another audio deconstruction method.

While presently using only summary statistics and GBM, our constructed classifier for assessing the Zelensky deepfake offers multiple opportunities for useful refinements and additional insights. For example, the individual classifiers from which the overall classifier derives its scores uses a training set where over 98% of the files have no connection with the speaker we are trying to classify. Of the 14,751 audio clips in the train set, 257 audio clips are non-English speakers and the classifier is assessing a Ukrainian speaker. In other

words, the overall classifier exhibited promising results when generalizing from an English-speaking dataset to the Ukrainian language. It is possible our use of the 257 authentic clips of Zelensky may not be necessary to effectively compare the Zelensky deepfake to the authentic broadcast. If we don't need those 257 clips for the classifier to discriminate the Zelensky deepfake as fake, then our classifier could potentially be applied to other spoofing vignettes in other non-English languages without the pre-requisite training.

Finally, the Zelensky classifier is only trained on five relatively rudimentary spoofing algorithms. Researchers may find value in training the overall classifier on newer methods for synthetically generating audio. Not only might different spoofing methods generalize to different categories, but it also provides another approach for narrowing which wavelet/parameter combinations are worth further exploration when distinguishing between speakers and methods. This filter for additional wavelet analysis can be applied by isolating which wavelet/parameters had the most lopsided true positive to false positive (TP/FP) ratios.

List of References

Agarwal S, Farid H, Gu Y, He M, Nagano K, Li H (2019) Protecting World Leaders Against Deep fakes. *CVPR Workshops*, volume 1.

Anumanchipalli GK, Prahallad K, Black AW (2011) Festvox: Tools for Creation and Analyses of Large Speech Corpora. *Workshop on Very Large Scale Phonetics Research, UPenn, Philadelphia*, 70.

Appiah-Dolphyne J (2020) Audio of Government Official caught for Money Laundering is Fake-ncsc (OCT 15), accessed April 10, 2022, https://asaaseradio.com/audio-of-government-official-caught-for-money-laundering-fake-ncsc/.

Atlantic Council (2022) Russian War Report: Hacked News Program and Deepfake Video Spread False Zelenskyy Claims (March 16), accessed May 20, 2022, https://www.atlanticcouncil.org/blogs/new-atlanticist/russian-war-report-hacked-news-program-and-deepfake-video-spread-false-zelenskyy-claims/.

Borrelli C, Bestagini P, Antonacci F, Sarti A, Tubaro S (2021) Synthetic Speech Detection through Short-term and Long-term Prediction Traces. *EURASIP Journal on Information Security* 2021(1):1–14.

Breiman L (2001) Random Forests. *Machine Learning* 45(1):5–32.

Brownlee J (2019) Xgboost with Python. *Machine Learning Mastery*.

Brümmer N, De Villiers E (2013) The Bosaris Toolkit: Theory, Algorithms and Code for Surviving the New DCF. *arXiv preprint arXiv:1304.2865*.

Cambre J, Kulkarni C (2019) One Voice Fits All? Social Implications and Research Challenges of Designing Voices for Smart Devices. *Proceedings of the ACM on human-computer interaction* 3(CSCW):1–19.

Charpentier F, Stella M (1986) Diphone Synthesis using an Overlap-add Technique for Speech Waveforms Concatenation. *ICASSP'86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 11, 2015–2018 (IEEE).

Chen T, Guestrin C (2016) Xgboost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794.

Chesney B, Citron D (2019a) Deep Fakes: A Looming Challenge for Privacy, Democracy, and National Security. *Calif. L. Rev.* 107:1753.

Chesney R, Citron D (2019b) Deepfakes and the New Disinformation War: The Coming Age of Post-Truth Geopolitics. *Foreign Aff.* 98:147.

Cohen A, Kovacevic J (1996) Wavelets: The Mathematical Background. *Proceedings of the IEEE* 84(4):514–522.

De Leon PL, Pucher M, Yamagishi J, Hernaez I, Saratxaga I (2012) Evaluation of Speaker Verification Security and Detection of HMM-based Synthetic Speech. *IEEE Transactions on Audio, Speech, and Language Processing* 20(8):2280–2290.

Dutoit T, Holzapfel A, Jottrand M, Moinet A, Perez J, Stylianou Y (2007) Towards a Voice Conversion System based on Frame Selection. *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 4, IV–513 (IEEE).

Fraser M, King S (2007) The Blizzard Challenge 2007. *Proc. BLZ3-2007 (in Proc. SSW6).*

Freund Y, Schapire RE (1997) A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting. *Journal of Computer and System Sciences* 55(1):119–139.

Fukada T, Tokuda K, Kobayashi T, Imai S (1992) An Adaptive Algorithm for Mel-Cepstral Analysis of Speech. *ICASSP*, volume 92, 137–140 (Citeseer).

Gantenbein D (2020) Advances in Text-to-Speech Technologies Help Computers Find Their Voice (APR 20), accessed April 18, 2022, https://www.amazon.science/latest-news/advances-in-text-to-speech-technologies-help-computers-find-their-voice.

Google (2022) Speech Processing. Accessed April 18, 2022, https://research.google/research-areas/speech-processing/.

Guardian,The (2022) Ukraine Reveals "Russian Warship, go fuck yourself!" Postage Stamp (MAR 12), https://www.theguardian.com/world/2022/mar/12/ukraine-reveals-russian-warship-go-fuck-yourself-postage-stamp.

Hellemans A (2021) How Wavelets Allow Researchers to Transform, and Understand, Data (Oct 13), accessed April 28, 2022, https://www.quantamagazine.org/how-wavelets-allow-researchers-to-transform-and-understand-data-20211013/.

Hollywood Reporter, The (2022) The Role of His Life: Comedian-Turned-President Volodymyr Zelensky Rises to the Moment (February 28), accessed May 20, 2022, https://www.hollywoodreporter.com/news/politics-news/how-ukraine-president-volodymyr-zelensky-transformed-comedian-to-wartime-leader-1235100944/.

Ito K, Johnson L (2017) The LJ Speech Dataset. https://keithito.com/LJ-Speech-Dataset/.

Lee G, Gommers R, Waselewski F, Wohlfahrt K, O'Leary A (2019) Pywavelets: A Python Package for Wavelet Analysis. *Journal of Open Source Software* 4(36):1237.

Lu L, Ghoshal A, Renals S (2011) Regularized Subspace Gaussian Mixture Models for Speech Recognition. *IEEE Signal Processing Letters* 18(7):419–422.

McFee B, Raffel C, Liang D, Ellis DP, McVicar M, Battenberg E, Nieto O (2015) librosa: Audio and Music Signal Analysis in Python. *Proceedings of the 14th Python in Science Conference*, volume 8, 18–25 (Citeseer).

Narendra N, Rao KS (2015) Robust Voicing Detection and F-null Estimation for HMM-Based Speech Synthesis. *Circuits, Systems, and Signal Processing* 34(8):2597–2619.

National Public Radio (2022) Deepfake Video of Zelenskyy could be 'Tip of the Iceberg' in Info War, Experts Warn (MAR 16), https://www.npr.org/2022/03/16/1087062648/deepfake-video-zelenskyy-experts-war-manipulation-ukraine-russia.

Network CN (2022) Zelensky Refuses US Offer to Evacuate, Saying 'I Need Ammunition, Not a Ride' (February 26), accessed May 20, 2022, https://www.cnn.com/2022/02/26/europe/ukraine-zelensky-evacuation-intl/index.html.

Newsweek (2022) Exclusive: U.S. Expects Kyiv to Fall in Days as Ukraine Source Warns of Encirclement (February 24), accessed May 20, 2022, https://www.newsweek.com/us-expects-kyiv-fall-days-ukraine-source-warns-encirclement-1682326.

New York Post (2022a) Deepfake Video of Zelensky telling Ukrainians to Surrender Removed from Social Platforms (March 17), Accessed May 20, 2022, https://nypost.com/2022/03/17/deepfake-video-shows-volodymyr-zelensky-telling-ukrainians-to-surrender/.

New York Post (2022b) 'I'm not afraid of anyone': Zelensky Defiantly Shares Location amid Russian Attacks (March 7), Accessed May 20, 2022, https://nypost.com/2022/03/07/ukrainian-president-volodymyr-zelensky-revealed-his-location/.

Observers (2022) A Pre-Recorded Video? the Pro-Russian Hoax Suggesting that Volodymyr Zelensky Has Left Ukraine (March 15), Accessed May 20, 2022, https://observers.france24.com/en/europe/20220315-a-pre-recorded-video-the-pro-russian-hoax-suggesting-that-volodymyr-zelensky-has-left-ukraine/.

Opitz D, Maclin R (1999) Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research* 11:169–198.

Panda SP, Nayak AK (2017) Vowel Onset Point Based Waveform Concatenation Technique for Intelligible Speech Synthesis. *2017 International Conference on Computing Methodologies and Communication (ICCMC)*, 622–626 (IEEE).

Paul D, Pal M, Saha G (2017) Spectral Features for Synthetic Speech Detection. *IEEE Journal of Selected Topics in Signal Processing* 11(4):605–617.

Ping W, Peng K, Gibiansky A, Arik SO, Kannan A, Narang S, Raiman J, Miller J (2017) Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning. *arXiv preprint arXiv:1710.07654* .

Poddar A, Sahidullah M, Saha G (2018) Speaker Verification with Short Utterances: A Review of Challenges, Trends and Opportunities. *IET Biometrics* 7(2):91–101.

Povey D, Burget L, Agarwal M, Akyazi P, Kai F, Ghoshal A, Glembek O, Goel N, Karafiát M, Rastrow A, et al. (2011) The Subspace Gaussian Mixture Model—A Structured Model for Speech Recognition. *Computer Speech & Language* 25(2):404–439.

Premium Times (2020) False; President Akufo-Addo did not Caution Ghanaians against a covid-19 Plan by Rockefeller Foundation (JUL 3), https://www.premiumtimesng.com/news/fact-checks/400867-false-president-akufo-addo-did-not-caution-ghanaians-against-a-covid-19-plan-by-rockefeller-foundation.html.

Reimao R, Tzerpos V (2019) For: A Dataset for Synthetic Speech Detection. *2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD)*, 1–10 (IEEE).

Saito D, Minematsu N, Hirose K (2012) Tensor-based Speaker Space Construction for Arbitrary Speaker Conversion. *2012 IEEE 11th International Conference on Signal Processing*, volume 1, 595–598 (IEEE).

Schiff KJ, Schiff D, Bueno NS (2021) The Liar's Dividend: How Deepfakes and Fake News Affect Politician Support and Trust in Media (October 21), accessed April 10, 2022, https://osf.io/qpxr8/.

Schremmer C, Haenselmann T, Bömers F (2000) Wavelets in Real-Time Digital Audio Processing: A Software for Understanding Wavelets in Applied Computer Science. *Workshop on Signal Processing Applications (WoSPA)*.

Shalizi C (2019) The Truth About Linear Regression. Accessed Apri1 15, 2022, https://www.stat.cmu.edu/ cshalizi/TALR/TALR.pdf.

Singh AK, Singh P (2021) Detection of AI-Synthesized Speech using Cepstral Bispectral Statistics. *2021 IEEE 4th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, 412–417 (IEEE).

Singh P, Singh P, Sharma RK (2011) Jpeg Image Compression based on Biorthogonal, Coiflets and Daubechies Wavelet Families. *International Journal of Computer Applications* 13(1):1–7.

Sky News (2022) Ukraine War: Deepfake Video of Zelenskyy telling Ukrainians to 'Lay Down Arms' Debunked (March 17), accessed May 20, 2022, https://news.sky.com/story/ukraine-war-deepfake-video-of-zelenskyy-telling-ukrainians-to-lay-down-arms-debunked-12567789.

Stylianou Y, Cappé O, Moulines E (1998) Continuous Probabilistic Transform for Voice Conversion. *IEEE Transactions on speech and audio processing* 6(2):131–142.

Talebi S (2020) The Wavelet Transform. *Towards Data Science*.

Taspinar A (2018) A Guide for Using the Wavelet Transform in Machine Learning (Dec 21), accessed May 03, 2022, https://ataspinar.com/2018/12/21/a-guide-for-using-the-wavelet-transform-in-machine-learning/.

Taylor P (2009) *Text-to-Speech Synthesis*. Text-to-speech Synthesis (Cambridge University Press), ISBN 9780521899277, URL https://books.google.com/books?id=BFnkm-FpBAUC.

Torlay L, Perrone-Bertolotti M, Thomas E, Baciu M (2017) Machine Learning–XGBoost Analysis of Language Networks to Classify Patients with Epilepsy. *Brain informatics* 4(3):159–169.

Umesh S, Cohen L, Nelson D (1999) Fitting the Mel Scale. *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258)*, volume 1, 217–220 (IEEE).

Valens C (1999) A Really Friendly Guide to Wavelets. *ed. Clemens Valens*.

VoxForge (2019) Free Speech ... Recognition (Linux, Windows and Mac). voxforge.org.

Wang Y, Skerry-Ryan R, Stanton D, Wu Y, Weiss RJ, Jaitly N, Yang Z, Xiao Y, Chen Z, Bengio S, et al. (2017) Tacotron: Towards End-to-End Speech Synthesis. *arXiv preprint arXiv:1703.10135*.

Wu Z, Yamagishi J, Kinnunen T, Hanilçi C, Sahidullah M, Sizov A, Evans N, Todisco M, Delgado H (2017a) Asvspoof: The Automatic Speaker Verification Spoofing and Countermeasures Challenge. *IEEE Journal of Selected Topics in Signal Processing* 11(4):588–604.

Wu Z, Yamagishi J, Kinnunen T, Hanilçi C, Sahidullah M, Sizov A, Evans N, Todisco M, Delgado H (2017b) Asvspoof: The Automatic Speaker Verification Spoofing and Countermeasures Challenge. *IEEE Journal of Selected Topics in Signal Processing* 11(4):588–604.

Yamagishi J, Nose T, Zen H, Ling ZH, Toda T, Tokuda K, King S, Renals S (2009) Robust Speaker-adaptive hmm-based Text-to-Speech Synthesis. *IEEE Transactions on Audio, Speech, and Language Processing* 17(6):1208–1230.

Yoshimura T, Tokuda K, Masuko T, Kobayashi T, Kitamura T (1999) Simultaneous Modeling of Spectrum, Pitch and Duration in HMM-based Speech Synthesis. *Sixth European Conference on Speech Communication and Technology*.

Yu H, Tan ZH, Ma Z, Martin R, Guo J (2017) Spoofing Detection in Automatic Speaker Verification Systems using DNN Classifiers and Dynamic Acoustic Features. *IEEE Transactions on Neural Networks and Learning Systems* 29(10):4633–4644.

Zhang C, Yu C, Hansen JH (2017) An Investigation of Deep-Learning Frameworks for Speaker Verification Antispoofing. *IEEE Journal of Selected Topics in Signal Processing* 11(4):684–694.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California