Deepfake Audio Detection

by

Tran Tuan Dat, Le Hoang Viet, Ngo Dinh Tuan Cuong

ITS Faculty *FPT University HCMC*

ACKNOWLEDGMENTS

We would like to express our deepest appreciation to all those who provided us the possibility to complete this report:

Le Thanh Hai, Ph.D.:

- Our supervisor, whose guidance and expertise throughout this project.
- His insightful feedback significantly improved the quality of our work.

Thanks to Viettel AI and Vbee AI, as well as the appearance of the VIVOS dataset and FPT student (Thinh Le Gia Phuoc) dataset, we were able to accomplish this project.

AUTHOR CONTRIBUTIONS

The contributions of the authors to this work are outlined as follows:

Conceptualization:

- Tran Tuan Dat: Identified the potential of deep learning for Fake Voice Detection and proposed the research project.
- Le Hoang Viet, Ngo Dinh Tuan Cuong: Contributed to refining the project scope and research objectives, focusing voice analysis, and voice detection.

Methodology:

- Tran Tuan Dat: Led the research on deep learning architectures for Fake Voice Detection, particularly exploring the use of Rawnet2.
- Ngo Dinh Tuan Cuong: Modified the Rawnet2 model.
- Le Hoang Viet: Crawl Vietnamese real/fake voice data.

Writing - Review and Editing:

- Tran Tuan Dat: Contributed to sections on data preprocessing, experiments and results of training.
- Ngo Dinh Tuan Cuong: Drafted the initial manuscript, mainly responsible for the project's report.
- Le Hoang Viet: Contributed to sections on dataset and feature extraction, revised the manuscript for clarity and ensured accuracy.

All Authors have read and agreed to the Final Capstone Project document.

ABSTRACT

The goal of this thesis is to create a reliable automatic speech recognition (ASR) system that detects phony Vietnamese voices utilizing advanced neural network models. Our dataset contains over 49000 Vietnamese utterances which include bona fide and spoofed voices. Our dataset is a combination of collected data from various sources which is crawling speeches from podcast videos on Youtube and public VIVOS dataset as real data; synthesizing fake data by commercial tools like Viettel Text to Speech, Vbee AIVoice and by FPT University Student's capstone ViXTTS demo regarding to cloning voice. The methodology includes extensive data preprocessing, such as directory generation, audio cleaning, feature extraction, and dataset separation. The Rawnet2 model, which extracts high-level features, and the SincNet filter, which detects relevant features, are key components. Training and evaluation make use of advanced loss functions and adaptive learning rate adjustments.The results demonstrate the model's high accuracy and reliability, achieving an impressive accuracy of nearly 96%, a low Equal Error Rate (EER) of 0.03, and a strong F1-score of 96%. These metrics underscore the model's potential as an effective tool for real-world applications in speech verification and security.

Keywords: Fake voice generated, fake voice detection.

CONTENT

1. INTRODUCTION

The advent of deep learning has significantly advanced many fields, with automated speech recognition (ASR) being a prominent beneficiary. As ASR systems grow increasingly accurate, they are now capable of processing and interpreting human speech with high precision. However, these advancements come with new challenges, particularly due to the rise of synthetic voice generation technologies, such as deepfake audio. These technologies pose substantial risks, especially in the context of verifying the authenticity of speech. The ability to produce highly convincing synthetic voices has created an urgent need for reliable systems that can differentiate between real and synthetic speech, especially in linguistically complex languages like Vietnamese.

This thesis aims to address this need by employing cutting-edge neural network architectures designed specifically for voice analysis. Rawnet2 and SincNet are at the forefront of these technologies, offering unique methods for processing raw audio data directly, bypassing the need for extensive feature extraction that often hinders traditional ASR systems. Rawnet2 utilizes a convolutional neural network (CNN) framework to operate on raw audio waveforms, capturing detailed patterns that might otherwise be overlooked. Conversely, SincNet introduces parameterized sin functions as convolutional filters, allowing for more interpretable and efficient feature extraction from raw audio.

To develop an effective Vietnamese fake voice detection system, this research will prioritize several key areas. A critical first step involves constructing a robust and diverse dataset. This dataset will encompass a wide range of genuine and synthetic Vietnamese voices, reflecting different accents, speech patterns, and recording environments. Such diversity is crucial for ensuring that the model generalizes well to various real-world scenarios.

Effective preprocessing techniques will also be implemented to enhance the audio data quality. Techniques such as noise reduction, normalization, and data augmentation will be used to expand the dataset's size and variability artificially. Proper preprocessing allows the model to focus on the most relevant audio features, thereby improving detection accuracy.

Regarding training methodologies, this thesis will explore both supervised and unsupervised learning approaches to optimize model performance. Supervised learning will involve training the model on labeled datasets where the difference between real and synthetic voices is clearly identified. In contrast, unsupervised learning techniques, such as autoencoders or generative adversarial networks (GANs), will be utilized to help the model learn underlying patterns in the data without explicit labels, potentially identifying subtle differences between real and synthetic audio that might not be immediately apparent.

Additionally, transfer learning is expected to play a crucial role in this research. By pretraining the model on large, general-purpose datasets of human speech and subsequently fine-tuning it on Vietnamese-specific data, the model can leverage existing knowledge while adapting to the unique challenges posed by the Vietnamese language.

The goal of this thesis is to develop a highly accurate and reliable Vietnamese fake voice detection system, applicable in various domains such as fraud prevention in telecommunication services and enhancing the security of voice-activated systems. By integrating neural network architectures like Rawnet2 and SincNet with robust data and sophisticated training techniques, this research aims to contribute significantly to the ongoing efforts to counter synthetic voice fraud, ensuring that ASR systems remain trustworthy and secure in an era of rapidly advancing threats.

2. RELATED WORK

The detection of synthetic or fake voices has become increasingly crucial as voice synthesis technologies have advanced, creating highly realistic and convincing audio outputs. Among the many methods developed to address this challenge, traditional techniques such as Gaussian Mixture Models (GMMs) and modern deep learning approaches like Rawnet2 have emerged as leading solutions. GMMs have been instrumental in speech processing for many years, particularly in tasks like speaker verification. Their strength lies in their ability to model the statistical properties of voice features, making them effective in distinguishing between real and synthetic speech.

In contrast, the rise of deep learning has led to the development of more advanced techniques for detecting synthetic voices, with models like Rawnet2 at the forefront. Rawnet2, introduced by Tak et al [1], represents a significant shift from traditional feature-based approaches by processing raw audio waveforms directly. This end-to-end learning approach allows the model to automatically learn feature representations, bypassing the need for manually crafted features that often limit the effectiveness of traditional methods. The architecture of Rawnet2 utilizes convolutional neural networks (CNNs) to capture the intricate details present in audio data, while incorporating residual connections and attention mechanisms to enhance the model's robustness and accuracy in detecting synthetic voices. These sophisticated components enable Rawnet2 to effectively handle the complexities of modern synthetic audio, making it one of the most effective tools for voice synthesis detection.

The choice of datasets for training and evaluating voice synthesis detection models is critical to their performance and ability to generalize. The ASVspoof dataset, as described by Todisco et al [2], is widely regarded as a comprehensive benchmark for anti-spoofing research. This dataset includes a wide range of spoofing attacks, making it an ideal resource for developing models that can withstand various types of synthetic voice manipulations. Similarly, the Aptly Lab [3] dataset provides a diverse collection of voice samples, which further supports the model's ability to generalize across different scenarios.

To address the specific challenges of detecting fake voices in the Vietnamese language, this research incorporates a custom Vietnamese voice dataset. This dataset, carefully crawled and

curated, plays a crucial role in the training process, ensuring that the models can effectively handle cross-linguistic variations and nuances. The importance of diverse and linguistically varied datasets is emphasized by cross-linguistic studies. In addition to these established datasets, the research explores the integration of other data sources to further enhance the training process. For instance, incorporating synthesized Vietnamese voices generated by different “Text to Speech” systems provides a broader range of synthetic voice characteristics, which can improve the model's ability to distinguish between real and fake voices. This approach is in line with the findings of Wang et al [4], who highlight the importance of diverse synthetic data in training robust detection models.

Moreover, the research employs transfer learning techniques to optimize model performance. By pretraining models on large, multilingual datasets and then fine-tuning them on Vietnamese-specific data, the research takes advantage of the strengths of both broad and narrow data sources.

The research also considers the application of domain adaptation techniques to improve the model's adaptability to different contexts. By adjusting the model parameters based on the specific characteristics of the target domain - such as background noise levels or recording conditions - the model's robustness can be further enhanced.

In summary, this research aims to develop a highly accurate and reliable system for detecting synthetic voices in Vietnamese, building on advanced neural network architectures, diverse and linguistically rich datasets, and training methodologies. Through the integration of these elements, the research seeks to make significant contributions to the field of synthetic voice detection, providing a robust tool for safeguarding against the growing threat of voice-based spoofing and fraud.

3. PROJECT MANAGEMENT PLAN

Task name	Priority	Owner	Date	Status
Find documents	High	All Members	06/05/24 - 11/05/24	Done
Review papers and Test	High	All Members	13/05/24 - 26/05/24	Done
Prepare data	High	Hoang Viet	03/06/24 - 30/06/24	Done
Training	High	Tuan Dat	03/06/24 - 14/07/24	Done
Experiments	High	Tuan Cuong	03/06/24 - 23/06/24	Done
Compare results	Medium	All Members	16/07/24 - 23/07/24	Done
Future work	Medium	All Members	25/07/24 - 28/07/24	Done
Write paper & Review	High	All Members	25/07/24 - 15/09/24	Done

4. MATERIALS AND METHODS

4.1. Dataset:

4.1.1. Dataset Description:

In this part, we give an overview of our data. It contains over 49000 Vietnamese spoken audio clips (WAV files), totaling nearly 58 hours of genuine recorded and produced audio files. The project's dataset was compiled from a number of sources in order to construct robust automatic speech recognition (ASR) systems, with a concentration on Vietnamese language datasets and synthesized voice data.

4.1.1.1. Real data

Our authentic speech audio is clearly based on the VIVOS Dataset [5], and it was collected by extracting utterances from podcast recordings on YouTube.

First, the VIVOS Dataset is a free Vietnamese voice corpus developed by AILAB, a computer science lab at VNUHCM - University of Science, for the Automatic voice Recognition challenge. It consists of 12420 speech snippets captured in a quiet area with a high-quality microphone, with speakers reading one line at a time, totaling around 15 hours of recording. The dataset is diverse, with 46 voices and a large number of unique syllables based on the accents of several regions of Vietnam. The speech content has been separated into sections for training and testing.

	Train	**Test**
Speakers	46	19
Utterances	11660	760
Durations	14 hours 55 minutes	45 minutes
Unique Syllables	4617	1692

Table 1. Summary of VIVOS Dataset

Second, we include samples created by cutting podcasts videos from YouTube into multiple short clips based on the speaker's punctuation by pausing throughout their speech. In our baseline data collection from YouTube, we apply the end-to-end approach Whisper model [6], which is a multitasking model that can perform multilingual speech recognition, speech translation, and language identification, in our baseline collecting data from Youtube.

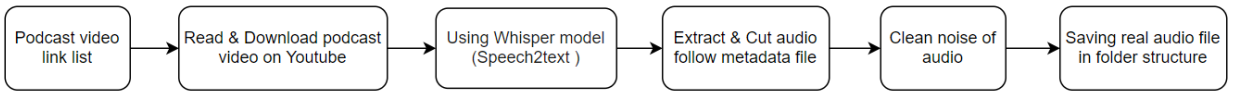


Figure 1: Pipeline of crawling real voices data from podcast on Youtube

We perform the pipeline of crawling real speech data from podcast videos on Youtube by the flowchart above. In the first step, we manually save the podcast link into the list, which is single speaker speech and as little noise as possible. The reason why we choose podcast videos to crawl the real speech data is the high quality of recording, that is less environmental noises and has prime microphones. After that, videos in the list were automatically downloaded. Afterwards, Whisper is an automatic speech recognition (ASR) system trained on 680,000 hours of multilingual and multitask supervised data collected from the web, which is open-source models and inference code to serve as a foundation for building useful applications and for further research on robust speech processing. By make use of multilingual speech transcription Whisper inference, thirdly, we take advantage of Speech-to-Text task in extracting script of each phrase in the video into metadata with following regular expression

```
8261  y_LSMqSArMc_22.wav|với Mao chạch đông, chống quân xâm lược để bảo vệ đất nước.|142.68-145.52
8262  y_LSMqSArMc_23.wav|năm 1945 thì Nhật Bản hoàn toàn thất bại trước quân đồng minh và đầu hàng vô điều kiện.|0.0-3.98
8263  y_LSMqSArMc_24.wav|Quốc dân Đảng và Đảng Cộng sản lại quay sang đối đầu nhau. Nhưng lần này, số phận đã không|3.98-9.44
8264  y_LSMqSArMc_25.wav|chọn lựa Tưởng Giới Thạch. Quân đội Trung Hoa Dân Quốc liên tục gặp thất bại và diện tích lãnh|9.44-14.32
8265  y_LSMqSArMc_26.wav|thổ bị thu hẹp giật. Vào tháng 9 năm 1949 thì quân đội của Mao Trạch Đông đã vào được thủ đô Bắc|14.32-19.86
8266  y_LSMqSArMc_27.wav|Kinh, chính thức kết thúc nội chiến quốc cộng. Tưởng Giới Thạch đã thất bại trên toàn bộ Đại|19.86-24.22
8267  y_LSMqSArMc_28.wav|Bắc Kinh. Tưởng đem toàn bộ quân đội còn sót lại, cùng với gia quyến và cả những báu vật trong|25.22-30.24
8268  y_LSMqSArMc_29.wav|tử Cấm Thành đem ra đảo Đài Loan. Hơn 2 triệu lính Trung Hoa Dân Quốc cùng gia đình cũng di chuyển theo.|30.24-35.34
```

Figure 2: Transcription after applied Whisper model

The phrase is splitted, that is based on the pausing in the speaker's flow of speech. Subsequently, because of efficient structure in metadata, we cut the video following the duration time in metadata to get the short videos, which is corresponding to the speaker's punctuations.

4.1.1.2. Fake data

To accomplish collecting Vietnamese spoofed human voice, the selection of high-performance generative tools is essential. We included a range of commercial tools in our dataset:

- **Viettel Text To Speech:** We include Viettel Text To Speech, which is one of the best commercial AI tools to automatically convert Vietnamese text into artificial voice with high naturalness and emotion, successfully applied in Audio Newspaper, Media, Content marketing on social media. In addition, it has 10 separate speakers, which is diverse in accent for three main regions in Vietnam. Moreover, it provides free advanced API for users to utilize in building Vietnamese Text To Speech applications.
- **Vbee AIVoice:** The most popular commercial tool in Vietnamese Text To Speech and AI Voice Dubbing. It provides 16 different speakers, which has full AI Voices by gender and the accent of three main regions in Vietnam. Beside that, it provides inspiring AI Voice with emotion like a human, utilized in creating voice for automatic announcements, greetings for switchboards, Youtube video, Audio book, etc.

Beside that, we generate spoofed data from a good performance open-source model Vietnamese Text To Speech ViXTTS [7], which is fine-tuned from the XTTS-v2.0.3 model of CoquiTTS

[8]. This model is fine-tuned by students at FPT University HCMC, as a component of graduation thesis. This model lets us clone voices into different 18 languages which has the best generated quality in Vietnamese, by using just a quick 6-second audio clip.

First of all, we generate spoofed speech with the first tool, that cycle synthesizes audio files by requesting the API of Viettel Text To Speech.

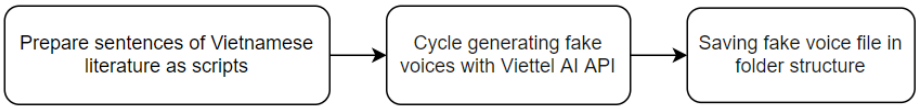


Figure 3: Pipeline of generating spoofed voices data from Viettel TTS API

To prepare scripts for generating, we split Vietnamese literature into sentences for generating the short clips. The literature has suitable and correct punctuation for the voice synthesizer to perform effective flow of natural speech is the reason why we use literacy essays for scripts. As flowchart shown above, scripts prepared are used to automatically generate speech with Vietnamese 10 voices. Finally, fake audio files are saved in a well-organized folder structure for easy access and management.

Second, generating by Vbee AIVoice is quite complicated because we do not request Vbee API to save on expenses. The baseline of synthesizing spoofed speech with this tool is inherited from a part of the pipeline of Crawling real voices from podcasts on Youtube.

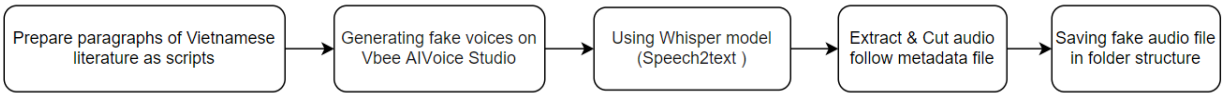


Figure 4: Pipeline of generating spoofed voices data from Vbee AIVoice Studio

To begin generating, as mentioned above, literature is corresponding to preparing scripts. We make the most of it, however, we just prepare paragraphs to manually create 16 long duration clips on Vbee AIVoice Online Studio for 16 speeches. Then, speech transcription Whisper model [6] is made use of in extracting format metadata with time of phrases in clips to cut clips into short duration audio files, which are appropriate to punctuation of speakers and distribution duration of dataset. After all, we save all spoofed speech audio files with the similar folder structure above.

Lastly, the ViXTTS is better in performing intonation in the flow of spoofed voices. Fake data generated from ViXTTS model demo with 9 high efficient sample voices, that utilized the advanced modules for the convenient crawling baseline.

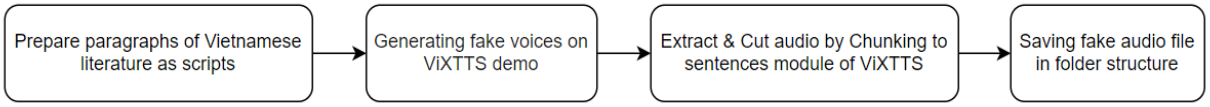


Figure 5 : Pipeline of generating spoofed voices data from viXTTS model

Similar to the first step in generating fake voices by Vbee AI tool, we input the paragraphs of Vietnamese literature in the ViXTTS demo for each sample voice. The output which is a long

duration audio will be chunked into sentences by the chunking module of demo. Finally, we store generated data in a folder as a label.

4.1.2. Dataset Preprocessing:

Initially, the directory structure is created to divide the audio files into various folders for training and testing. The audio data is subsequently loaded, and each file is validated to confirm its functionality. Files that are deemed to be invalid - due to excessive quiet, noise, or other errors - are moved to an error folder to keep them from infecting the training process.

For valid files, a resampling step is applied to ensure that all audio is at a consistent sampling rate of 16kHz, which is crucial for uniformity across the dataset. This resampling can be mathematically represented as:

$$W_{resampled}(t) = W(\frac{t}{rate}),$$

where $W(t)$ is the original waveform, and rate is the ratio of the target sampling rate (16kHz) to the original sampling rate.

The audio recordings are then chunked into four-second parts to ensure that the model receives uniform input lengths. This phase is critical for models that need fixed-size input, such as many deep learning architectures.

After chunking, the audio data is normalized with MaxAbs scaling. The normalization technique scales the waveform's maximum absolute amplitude to 1. This ensures that all audio inputs fall within a consistent range, which may be mathematically stated as:

$$W_{normalized}(t) = \frac{W(t)}{max(|W(t)|)},$$

where $W(t)$ is the original waveform, and $max(|W(t)|)$ is the maximum absolute value of the waveform.

To further enhance the robustness of the model, Gaussian noise is added to 50% of the audio files. This step increases the complexity of the dataset by simulating real-world conditions where background noise may be present. The noisy waveform can be represented as:

$$W_{noisy}(t) = W(t) + N(t),$$

where $N(t)$ is Gaussian noise with a mean of 0 and variance sigma. This variance is controlled by the standard deviation, allowing for adjustable levels of noise intensity.

Finally, the preprocessed audio data is split into training and testing sets in an 80:20 ratio, ensuring that the model has sufficient data for both learning and evaluation. This step concludes the preprocessing pipeline, resulting in a dataset that is well-suited for feature extraction and

subsequent model training. The comprehensive preprocessing ensures that the dataset is not only clean and consistent but also reflective of the diverse conditions under which the model will be deployed, thereby improving the generalizability and performance of the machine learning model.

Dataset	Train	Val	Test
Real	15,459	3,865	4,831
Fake	15,958	3,990	4,987

Table 2. Train, Valid and Test set

4.1.3. Data Insight:

For this heading we provide an analysis of our dataset. The first one is distribution of audio duration, the majority of the audio files exhibit a consistent duration.

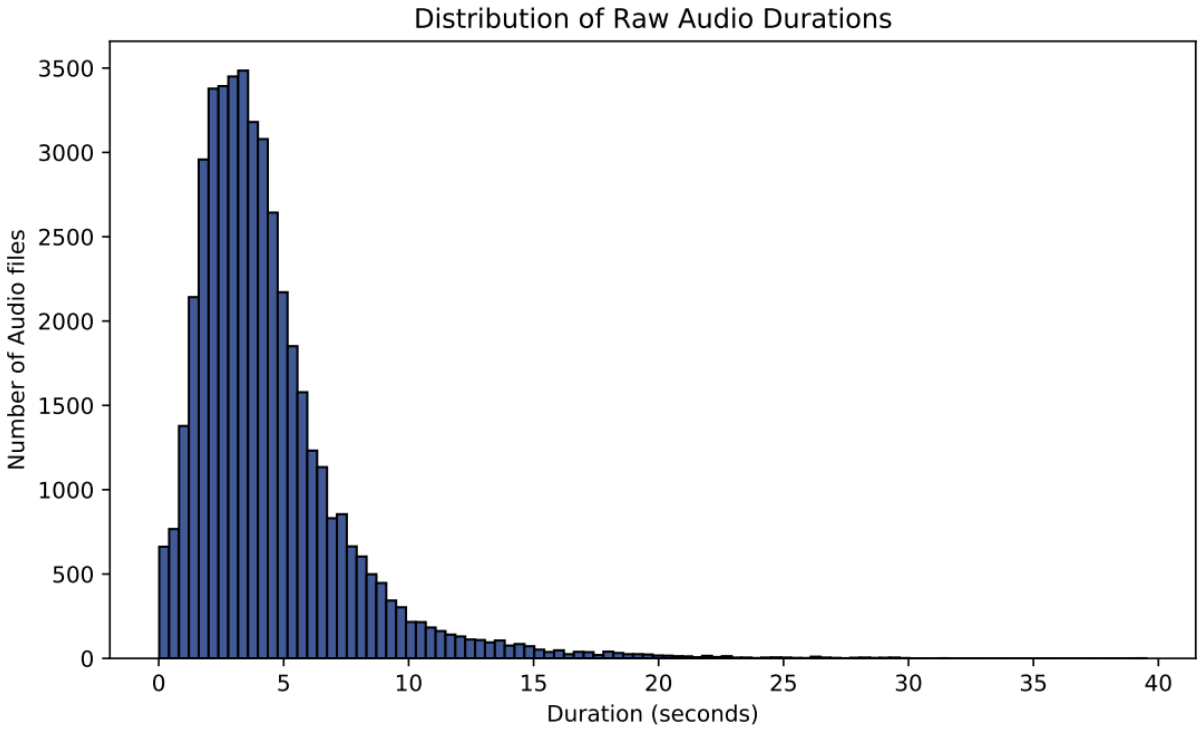


Figure 6: Distribution of Raw Data Duration

As evident from above plots, the most of the durations are bunched up from 2 to 5 seconds. The average duration of voice durations is approximately 3.8 seconds, which is suitable to average the majority of sentences spoken by humans. Beside that, a little data is scattered with over 15 seconds duration because of complex, long sentences recording in reference real dataset and in preparing scripts for generating fake data. From 0 to 1 seconds, a large number but not the majority data is observed as a result of invalid, silent audio or short sentences in communication such as Vietnamese greeting sentences, short exclamation sentences, short questions, etc. To resolve imbalance in duration of data from varied sources, we remove the silent or error audio

and the very short duration audio. Furthermore, we cut the over 5 seconds duration speech file into small files which are under 5 seconds duration. The audio file with 5 seconds duration is optimal for an average duration of normal Vietnamese speeches and for processing to extract the audio features.

Secondly, to prove the diversity of our dataset, the accent of speech is a critical and individual feature of Vietnamese dialects from different provinces in Vietnam. As mentioned above, in bona fide data, which combines the VIVOS dataset and data crawling from podcasts. The VIVOS dataset is recorded by 46 speakers with enormous different accents and dialects of Vietnamese diverse enough, which are represented by a number of unique syllables in section 4.1.1.1. The data crawled from podcast video on Youtube is the varied recording of podcasters, which is more natural and various voices in performance of Vietnamese accents and dialects. In spoofing voices, we synthesized data by the best commercial tools for Vietnamese Text To Speech which provide emotional and natural speeches and varied speakers from three main regions in Vietnam.

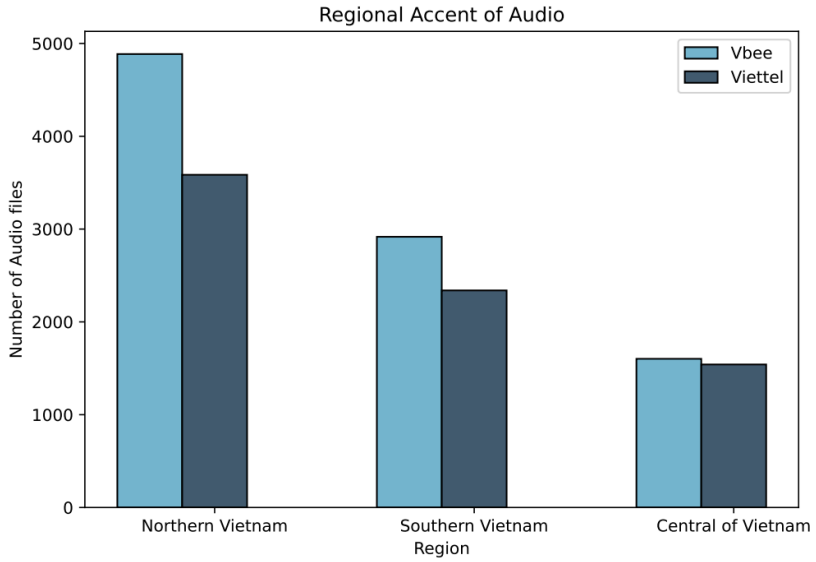
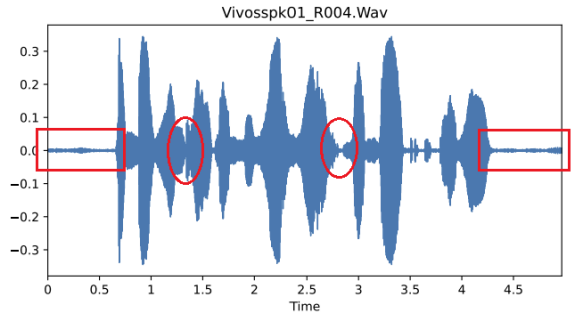


Figure 7: Regional Accent of Vietnamese generated Speeches data from Viettel TTS and Vbee AIVoice

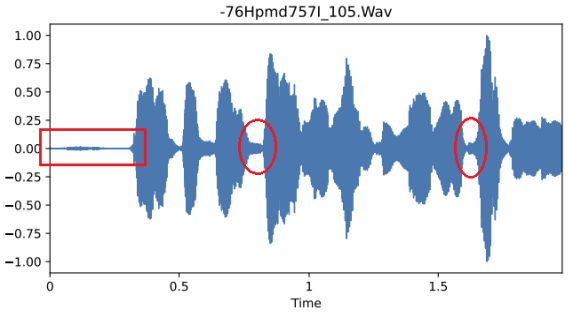
It is clear that the most voices in fake data are the accents in Northern Vietnam. It is also true that Northern Vietnamese is almost a standard Vietnamese dialect. Furthermore, the spoofed data also consists of a huge number of accents and dialects in Southern and Central Vietnam. Also, the strength of fake data cloned from ViXTTS is the intonation which is generated with 9 styles of verbalization performed by 9 different speakers.

On the other hand, to visually compare the between real and fake data, we visualize the audio signal features through images and statistics.

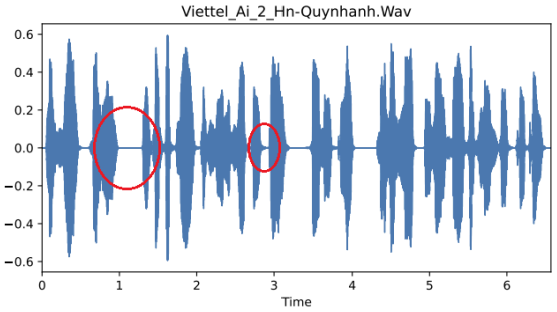
The first comparison is waveform visualization, which is graphical representation of sound waves, describing a depiction of the pattern of sound pressure variation (or amplitude) in the time domain. By that advance, we can clearly analyze the sound wave.



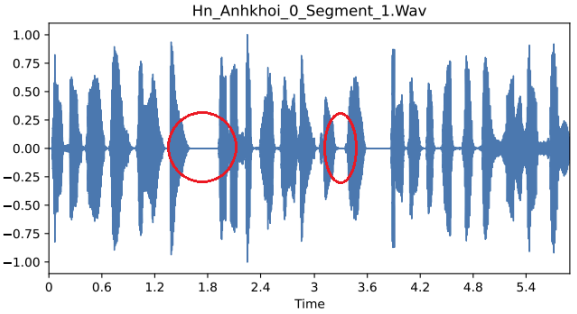
a. Real data collected VIVOS dataset



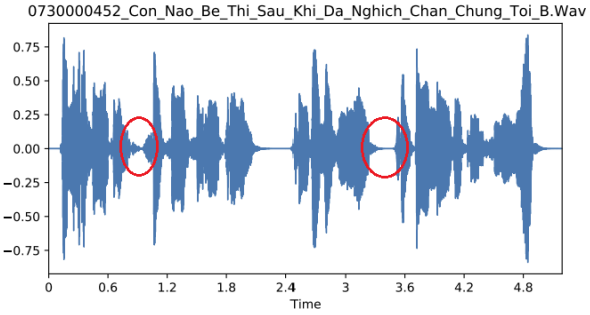
b. Real data crawled podcast (podcast data)



c. Fake data generated by Viettel TTS



d. Fake data generated by Vbee AIVoice



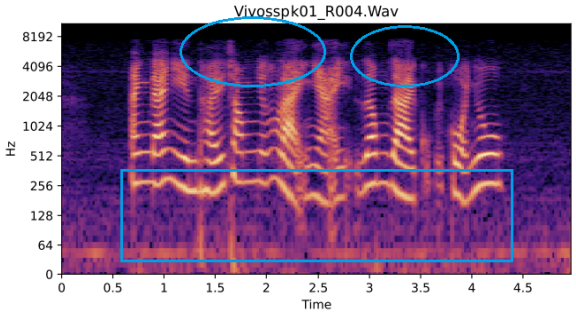
e. Fake data generated by VixTTS

Figure 8: Waveform of audio data, difference between raw real and fake data.

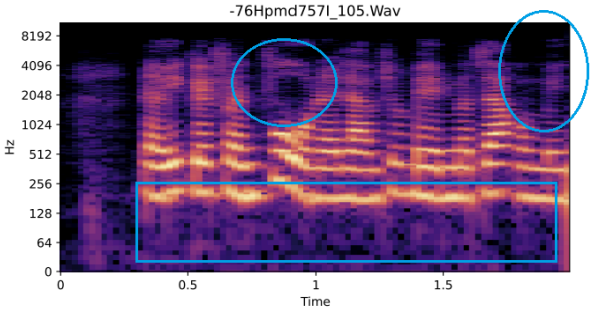
We perform a waveform analysis of the differences between real and fake data. A waveform is a basic visual representation of an audio signal that reflects how an amplitude changes over time. The graph displays the time on the horizontal (x) axis and the amplitude on the vertical (y) axis which can be thought of as loudness but it doesn't tell us what's happening to frequencies. Overall, we can see the noticeable difference that is the silence or the non-speech duration at the beginning and ending of the waveform (the red rectangles) of bona fide data. This difference is considered by the inconsistency of the speakers in recording real voice in VIVOS dataset. Another reason is punctuation of the speaker and the unavoidable problem in applying speech transcription Whisper model on podcast clips to get duration time. Basically, the speaker pausing in their flow of speech, then the Whisper model generates the end time at that pausing. That end time actually is the start time for the next duration, that is why there is the silence duration at the beginning of the waveform of podcast data. However, there are a few audio files in podcast data that have this problem. To handle silence duration, we apply trimming silence to remove this redundant duration. Beside that, the other difference is the amplitude at the end of each word in the utterance (the red ovals). In the fake data set, the amplitude equals 0 at the

bottleneck of the waveform, but in real data, at the similarity to the bottleneck, there is still the appearance of a low amplitude by the low noise in recording process or the low frequency oscillation of real human voices.

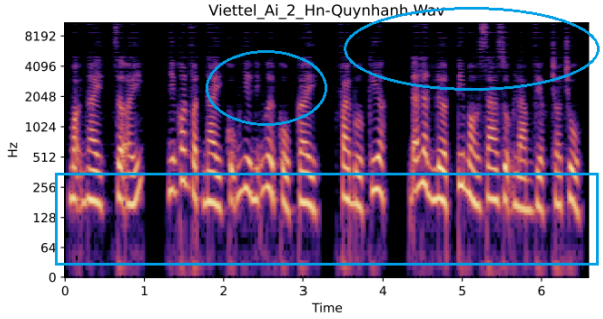
Beside visualization of waveform, we plot the spectrogram of sample audio in four sources of dataset to make a comparison. A spectrogram is a detailed view of a signal that covers all three characteristics of sound. You can learn about time from the x-axis, frequencies from the y-axis, and amplitude from color. The louder the event the brighter the color, while silence is represented by black. Having three dimensions on one graph is very convenient: it allows you to track how frequencies change over time, examine the sound in all its fullness, and spot various problem areas (like noises) and patterns by sight.



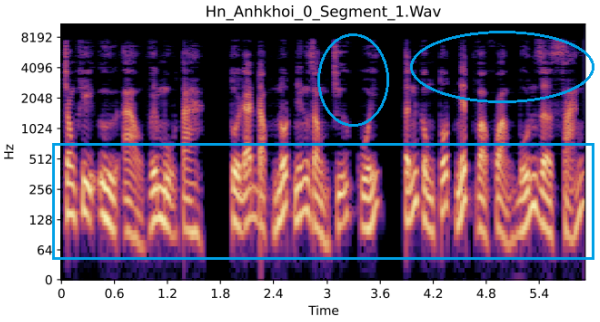
a. Real data collected VIVOS dataset



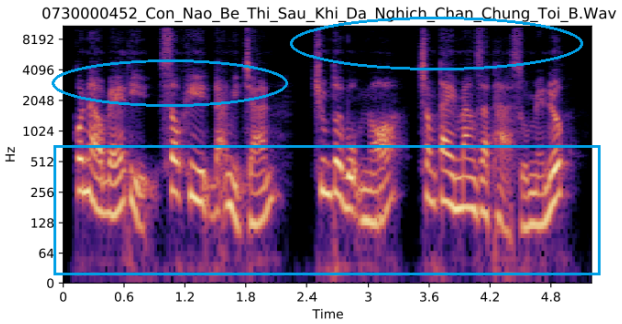
b. Real data crawled podcast



c. Fake data generated by Viettel TTS



d. Fake data generated by Vbee AIVoice



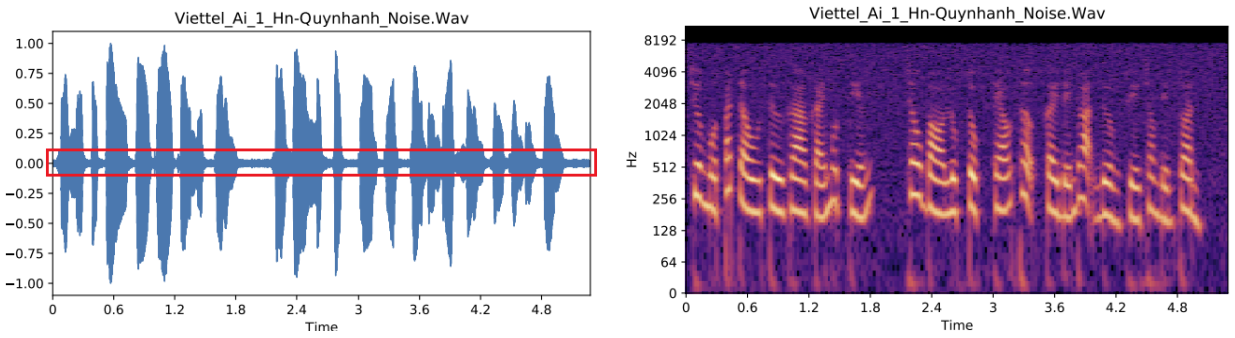
e. Fake data generated by VixTTS

Figure 9: Spectrogram of audio data, difference between raw real and fake data

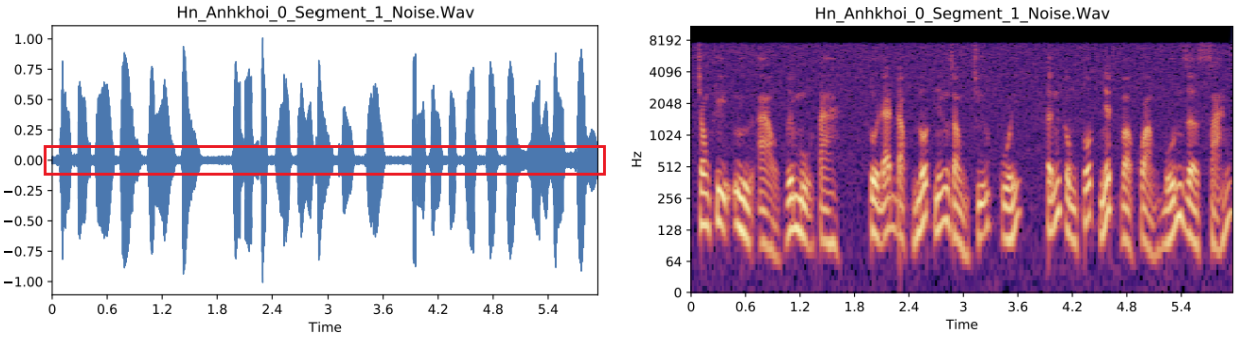
Focus on the overall of the spectrograms, especially the low frequency (the blue rectangles), we can see the sharp signal in the spectrogram of the fake data sets and the silence after each word. In contrast, spectrograms of real data, we can clearly see the low amplitude of sound which

makes the spectrogram smoother. This low amplitude can be the result of the voice modulation of the speaker or only the noise in real voice recording which can not clean completely. Beside that, in the spectrograms of spoofed speeches, the highest frequencies of each word which contain the energy are quite uniform. Moreover, the missing amplitudes that only appear at high frequency (blue ovals) are repeated at the same frequency levels. On the other hand, we clearly can not find that in the spectrograms of bona fide voices. The black colors in spectrograms occurred randomly and not only at the high frequency which are the result of the natural pronunciation of the speakers.

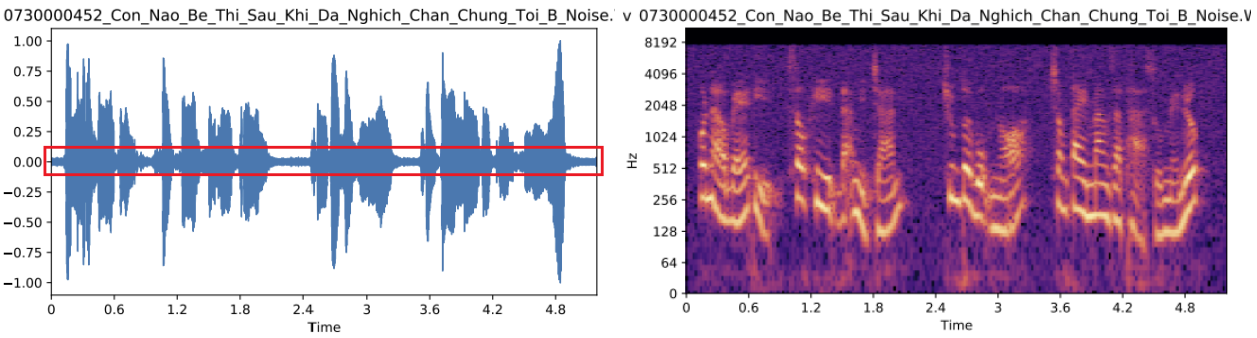
To improve the complexity of spoofed data, based on the difference at the end of each word in utterances and the sharpness in spectrogram image, half of the quantity of fake data is added to low noise to create the small oscillation in waveform of speech audio. We plotted the waveforms and spectrograms after added noise to practically consider the complexity enhanced in the visualizations of audio features.



a, Fake data generated by Viettel TTS



b, Fake data generated by Vbee AIVoice



c, Fake data generated by VixTTS

Figure 10: Waveform and spectrogram of fake audio data after added noise.

Compared to the raw waveforms in Figure 8, we notice the low amplitude throughout the plot (the red rectangles) which is the oscillation of the noise synthesized. It makes the waveforms more quite similar to the waveforms of real data with the non-silence after each word in speech. As mentioned before, the smoothness observed in the spectrograms of bonafide data contribute to the idea of adding noise in voice audios. We try to reduce the level of sharpness in spectrograms of spoofed voices. Moreover, we add noise in half of the quantity of fake data to enhance the prediction step in the demo such as in some cases, the fake voices can be re-recording.

To investigate further, we perform an additional, more fine-grained analysis by plotting a histogram of the energy contained in each frequency bin. We compute the average energy per frequency bin in Decibel of bona fide data (vivos+podcast) and spoofed data (viettel+vbee) by torchaudio library which is part of the PyTorch machine learning framework. Firstly, we convert all audio signals to spectrograms with torchaudio.transforms.Spectrogram. Next, apply torch.mean to each spectrogram in the list of spectrograms. Then, each mean spectrogram is converted to the Decibel Scale.

Lastly, applying torch.stack to convert the list of spectrograms which is the list of tensors into one tensor and applying another torch.mean to get average Decibel for each frequency bin. After all, we plot the histogram of computational results worked for separate real data and fake data. Additionally, we plot the relative difference between them.

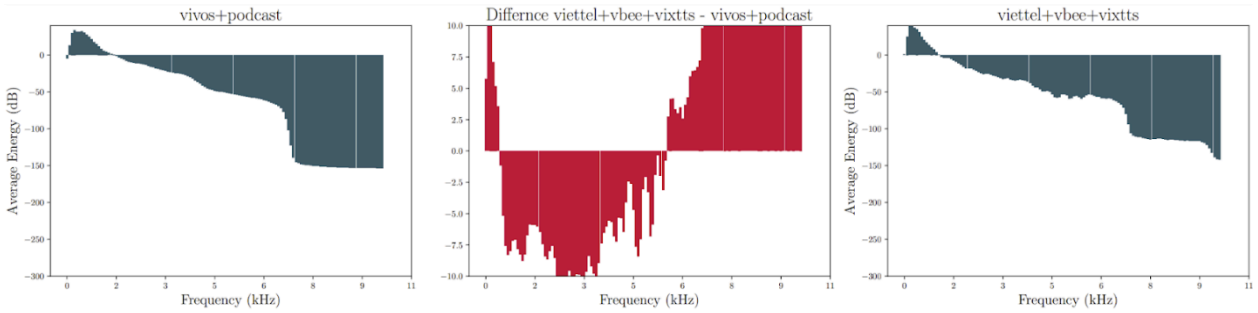


Figure 11: Average energy per frequency bin in dB of bona fide and spoofed data, the difference between two raw data sets.

Predicated on histograms of average energy, we can find out the difference in shape of two histograms. Generally, the difference is clearly noticed by the red plot in every frequency bin. The difference is the result of the variance in amplitude of audio files, basically is how loud the speaker's spoken recording or the generated voices is. This difference can prove the imbalance in our dataset regarding the energy of speeches from various Vietnamese speech sources. Due to the problem, we try to resolve it by preprocessing data that can improve the complexity of data for our model. Afterward, we compute and visualize the average energy of the preprocessed dataset.

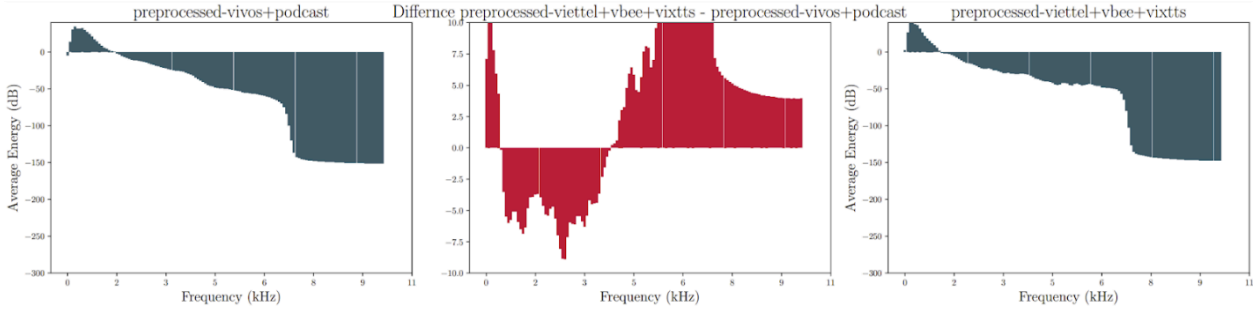


Figure 12: Average energy per frequency bin in dB of preprocessed bona fide and spoofed data, the difference between two preprocessed data sets.

The overall shape of histograms is identical, but the generated samples exhibit apparent differences, especially during 5500 Hz to 7000 Hz which are less crucial frequencies in common human voices. The differences at the low frequency bins are reduced than in the raw data.

And then, at the end of this section, we compute the fundamental frequency (pitch) by using normalized cross-correlation and median smoothing in the range 50−500 Hz for each real and fake data set. The perception of pitch is influenced by multiple acoustic parameters such as amplitude and resonant (or formant) frequencies; however, the primary determinant of pitch is fundamental frequency (F0), the rate of vocal fold vibration during phonation. Basically, pitch is the most salient property of the voice and describes its “highness” or “lowness”. Pitch is one of the most powerful cues for speech recognition.

	Real data			Fake data			
	VIVOS	Podcast	Real Aggregation	Vettel TTS	Vbee TTS	VixTTS	Fake Aggregation
Avg .Pitch	168.74	142.23	155.55	144.27	146.21	136.62	139.26
Std. Pitch	47.23	39.7	44.13	38.29	41.44	39.83	39.49

Table 3: Basic statistics for all collected Vietnamese real and fake datasets.

In this task, we do not use pitch as the audio feature in Fake Voice Detection, we utilize it to analyze our dataset. Men's average ranged from 78 to 182 Hz and women's from 126 to 307 Hz [9]. Average pitch of the overall dataset is standard in range of human pitch. However, the average pitch of the fake data set is lower than the real data set, which is dependent on the pitch of each speaker. Based on the average pitch standard deviation above, the pitch in spoofed data is less varied than bona fide data. It is also true that, number of speakers in real data is much higher than the rest.

4.2. Method

We utilize the RawNet2 [1] model, which incorporates advanced technologies compared to RawNet1. This includes the use of a SincNet [9] filter bank for preprocessing raw audio waves. Key innovations in RawNet2 are feature-wise map scaling (FMS) [10] and the integration of Gated Recurrent Units (GRUs). Additionally, we have enhanced the model by incorporating a Transformer encoder [11], which customizes the architecture further and improves training outcomes.

4.2.1. Overview Rawnet2

Rawnet2's architecture operates as an end-to-end system, consisting of a coder and a voder network. The coder transforms raw audio samples from the time domain into frame sequences within a frequency-like space. This transformation is governed by the stride of the convolutional layers and the pooling size, which together shape the acoustic features and determine the frame sizes. Once these features are generated, the voder network reconstructs the audio from them. This integrated process ensures efficient audio synthesis, making it particularly effective for detecting subtle anomalies in synthetic voices.

The voder network in Rawnet2 is designed with a simplified structure, utilizing only the current predicted sample and conditioning acoustic features as inputs. This design reduces both the complexity of the network and the time required for generating audio, making it highly suitable for real-time applications such as fake voice detection. By streamlining the input requirements, Rawnet2 improves the speed and efficiency of the audio generation process while maintaining accuracy.

To enhance the model's robustness and reduce the risk of overfitting, Rawnet2 employs noise injection during training. Gaussian noise is introduced into the input data, exposing the model to varied data and promoting better generalization. Additionally, a post-synthesis denoising process is applied to eliminate any residual noise, particularly in silent sections of the generated audio. This denoising step is crucial for maintaining the clarity of the speech, which is essential for accurately detecting anomalies in synthetic voices.

Rawnet2 employs a simple argmax method for sampling, which has proven effective in generating clear audio samples with minimal noise. Compared to more complex sampling methods, which can sometimes introduce unnecessary noise, the argmax approach produces clearer audio, enabling the detection of subtle differences and irregularities in synthetic speech. This clarity is critical for tasks like fake voice detection, where precision in audio generation plays a key role.

In subjective AB preference tests, Rawnet2 consistently outperformed other models in terms of the naturalness and preference of the generated speech. Participants frequently favored samples generated by Rawnet2, especially in speaker-independent scenarios. This superior performance highlights Rawnet2's ability to generate high-quality, natural-sounding speech, which is essential for accurately identifying synthetic voices.

Rawnet2's coder network is capable of learning and extracting high-level features directly from raw audio data, without relying on prior domain knowledge. The network automatically captures interpretable features, such as pitch information, improving its capacity to identify fake voices. By detecting subtle inconsistencies and unique characteristics in synthetic speech, Rawnet2 excels in distinguishing between genuine and synthetic audio, making it an effective tool in detecting fake voices.

4.2.2. Sincnet filter

The SincNet architecture, introduced by Mirco Ravanelli and Yoshua Bengio in their paper "Speaker Recognition from Raw Waveform with SincNet", utilizes Sinc functions as the basis for its convolutional filters. This approach ensures that the filters are more tailored to the specific nature of the input signal, which in this case is raw audio waveform data.

The low and high frequencies of the filters are initialized to ensure that their frequency responses are spaced uniformly on the Mel scale, which corresponds more closely to human auditory perception. By utilizing the transformation between Hertz and Mel scales, the filters are aligned to cover the frequency spectrum in a perceptually relevant manner.

Parameterization:
Filter parameters are defined by their lower cutoff frequencies and bandwidths. This parameterization reduces the number of trainable variables compared to traditional convolutional layers, which require each filter coefficient to be learned independently, making the model more efficient.

Hamming Window:
To reduce spectral leakage and improve frequency resolution, a Hamming window is applied to the filters. This windowing function smooths the edges of the filters, minimizing distortions in the frequency domain and enhancing filter performance.

Sinc Function:
At the core of each filter is the Sinc function, mathematically defined as:

$$\text{sinc}(x) = sin(\pi x)/\pi x.$$

Since the Sinc function is naturally band-limited, it is well-suited for designing filters that target specific frequency ranges.

Filter Design:
The filters are constructed to be symmetric, ensuring uniform response across the frequency spectrum. This symmetry is achieved by calculating only one-half of the filter and reflecting it to complete the design.

Forward Pass:
During the forward pass, the sinc-based filters convolve with the input waveform, producing feature maps that emphasize key frequency components of the signal. This convolution process is optimized to leverage modern GPU architectures for efficient computation.

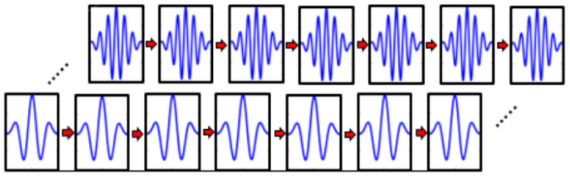


Figure 13: Sincnet filter

4.2.3. FMS

Feature Map Scaling (FMS) is a technique designed to adjust the contribution of different filters (or feature maps) during the forward pass of a neural network. This method helps in dynamically weighting the importance of each filter's output, enhancing the network's ability to focus on the most relevant features. Here is a detailed explanation based on the provided image and the concept of FMS:

The primary objective of the Feature Map Scaling (FMS) mechanism is to dynamically adjust the significance of feature maps produced by each filter in a convolutional layer, allowing the network to emphasize more relevant features while attenuating less important ones. This approach enhances the network's performance, particularly in complex tasks like fake voice detection or signal processing. The FMS mechanism is implemented using a fully connected (FC) layer, which processes the feature maps generated by the convolutional layers. The FC layer learns the relationships between different feature maps and determines their relative importance. The outputs from the FC layer are then passed through a sigmoid activation function, which scales the outputs to a range between 0 and 1, acting as a gating mechanism that modulates the contribution of each feature map during the forward pass.

The FMS mechanism operates by first taking the feature maps generated by the convolutional layers as input. These feature maps are fed into the fully connected layer, and the resulting outputs are scaled using the sigmoid function. Each feature map is then multiplied by its corresponding scaling factor, dynamically adjusting the contribution of the feature map to the network's decision-making process. This ensures that the most relevant features have a greater impact on the network’s output, while less relevant features are downscaled.

The benefits of FMS include enhanced feature discrimination, as the network can better differentiate between important and unimportant features. This is critical for tasks requiring detailed analysis and fine differentiation, such as detecting fake voices or other complex signal processing tasks. Additionally, FMS improves learning efficiency by focusing the network's attention on the most relevant features, which can lead to faster convergence and improved generalization. The dynamic nature of FMS also makes the network adaptable, allowing it to handle various types of input data with differing feature importance, making it versatile for a range of tasks.

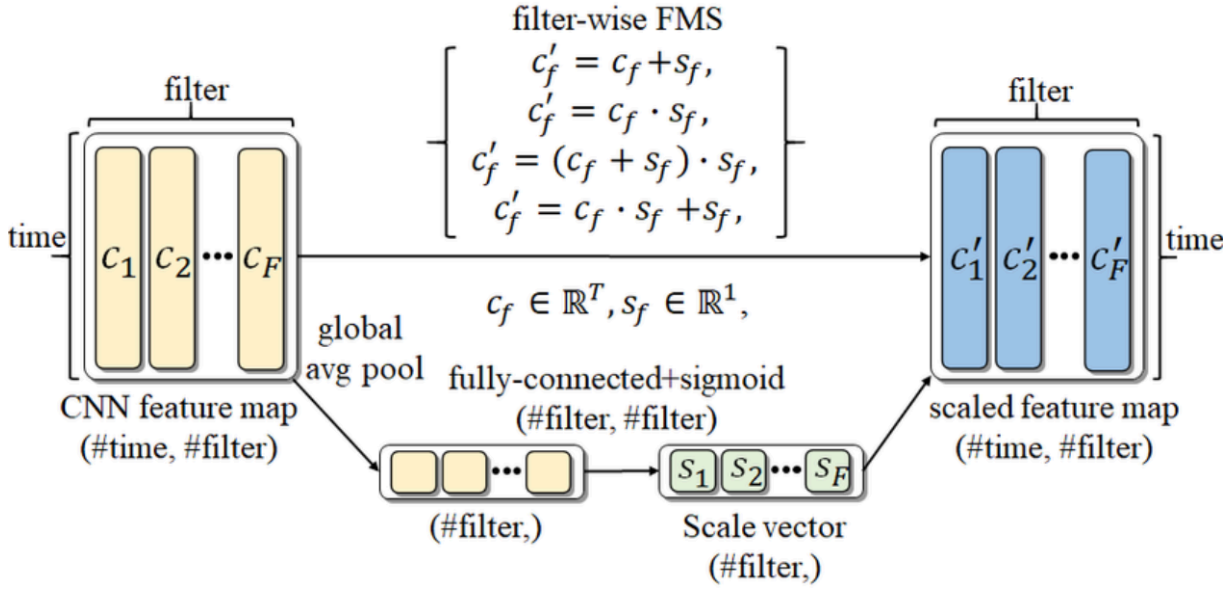


Figure 14: Feature Map Scaling

4.2.4. GRU

The **Gated Recurrent Unit (GRU)** is a type of recurrent neural network (RNN) architecture designed to handle sequential data and retain long-term dependencies without the complexity of traditional Long Short-Term Memory (LSTM) networks. GRUs are known for their efficiency and ability to learn temporal patterns effectively, making them suitable for various tasks such as language modeling, time-series prediction, and speech recognition.

The Update Gate (Zt) in a Gated Recurrent Unit (GRU) determines how much of the previous hidden state should be passed to the current hidden state, balancing the retention of past information with the integration of new input. This gate decides whether to keep the existing memory or update it with new information. Mathematically, the update gate is expressed as:

$$Z_t = \sigma(X_t W_{xz} + H_{t-1} W_{hz} + b_z),$$

where X_t is the current input, H_{t-1} is the previous hidden state, W_{xz} and W_{hz} are weight matrices, b_z is the bias vector, and σ is the sigmoid activation function.

The Reset Gate (Rt) controls how much of the previous hidden state should be ignored when incorporating new information. This gate determines how much of the past information is relevant for the current step. The formula for the reset gate is:

$$R_t = \sigma(X_t W_{xr} + H_{t-1} W_{hr} + b_r),$$

where W_{xr} and W_{hr} are the weight matrices, and b_r is the bias vector. The reset gate is key for determining whether the network should "forget" previous memory content.

The Candidate Activation (Ht) represents the new memory content to be potentially added to the current hidden state. It is a combination of the current input and the previous hidden state,

modulated by the reset gate, which controls how much past information influences the current activation. The candidate activation is given by:

$$\widetilde{H}_t = \tanh(X_t W_{xh} + \left(R_t \odot H_{t-1}\right) W_{hh} + b_h),$$

where $\odot$ denotes element-wise multiplication and tanh is the hyperbolic tangent activation function. This formula computes the new candidate memory content, incorporating both current input and selectively ignored past information.

Finally, **the Final Hidden State** (Ht) is a linear interpolation between the previous hidden state and the candidate activation, with the update gate controlling the degree to which the candidate activation influences the new hidden state. The formula for the final hidden state is:

$$H_t = Z_t \odot H_{t-1} + (1 - Z_t) \odot \widetilde{H}_t,$$

where Z_t is the update gate, and $\odot$ represents element-wise multiplication. This interpolation allows the GRU to efficiently retain relevant information from previous time steps while integrating new input.

GRUs offer several advantages. They have a **simplified structure** compared to Long Short-Term Memory (LSTM) networks, with fewer gates and no separate memory cell, leading to faster training times and lower computational costs. Despite their simplicity, GRUs are capable of **efficiently learning long-term dependencies**, making them particularly effective in tasks that require modeling temporal patterns over long sequences. Moreover, the gating mechanisms in GRUs help mitigate the **vanishing gradient problem**, allowing the network to retain and propagate gradients effectively during backpropagation.

In conclusion, GRUs provide a powerful yet efficient mechanism for modeling sequential data. By simplifying the architecture relative to LSTMs while retaining essential information over long sequences, GRUs are well-suited for various applications such as natural language processing and time-series forecasting, where modeling temporal dependencies is crucial.

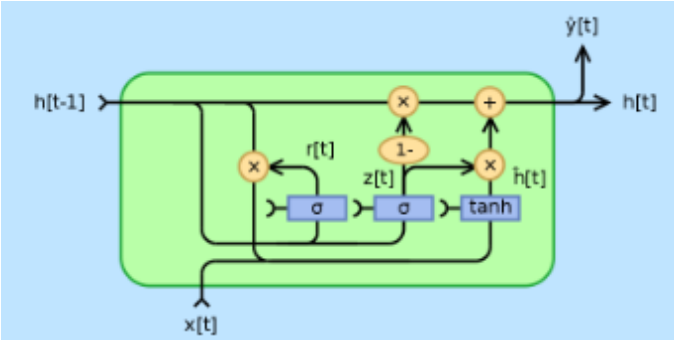


Figure 15: Gated Recurrent Unit

4.2.5. Transformer

The Transformer Encoder, introduced by Vaswani et al. in "Attention is All You Need," has transformed the landscape of natural language processing and other sequence-based tasks due to its ability to capture long-range dependencies and its high parallelization efficiency during training. The architecture is built on several core components that contribute to its powerful performance.

One of the foundational components is **input embedding**, which converts raw input tokens, such as words or subwords in text data, into dense vectors of fixed size. This embedding process is essential for transforming symbolic input into numerical representations that can be processed by the model. To compensate for the lack of inherent order understanding in the Transformer architecture, **positional encoding** is added to the input embeddings. Positional encodings are computed using sine and cosine functions of different frequencies, allowing the model to distinguish the position of each token in the sequence. The positional encoding is defined as:

$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}),$$

$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}),$$

where pos represents the position in the sequence and i denotes the embedding dimension.

The multi-head self-attention mechanism is central to the Transformer Encoder's ability to capture dependencies across the entire input sequence. In this mechanism, each position in the sequence can attend to every other position, regardless of distance, which allows the model to learn contextual relationships. The scaled dot-product attention is expressed as:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})\,V,$$

where Q (queries), K (keys), V (values) are derived from the input embeddings, and d_k is the dimension of the keys.

To enhance the model's expressiveness, **multi-head attention** is applied, where multiple attention heads allow the model to attend to information from different subspaces. The outputs from the multiple heads are concatenated and then linearly transformed.

Following the attention mechanism, each position independently passes through a **feed-forward neural network (FFN)**. This FFN consists of two linear transformations with a ReLU activation in between, expressed as:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2.$$

The FFN is applied identically to each position, further refining the representations learned by the attention mechanism.

To ensure stability and support the training of deeper models, **residual connections** and **layer normalization** are applied after both the multi-head attention and FFN sublayers. The residual connection enables the model to bypass the non-linear transformations, which helps prevent vanishing gradients and aids in stabilizing the training process. Layer normalization is then applied to the sum of the input and the output of each sublayer, ensuring that each layer's output maintains stable distributions. This is represented as:

$$LayerNorm(x + Sublayer(x)).$$

A **Transformer Encoder block** consists of these components in a specific sequence, often stacked NNN times to form the full encoder. The input sequence is first embedded and combined with positional encodings, after which it passes through the multi-head self-attention mechanism. A residual connection and layer normalization are applied, followed by the feed-forward neural network. Another residual connection and layer normalization complete the block. These layers are crucial for transforming the input data into useful representations for downstream tasks.

Incorporating a Transformer Encoder layer into neural networks for processing voice data can significantly improve the model's ability to understand and recognize patterns over time. The self-attention mechanism, by capturing global dependencies within the input sequence, makes it an ideal solution for tasks such as speech recognition and speaker identification, where context and sequential relationships are vital. By leveraging the advanced capabilities of the Transformer architecture, superior results can be achieved in various sequence-based tasks.

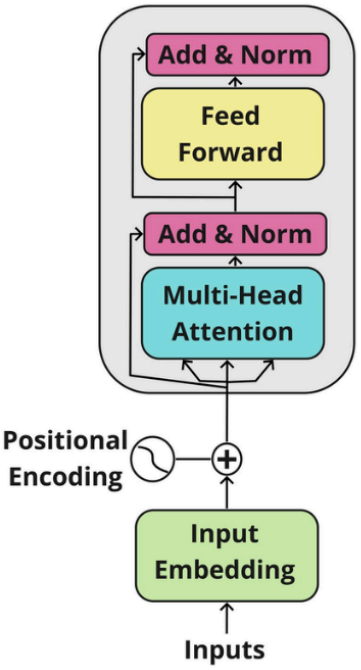


Figure 16: Transformer Encoder

4.2.6. Resblocks

The Residual Block (Resblock) is a crucial component in deep neural network architectures, particularly in Residual Networks (ResNets). Its introduction has enabled the effective training of very deep networks by addressing the vanishing gradient problem, a common challenge in deep learning. This problem arises when gradients become exceedingly small during backpropagation, which hampers the learning process. Resblocks are designed to allow gradients to flow more efficiently through the network, resulting in improved learning performance, particularly in architectures with many layers.

At the core of a Resblock is the concept of **residual learning**, which involves the addition of a shortcut or skip connection that bypasses one or more layers. This shortcut connection enables the input to the Resblock to be directly added to the output, forming what is known as a residual function. Rather than forcing the network to learn a direct mapping from input to output, residual learning focuses on modeling the difference, or residual, between the input and the target mapping. This approach not only preserves the gradient flow across layers but also simplifies the learning process, allowing the network to focus on refining the residual rather than attempting to learn the entire transformation from scratch.

One of the key advantages of Resblocks is their ability to **enhance gradient flow** through the network. The skip connections help ensure that gradients can propagate back through the layers more effectively, preventing them from diminishing as they travel through deep networks. This addresses the vanishing gradient problem, which is particularly problematic in very deep architectures, and enables the training of networks with significantly more layers than was previously feasible.

In addition to improved gradient flow, Resblocks contribute to **improved learning efficiency**. By focusing on residual mappings, the network simplifies the task of learning complex functions, as it only needs to learn how to adjust the residual between the input and the output. This often leads to faster convergence and more efficient learning compared to networks without residual connections.

Finally, Resblocks also aid in **better generalization**. By enabling the network to learn complex functions more effectively, Resblocks contribute to improved performance not only on training data but also on unseen data, leading to better overall generalization. This makes Resblocks a valuable tool in modern deep learning architectures, particularly in applications that require the training of very deep networks.

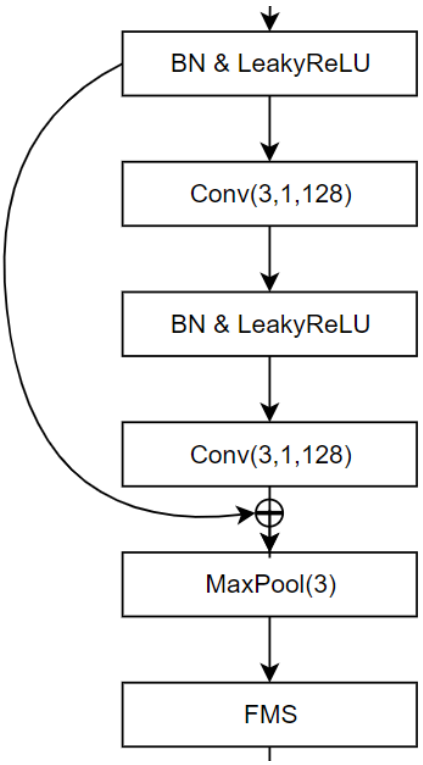


Figure 17: Resblock

4.2.7. Base model Rawnet2

In this network architecture for audio processing, the model begins by ingesting raw audio samples in the form of a waveform. The input consists of 64,000 raw samples, which typically correspond to several seconds of audio, depending on the sample rate used (e.g., 16 kHz results in 4 seconds). This input layer is designed to capture and represent the raw audio data for further processing by the subsequent layers of the network.

The first major operation applied to the input is a convolution through a fixed Sinc filter layer. This layer consists of 129 fixed, non-trainable filters modeled on the Sinc function, which act as band-pass filters. These filters decompose the audio signal into different frequency components, generating 128 feature channels. Following this convolutional operation, a max-pooling layer with a kernel size of 3 reduces the temporal dimension of the signal from 64,000 to 21,290 samples. Batch normalization is applied to stabilize the learning process, followed by the LeakyReLU activation function, introducing non-linearity to enable the network to capture more complex patterns in the audio signal. The resulting output from this layer has a shape of (21,290, 128), corresponding to the reduced time dimension and the 128 feature channels.

Next, the network refines the extracted features through a series of residual blocks. In the first residual block, batch normalization and LeakyReLU activation are used, followed by a convolutional layer with 128 filters and a kernel size of 3, which processes the features. Another convolutional layer further refines the feature map, maintaining the 128-channel depth. A max-pooling operation is applied to reduce the temporal dimension once more. To facilitate gradient flow during training, a skip connection adds the input to the output of the block, ensuring that information from earlier layers is preserved. This structure is repeated twice, ultimately yielding an output shape of (2,365, 128), with the time dimension further reduced due to pooling operations.

Following this, a second residual block performs similar operations, but with a larger feature depth. Batch normalization and LeakyReLU activation are again applied, followed by a 1D convolutional layer with 512 filters. The depth of the feature space is expanded to 512 channels, and another convolutional layer continues processing the data at this depth. Max-pooling is employed once again to reduce the temporal size. As with the first residual block, a skip connection is used to stabilize training by summing the input and output of the block. This structure is repeated four times, further refining the features and reducing the temporal dimension significantly. The output at this stage is shaped (29, 512), a heavily compressed version of the original input but enriched with complex feature representations.

Once the residual blocks have extracted and refined the features, a Gated Recurrent Unit (GRU) layer processes the output. The GRU, which consists of 1,024 hidden units, is well-suited for capturing long-term dependencies in sequential data, such as audio. By passing the refined features through the GRU, the network summarizes the entire sequence into a single 1,024-dimensional vector, which encodes important temporal information about the input audio signal.

This 1,024-dimensional vector is then passed through a fully connected layer, which transforms the features into a higher-level representation while maintaining the same dimensionality. Finally, the network produces a 2-dimensional output, typically used for binary classification tasks. In the context of speaker verification, for instance, this output could represent two classes, such as "same speaker" or "different speaker." Thus, the network processes raw audio input and transforms it into a representation suitable for classification by progressively refining and abstracting features through its various layers.

Layer	Input 64,000 samples		Output shape
Fixed Sinc filters	Conv(129,1,128) Maxpooling(3) BN & LeakyReLU		(21290,128)
Res block	BN & LeakyReLU Conv(3,1,128) BN & LeakyReLU Conv(3,1,128) Maxpooling(3) FMS	x2	(2365,128)
Res block	BN & LeakyReLU Conv(3,1,512) BN & LeakyReLU Conv(3,1,512) Maxpooling(3) FMS	x4	(29,512)
GRU	GRU(1024)		(1024)
FC	1024		(1024)
Output	1024		2

Figure 18: Rawnet2 model

4.2.8. Adding Transformer Encoder Layer to base model

The Transformer Encoder has emerged as a powerful neural network architecture for capturing intricate relationships within sequential data. Initially introduced in the context of Natural Language Processing (NLP) through the landmark work "Attention is All You Need," this architecture has since been successfully adapted for various domains, including speech and audio processing. In contrast to traditional recurrent neural network (RNN) models such as GRUs and LSTMs, which process data sequentially, the Transformer Encoder leverages a self-attention mechanism. This mechanism enables the model to focus on multiple parts of the input sequence in parallel, allowing it to capture both short-term and long-term dependencies with greater efficiency. As a result, it is particularly well-suited for tasks requiring an understanding of global context across a sequence.

In this model, the input to the Transformer Encoder is derived from the second residual block, which outputs a representation with a shape of (29, 512). This input consists of 29 time steps, each containing 512 feature channels that represent a refined and downsampled version of the original audio waveform. These features are further processed through the Transformer Encoder to capture more sophisticated dependencies within the sequence.

At the heart of the Transformer Encoder is its self-attention mechanism, which calculates the relationships between different positions in the input sequence. For each time step, the model dynamically computes "attention scores" that indicate the relevance of other time steps to the current one. This allows the model to focus on important time steps, regardless of their proximity, enabling it to capture dependencies across the entire sequence. The self-attention mechanism is particularly useful for handling complex data such as audio, where relevant features may be distributed throughout the sequence.

Since Transformer Encoders lack the inherent ability to model the order of the sequence, positional encodings are added to the input. These encodings provide the model with information about the relative positions of time steps within the sequence, preserving the temporal structure of the audio data. This step ensures that the model retains awareness of the sequential nature of the input, which is crucial for processing time-series data like speech or audio.

To enhance the model’s ability to capture diverse patterns within the sequence, the Transformer Encoder employs multi-head attention. This involves splitting the input into multiple "heads", each of which independently applies the attention mechanism. The results from these heads are then combined, allowing the model to process the input at different levels of abstraction simultaneously. This multi-head attention mechanism helps the model identify both local and global patterns in the audio data.

After the attention mechanism has been applied, the Transformer Encoder processes the features further through a feed-forward neural network, which adds an additional layer of abstraction to the learned representations. To stabilize the learning process and improve training efficiency, layer normalization is applied at this stage. The combination of self-attention, feed-forward processing, and normalization allows the Transformer Encoder to transform the input into a

more robust representation that better captures both local and global dependencies within the audio sequence.

The output from the Transformer Encoder retains the same shape as its input (29, 512), but the features have been enriched through the self-attention mechanism. These transformed features are then passed on to subsequent layers, such as a GRU, for further sequential processing. By refining the input features, the Transformer Encoder enhances the model's ability to extract meaningful patterns from the audio signal, ultimately contributing to improved performance on tasks such as speaker recognition or verification.

Adding a Transformer Encoder to the model introduces several advantages. First, the ability to capture long-range dependencies efficiently is a key benefit. Unlike RNNs, which process sequential data step-by-step, the Transformer Encoder can analyze relationships across the sequence in parallel. This not only reduces computational bottlenecks but also allows the model to better handle long sequences of audio data. Furthermore, the parallelization enabled by the self-attention mechanism significantly speeds up both training and inference, making the model more scalable for large datasets or long audio sequences.

Additionally, the self-attention mechanism within the Transformer Encoder provides the model with improved contextual understanding of the input. By dynamically adjusting the importance of different time steps based on their relevance, the model is able to better capture the nuances of the audio signal. This flexibility is particularly advantageous in tasks where the relevant features are subtle or distributed across the sequence. Moreover, the ability to handle variable-length sequences makes the Transformer Encoder well-suited for real-world applications where audio inputs may vary in duration.

Overall, the integration of a Transformer Encoder enhances the model’s ability to extract and process complex features, making it more effective for tasks like speaker recognition or verification. By allowing the model to consider both local and global interactions within the sequence, the Transformer Encoder helps achieve a richer, more comprehensive understanding of the input data, leading to improved generalization and performance in challenging scenarios.

Layer	Input 64,000 samples		Output shape
Fixed Sinc filters	Conv(129,1,128) Maxpooling(3) BN & LeakyReLU		(21290,128)
Res block	BN & LeakyReLU Conv(3,1,128) BN & LeakyReLU Conv(3,1,128) Maxpooling(3) FMS	x2	(2365,128)
Res block	BN & LeakyReLU Conv(3,1,512) BN & LeakyReLU Conv(3,1,512) Maxpooling(3) FMS	x4	(29,512)
Transformer Encoder	Encoder(512)		(512)
GRU	GRU(1024)		(1024)
FC	1024		(1024)
Output	1024		2

Figure 19: Rawnet2 with transformer encoder layer

4.3. Evaluation

The evaluation of machine learning models is crucial to assess their performance, particularly in classification tasks. This section presents the evaluation metrics and methodologies employed to assess the efficacy of the proposed fake voice detection model. By using a combination of loss, accuracy, the Receiver Operating Characteristic (ROC) curve [12], Equal Error Rate (EER) [13], and F1-score, we provide a comprehensive analysis of the model's ability to generalize and perform reliably across different datasets and evaluation phases.

4.3.1. Loss and Accuracy

Loss and accuracy are fundamental metrics used to evaluate the performance of a machine learning model, particularly in classification tasks. In the context of our fake voice detection system, these metrics provide a direct measure of how well the model is learning from the data during training and how accurately it can classify unseen data during testing.

Loss is a measure of the model's prediction error, calculated as the difference between the predicted output and the actual target. It reflects how well or poorly the model is performing, with lower values indicating better performance. Loss functions are typically minimized during training to improve the model's predictive accuracy. In our model, we likely used a categorical cross-entropy loss function, which is well-suited for multi-class classification problems like fake

voice detection. This function penalizes predictions that deviate from the true labels, driving the model to become more accurate over time.

Accuracy is the proportion of correct predictions made by the model out of all predictions. It is a straightforward metric that provides an overall indication of the model's performance. High accuracy suggests that the model can correctly classify a significant portion of the samples in both the training and testing datasets. However, accuracy alone does not always give a complete picture, especially in cases where class imbalances exist. Thus, it is often supplemented with other metrics such as precision, recall, and the ROC curve.

In our system, we track loss and accuracy across different phases of model training and evaluation:

- Training Accuracy and Loss: These metrics indicate how well the model is fitting the training data. Overfitting can be detected if the training accuracy is significantly higher than the validation accuracy, accompanied by a decreasing training loss and a stable or increasing validation loss.
- Validation Accuracy and Loss: These are monitored to assess the model's ability to generalize to unseen data. A consistent or improving validation accuracy with a corresponding decrease in validation loss is desirable, indicating that the model is not only learning the training data but also performing well on new data.
- Test Accuracy and Loss: After finalizing the model, it is evaluated on a separate test set to provide an unbiased estimate of its real-world performance. High test accuracy and low test loss are indicative of a model that is both effective and reliable for deployment.

By carefully analyzing the loss and accuracy during training, validation, and testing, we ensure that our fake voice detection model achieves optimal performance, minimizing errors while maximizing correct classifications. This forms the foundation upon which more detailed analyses, such as the ROC curve and EER, are built, providing a comprehensive understanding of the model's effectiveness in practical scenarios.

4.3.2. ROC Curve

The Receiver Operating Characteristic (ROC) curve is a widely-used graphical representation that illustrates the diagnostic performance of a binary classifier as its discrimination threshold is varied. The ROC curve is constructed by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) across different threshold levels. This curve provides a detailed assessment of the model's ability to distinguish between the two classes under various threshold settings.

Mathematically, the TPR (also known as sensitivity or recall) is defined as:

$$TPR = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Negatives\ (FN)}.$$

This metric represents the proportion of actual positive cases that are correctly identified by the model. Conversely, the FPR is calculated as:

$$FPR = \frac{False\ Positives\ (FP)}{True\ Positives\ (TP) + False\ Negatives\ (FN)}.$$

FPR reflects the proportion of actual negative cases that are incorrectly classified as positive by the model.

The ROC curve originates from the bottom-left corner of the plot, where both TPR and FPR are zero, and progresses toward the top-right corner as the threshold decreases. An ideal classifier would achieve a point in the top-left corner, which corresponds to a high TPR and a low FPR. A diagonal line from the bottom-left to the top-right corner of the plot represents the performance of a random guess classifier, where the model has no discriminative ability between the classes.

To quantitatively evaluate the overall performance of the classifier, the Area Under the ROC Curve (AUC) is employed. The AUC value ranges from 0 to 1, with a value closer to 1 indicating superior model performance. Specifically, an AUC of 1 signifies perfect classification, while an AUC of 0.5 implies that the model performs no better than random guessing. The AUC provides a single scalar value that summarizes the model's effectiveness across all possible thresholds, making it a robust metric for evaluating the model's generalization capability.

The ROC curve and AUC are particularly useful in scenarios where the consequences of false positives and false negatives are significant, as they offer a more comprehensive evaluation of the model's performance than accuracy alone. This approach is crucial for applications such as fake voice detection, where the ability to accurately distinguish between genuine and fake audio is essential.

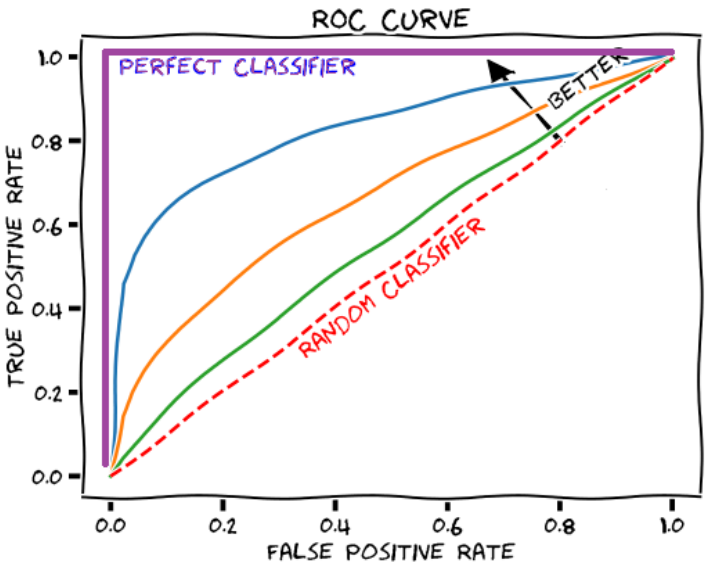


Figure 20: ROC Curve plotted when threshold β is varied

4.3.3. Equal Error Rate (EER)

The Equal Error Rate (EER) serves as a pivotal performance metric in evaluating binary classification systems by identifying the point at which the False Positive Rate (FPR) equals the False Negative Rate (FNR). This intersection provides a single value that effectively balances

the two types of classification errors, making EER a valuable measure for comparing the effectiveness of different classifiers or biometric systems.

At this threshold, the model exhibits an equal rate of false positives and false negatives, offering a clear and concise indicator of its performance.

In the context of our fake voice detection system, the EER allows for a comprehensive evaluation by ensuring that the model maintains an optimal balance between misclassifying genuine voices as fake and failing to detect fake voices. Achieving a low EER is crucial for real-world applications where both types of errors carry significant consequences. Minimizing false negatives is essential to prevent the undetected use of fake voices, while reducing false positives is important to avoid incorrectly flagging legitimate voices as fraudulent. Thus, the EER provides a robust measure of the model's ability to perform reliably under varying conditions, highlighting its resilience and suitability for deployment in environments where accuracy and balance are paramount.

By incorporating the EER metric alongside other evaluation measures, we ensure that our fake voice detection model not only achieves high accuracy but also maintains a critical equilibrium between different classification errors. This balanced approach is fundamental for developing a trustworthy and effective system capable of performing consistently in diverse real-world scenarios.

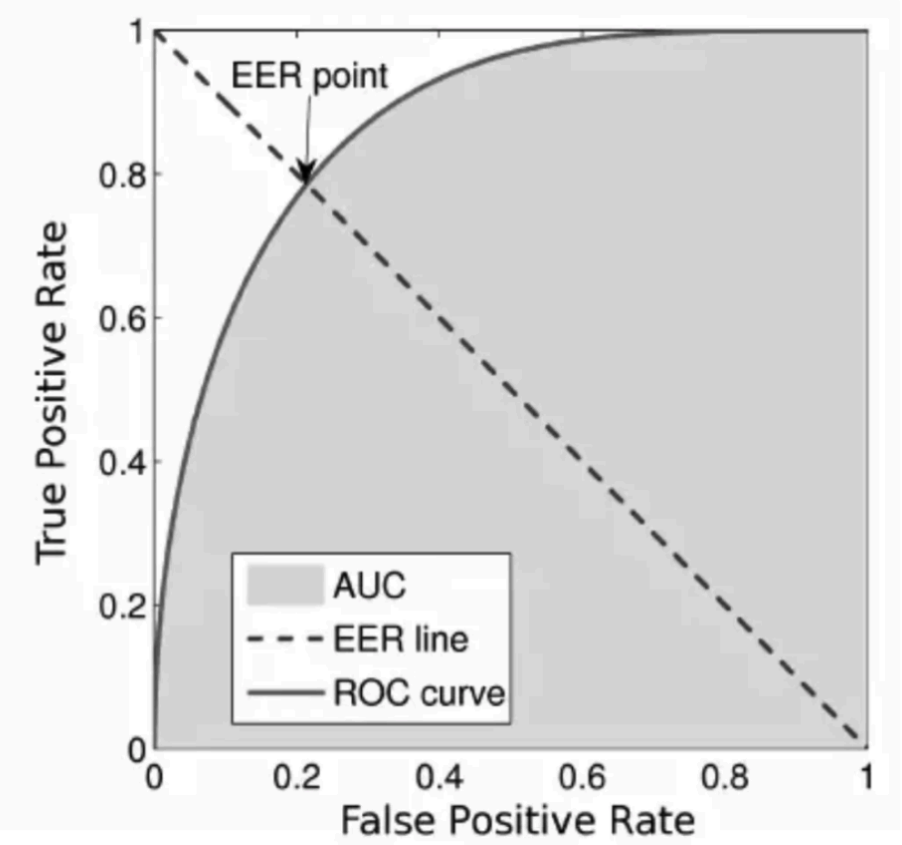


Figure 21: Illustration of the Equal Error Rate (EER)

4.3.4. F1-score

For evaluating our Vietnamese fake voice detection model, we employed the F1-score, which is the harmonic mean of precision and recall. This metric is particularly valuable in situations where it is crucial to balance false positives and false negatives. The F1-score is calculated using the formula:

$$F1\text{–}score = 2 \times \frac{Precision \times Recall}{Precision + Recall}.$$

- **Precision:** The ratio of correctly predicted positive observations to the total predicted positives, measuring a classifier's exactness.

$$Precision = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Positives\ (FP)}.$$

- **Recall:** The ratio of correctly predicted positive observations to all observations in the actual class, measuring a classifier's completeness.

$$Recall = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Negatives\ (FN)}.$$

The F1-score is critical in scenarios where both false positives and false negatives have significant consequences. By balancing precision and recall, the F1-score provides a more accurate measure of misclassified cases than the accuracy metric alone. This is particularly important for our application, where accurately detecting fake voices and minimizing the misclassification of genuine voices as fake are both essential.

Using the F1-score to evaluate our customized Rawnet2 model allows for a nuanced understanding of its performance, especially in the context of an unbalanced dataset where precision and recall might be misleading if considered separately. The model demonstrates robustness in its ability to generalize across different datasets, ensuring reliable performance in real-world applications.

By incorporating these evaluation methodologies, we offer a comprehensive analysis of the model's effectiveness, highlighting its strengths and areas for potential improvement. This thorough evaluation ensures that the model is well-prepared for deployment in scenarios requiring reliable and accurate voice verification.

5. EXPERIMENTS

This section outlines the critical components and methodologies employed in the development and training of our Vietnamese fake voice detection model. It encompasses the data preprocessing pipeline, advanced training enhancements, and the hardware infrastructure that collectively contribute to the model's robustness and efficiency.

5.1. Data Loader

The audio data preprocessing pipeline begins with the loading of audio files, which are then split into training and valid sets in an 8:2 ratio. This division ensures that the model is trained and evaluated on different subsets of data, improving the model's ability to generalize to new, unseen data.

After splitting the data, each audio file is converted into a waveform representation. This step is crucial as it transforms the audio signal into a numerical format that can be processed by machine learning models. Once converted, the waveform undergoes trimming to remove silence at the beginning and end of the audio file. This trimming step is essential to eliminate non-informative parts of the audio, allowing the model to focus on the relevant signal.

Following the trimming, each audio file is labeled, typically for a supervised learning task. Labels such as '1' or '0' may represent different classes, like 'speech' versus 'non-speech' or 'real' versus 'fake' audio, depending on the specific application.

To address the variability in waveform lengths - especially in datasets where the audio clips vary significantly in duration - the waveforms are padded to ensure uniform length. Padding is done to make each input consistent in size, which is required by most deep learning models. The padding procedure involves either truncating longer waveforms or repeating shorter ones to reach a predefined length, which, in this case, is 4 seconds. This consistency allows for efficient batch processing during model training.

This preprocessing pipeline ensures that the audio data is effectively prepared for input into machine learning models, particularly those focused on tasks like speech recognition, audio classification, or fake audio detection. By focusing on essential aspects of the audio signal, the pipeline enhances the model's ability to learn and make accurate predictions.

5.2. Advanced Training Enhancements

5.2.1. Loss Calculation Using BCEWithLogitsLoss

In binary classification problems, choosing the suitable loss function is critical for successful machine learning model training. BCEWithLogitsLoss from PyTorch is especially well-suited for these applications since it combines the sigmoid activation function and binary cross-entropy

loss into a one operation. This combination improves numerical stability by employing a solitary sigmoid layer followed by the BCELoss function. The loss is calculated as:

$$Loss = \frac{1}{N}\sum_{i=1}^{N}[y_i log(\sigma(p_i)) + (1 - y_i) log(1 - \sigma(p_i))].$$

BCEWithLogitsLoss is particularly beneficial in scenarios where model stability and performance are crucial, such as in binary classification tasks like fake voice detection. The function effectively measures the difference between the predicted logits and the true labels, guiding the model to minimize this discrepancy and, consequently, improve accuracy over time.

By penalizing incorrect predictions more effectively, BCEWithLogitsLoss encourages the model to distinguish between the two classes more sharply, leading to better overall performance. Additionally, the stability provided by this loss function helps ensure a smoother training process with fewer numerical issues, such as vanishing gradients, which can hinder the learning process.

In summary, BCEWithLogitsLoss is a robust choice for binary classification tasks, offering both practical and theoretical advantages that enhance model training and performance. Its ability to seamlessly integrate sigmoid activation with cross-entropy loss ensures that the training process is both efficient and stable, making it ideal for tasks that require precise binary classification, such as detecting fake voices.

5.2.2. Implementing Early Stopping

Early stopping is a technique used to prevent overfitting during training by halting the process when validation loss stops decreasing. This ensures the model does not continue to learn from noise in the training data. The implementation involves:

The model's training loss is monitored throughout the epochs. If an epoch's training loss is lower than the best recorded loss, the best loss is updated, and the non-improvement count is reset.

Incremented if the training loss does not significantly improve (i.e., the decrease is below a defined threshold).

If the non-improvement count exceeds a defined patience threshold, training stops, and the event is logged. This threshold balances allowing sufficient learning time and preventing overfitting.

Early stopping ensures the model halts at an optimal performance point, conserving computational resources and preventing over-complexity. This is particularly useful when computational efficiency and resource limitations are significant considerations.

Incorporating advanced training techniques like BCEWithLogitsLoss for a stable loss function, dynamic learning rate adjustments with ReduceLROnPlateau, and early stopping, provides a substantial advantage in developing a robust fake voice detection model. These methods enhance the model's performance on unseen data and make the training process more efficient by avoiding unnecessary computations after convergence. Applying these techniques in the

training pipeline is essential for achieving high accuracy and robustness in the final model, ensuring its effectiveness in real-world applications.

5.3. Hardware Resources for Training

A powerful computational environment was required to train the Vietnamese false voice detection model. We used an NVIDIA RTX 4060 Ti GPU with 16 GB of memory, which provided robust computing capabilities required for deep learning. The GPU's considerable memory allows for speedier training times by efficiently handling big batch sizes.

The machine has 62 GB of RAM, which allows huge datasets to be loaded rapidly and minimizes training delays. The CPU's 32 threads enable multi-threaded activities, which are critical for data preparation and parallel processing workloads.

This hardware configuration strikes the ideal mix between cost and performance, resulting in efficient and fast training. The combination of a strong GPU, plenty of RAM, and a multi-threaded CPU produces an environment ideal for the heavy demands of training deep learning models, resulting in a robust and accurate false speech detection system.

6. RESULTS

6.1. Training Performance of the Customized RawNet2 Model

The development of the Vietnamese fake voice detection model utilized a tailored version of the Rawnet2 model, specifically customized for this task. This section examines the model's performance across training epochs, focusing on accuracy and loss metrics for both training and validation sets. The results underscore the model's learning capabilities and its ability to generalize to unseen data.

The customized Rawnet2 model exhibited strong performance throughout the training process. The following analysis provides a detailed look at the trends in accuracy and loss, illustrating the model's learning progression and effectiveness.

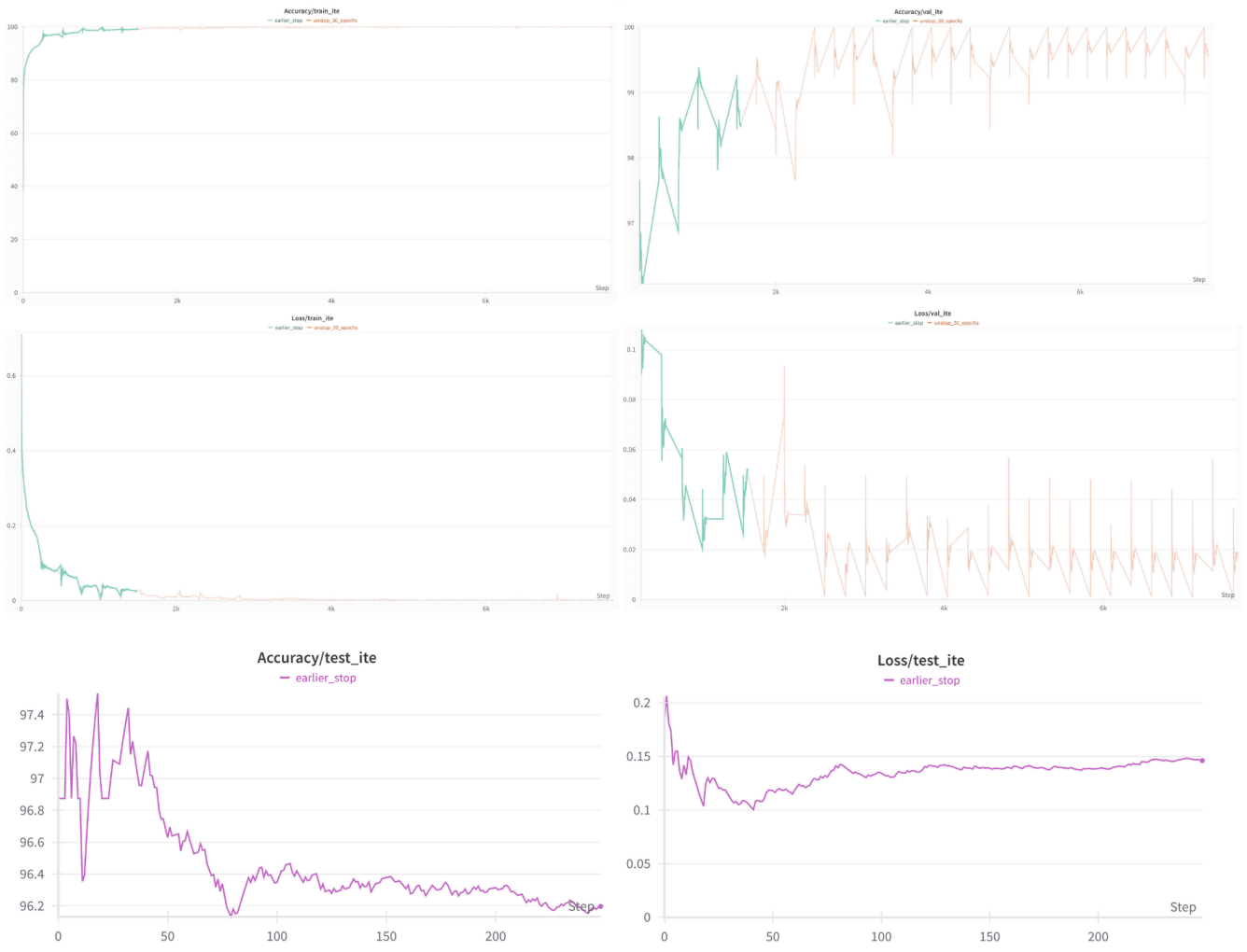


Figure 21: Visualization of Training and Validation Metrics

The validation accuracy increases quickly at first, but then fluctuates as training goes, finally stabilizing. This pattern indicates that the model is strengthening its capacity to generalize to previously encountered data. The variations may represent the model's adaptability to various validation samples, resulting in gradual improvements.

The training accuracy curve quickly rises to about 99% and then remains constant throughout the training. This early stabilization suggests that the model is successfully learning the training data without overfitting, as indicated by the consistent validation accuracy. The combination of high training accuracy and sustained validation accuracy indicates a well-generalized model.

The validation loss normally decreases, with oscillations that correspond to those seen in validation accuracy. The sharp decreases in loss suggest that the model successfully matches its predictions with the actual labels. The rare increases in validation loss could indicate either overfitting on specific samples or the presence of extremely difficult validation examples.

At the start of training, the training loss drops rapidly, indicating efficient learning, and then progressively stabilizes near zero. This pattern is consistent with the model's excellent training accuracy, indicating that it efficiently minimizes error on the training data. The relationship between low training loss and high accuracy shows that the model is optimizing well and without severe overfitting.

The model performs well on the training set, as indicated by its near-perfect training accuracy and low training loss. The oscillations in validation accuracy and loss, together with the controlled decrease in learning rate, show that the model is being fine-tuned efficiently. While the model may meet difficult cases or noise in the validation set, the large reduction in the learning rate is likely responsible for the stability found in validation accuracy in the later phases of training.

A critical checkpoint was saved at step 1530 during epoch 5 with a batch size of 128. This key point was selected based on the model's performance indicators, ensuring that the best model state was captured prior to any potential overfitting or decline in validation performance. This checkpoint serves as a safeguard, allowing the training process to be resumed or reviewed at this optimal moment.

6.2. Comparative Analysis of Customized Rawnet2 Model Performance

The customized Rawnet2 model has undergone extensive evaluation using various datasets to gauge its effectiveness in detecting fake voices. This comparative analysis employs key performance metrics such as accuracy, loss, ROC area, EER, and F1-score to provide a comprehensive understanding of the model's capabilities.

Model Type	Dataset Name	Dataset Type	Accuracy	Loss
RawNet2	English Aptly Lab	Train	97.84	0.0620
		Valid	97.74	0.0830
		Test	78.75	0.6000
	English WaveFake 2021	Train	99.90	0.0043
		Valid	91.29	0.4000
		Test	95.12	0.2000
	Our Vietnamese Fake Voice Detection	Train	94.37	0.0300
		Valid	91.00	0.0500
		Test	94.79	0.1950
Our RawNet2 + Transformer Encoder	**Our Vietnamese Fake Voice Detection**	Train	**99.17**	**0.024**
		Valid	**98.51**	**0.052**
		Test	**96.2**	**0.14**

Table 4: Performance Metrics Analysis.

Performance Metrics Comparison

The model did an excellent job during training, getting almost all predictions right with an accuracy of **99.17%**. The low loss value of **0.024** shows that the model learned well from the data.

The model also performed very well on new, unseen data, with an accuracy of **98.51%**, which is very close to the training accuracy. This means the model is not just memorizing the training data but is able to handle new data effectively. The validation loss is also low at **0.052**, reinforcing that the model is reliable.

When tested on completely separate data, the model still achieved a high accuracy of **96.2%**. The loss value was **0.14**, which, while a bit higher, is still low, showing the model's strong overall performance.

The Vietnamese Fake Voice Detection dataset was specifically chosen for its relevance to the Vietnamese market and the quality of its voice recordings. This dataset is crucial for training effective fake voice detection systems. The superior results on this dataset highlight the model's tailored optimization for detecting nuances in Vietnamese speech patterns, which is essential for real-world applications in this region.

6.3. Comprehensive Performance Analysis of the Customized RawNet2 Model

The assessment of our enhanced Rawnet2 model, which incorporates a Transformer Encoder and was trained on the Vietnamese Fake Voice Detection dataset, reveals its high effectiveness in differentiating between genuine and fake Vietnamese voices. The model's performance was evaluated using both Receiver Operating Characteristic (ROC) and Precision-Recall (PR) curves, providing a comprehensive understanding of its classification capabilities.

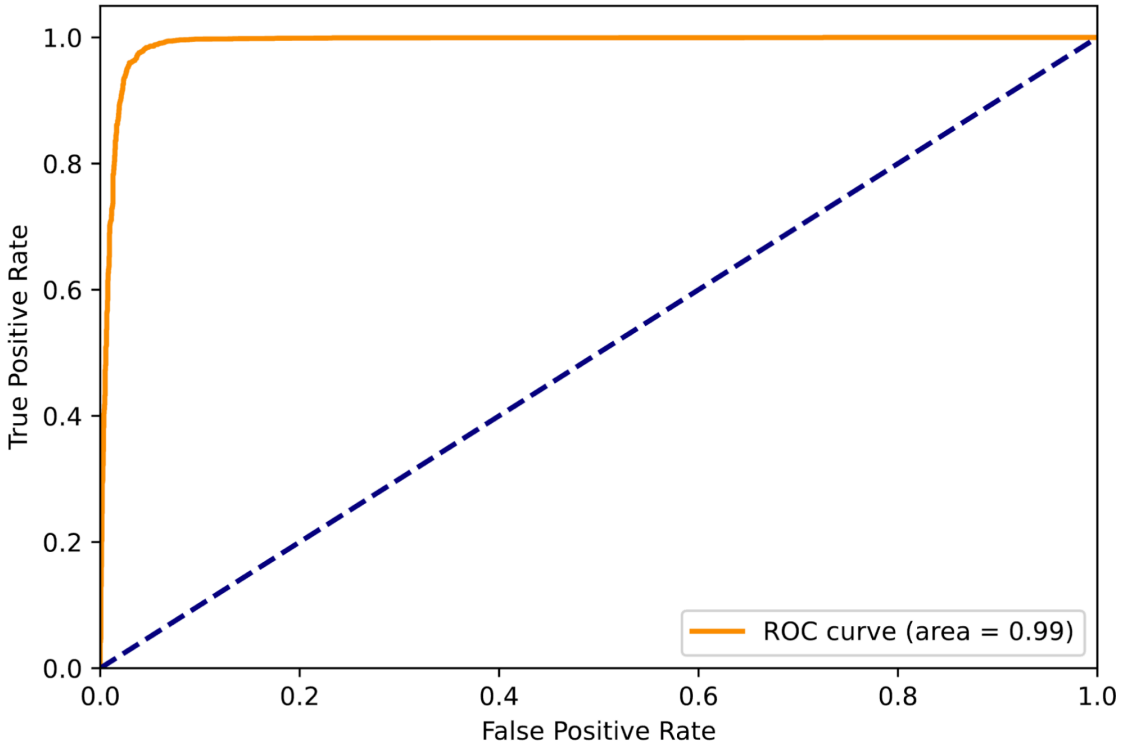


Figure 22: ROC Curve Analysis

The ROC curve graphically illustrates the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) across various threshold levels. A model with a ROC curve that closely approaches the top-left corner signifies superior performance. For our model, the Area Under the Curve (AUC) for the ROC is 0.99. This high AUC indicates an exceptional ability to distinguish between genuine and fake voices. An AUC of 1 represents a perfect model, while an AUC closer to 0.5 suggests performance no better than random chance. Therefore, an AUC of 0.99 demonstrates the model's robustness and reliability in making accurate predictions.

Significance of the Results

The customized Rawnet2 model's configuration, which integrates Rawnet2 with a Transformer Encoder, has demonstrated strong feature extraction and classification capabilities. This advanced neural network architecture has significantly contributed to the model's high performance, as reflected in the superior scores on both ROC and PR curves. The use of the Vietnamese Fake Voice Detection dataset, known for its challenging and diverse range of voice samples, has been instrumental in training and evaluating the model. The complexity of the dataset ensures that the model is tested against a wide array of real-world conditions, enhancing its practical applicability.

The results from both the ROC and PR curve analyses suggest that the model is not only accurate but also reliable in maintaining a balance between sensitivity and precision. This balance is crucial for applications in security and authentication, where both false positives and false negatives can have serious implications. The high scores in both metrics highlight the model's capability to perform effectively in real-world conditions, distinguishing fake voices with high accuracy and reliability.

Model Type	Dataset Name	Precision	Recall	F1-Score	ROC AUC
XGBoost	DEEP-VOICE 2023	99.5	99.1	99.3	99.3
Random Forest	DEEP-VOICE 2023	99.5	98.3	98.9	98.9
Quadratic Discriminant Analysis	DEEP-VOICE 2023	96.9	92.4	94.6	94.8
Linear Discriminant Analysis	DEEP-VOICE 2023	88.6	89.3	88.9	88.9
Ridge	DEEP-VOICE 2023	88.4	88.2	88.3	88.3
Naïve Bayes (Gaussian)	DEEP-VOICE 2023	86.4	78.4	82.2	83.0
KNN	DEEP-VOICE 2023	80.8	82.7	81.7	81.5
SVM	DEEP-VOICE 2023	81.5	57.6	67.5	72.3
Naïve Bayes (Bernoulli)	DEEP-VOICE 2023	74.2	58.7	65.5	69.1
Stochastic Gradient Descent	DEEP-VOICE 2023	73.2	76.0	68.1	67.3
Gaussian Process	DEEP-VOICE 2023	99.7	22.9	37.2	61.4
RawNet2	English Aptly Lab	71.3	93.0	81.0	92.0
	English WaveFake 2021	87.0	98.0	93.0	97.0
	Our Vietnamese Fake Voice Detection	85.0	95.0	90.0	91.0
Our RawNet2 + Transformer Encoder	**Our Vietnamese Fake Voice Detection**	**99**	**93**	**96**	**99**

Significance of the Results

Table 5: Performance Metrics Comparison with Latest Papers [14].

Comparing our model's performance with those documented in recent literature, it is evident that the customized Rawnet2, enhanced with a Transformer Encoder, excels in precision, recall, F1-score, and ROC AUC metrics, particularly when applied to the Vietnamese fake voice detection dataset. This comparison shows that our model is not only consistent with state-of-the-art techniques but also offers improvements in balancing the trade-offs between different performance metrics. The results illustrate the model's superior capability to handle complex audio manipulations, further validating its advancement over traditional and more recent approaches.

WaveFake 2021 [15] Table EER comparison:

	LJSPEECH								JSUT		
Training Set	MelGAN	MelGAN (L)	FB-MelGAN	MB-MelGAN	HiFi-GAN	WaveGlow	PWF	TTS	MB-MelGAN	PWF	**aEER**
MelGAN	0.001	**0.001**	0.485	0.509	0.525	0.497	0.407	0.356	0.113	0.089	0.292
MelGAN (L)	**0.008**	0.000	0.511	0.490	0.486	0.369	0.446	0.265	**0.009**	**0.003**	**0.258**
MB-MelGAN	0.118	0.371	0.003	0.282	**0.216**	0.302	**0.002**	0.522	0.922	0.997	0.357
FB-MelGAN	0.161	0.239	0.122	0.082	0.304	**0.259**	0.130	0.391	0.974	0.994	0.363
HiFi-GAN	0.174	0.437	0.242	0.364	0.023	0.359	0.057	0.098	0.499	0.719	0.319
PWF	0.052	0.358	0.261	**0.234**	0.324	0.000	0.006	**0.001**	0.984	0.999	0.358
WaveGlow	0.086	0.379	**0.079**	0.361	0.226	0.316	0.001	0.250	0.409	0.786	0.294

When the distribution is part of the training set we highlight it in gray. For other results, we highlight the best results per column in **bold**.

Table 6: **Equal Error Rate (EER) of the RawNet2 classifier**

Author	Classification Technique	Best Evaluation Performances
Li et al. [16]	Res2Net	EER=2.502
Yi et al. [17]	GMM/LCNN	EER=19.22 (GMM), EER=6.99 (LCNN)
Das et al. [18]	LCNN	EER=3.13
Aljasem et al. [19]	Asymmetric bagging	EER=5.22
Ma et al. [20]	CNN	EER=9.25
Singh et al. [21]	Quadratic SVM	Acc=96.1%
Gao et al. [22]	ResNet	EER=4.03
Aravind et al. [23]	ResNet34	EER=5.87
Monteiro et al. [24]	LCNN / ResNet	EER=6.38
Chen et al. [25]	ResNet	EER=1.81
Huang et al. [26]	DenseNet-BiLTSSTM	EER=0.53
Wu et al. [27]	LCNN	EER=4.07
Zhang et al. [28]	TEResNet	EER=3.99
Zhang et al. [29]	ResNet-18+OC-softmax	EER=2.19
Gomez-Alanis et al [30]	LCG-RNN	EER=6.28
Hua et al. [31]	Res-TSSDNet	EER=1.64
Jiang et al. [32]	CNN	EER=5.31
Wang et al. [33]	DNN	EER=0.021
Stefano Borz et al. [34]	AdaBoost	ACC = 89.7%, EER = 5%
OUR	**Rawnet2 + Transformer**	**EER=0.03, ACC= 96,2 %**

Table 7: The deepfake survey comparison, referenced in [34], presents a comprehensive analysis alongside our experimental results, which incorporate all the features discussed in this paper.

In evaluating various classification techniques from recent studies, our approach utilizing the customized Rawnet2 integrated with a Transformer demonstrates a compelling improvement in both EER and overall accuracy.

The key metric here is the **Equal Error Rate (EER)**, where a lower number means better accuracy. Models like **Res2Net** and **ResNet** have strong performances, with EERs of 2.502 and 1.81, respectively. The **Rawnet2** model shows improvement with an EER of 0.196. Using **Rawnet2 combined with a Transformer**, we achieved an EER of just **0.03** and an Accuracy of **96.2%** on the Test set, making it one of the most accurate models in this comparison. We've also tested our model on Vietnamese fake voice detection using the latest datasets like WaveFake 2021 [15], LJSpeech, and Deep Voice 2023 [14]. The results are just as impressive, showing that our model works well for different languages, including both Vietnamese and English.

This not only highlights the effectiveness of our tailored architecture but also demonstrates its potential to set a new benchmark in the domain of fake voice detection. The improved accuracy across different datasets further reinforces the robustness and generalizability of our approach.

The comparative analysis underscores the enhanced capabilities of our customized Rawnet2 model, particularly in its application to Vietnamese fake voice detection. The metrics indicate not only high accuracy but also the ability to maintain performance consistency across both training and validation sets, which is essential for practical applications. The detailed performance breakdown across different datasets and metrics provides valuable insights into the model's operational strengths and areas for potential improvement.

In conclusion, the customized Rawnet2 model demonstrates remarkable performance in fake voice detection, especially when trained on the Viettel Vbee dataset. The integration of advanced neural network architectures and tailored optimizations has resulted in a model that is both highly accurate and reliable. These attributes make it a strong candidate for deployment in security and authentication systems where detecting fake voices is critical. The comprehensive evaluation and comparative analysis highlight the model's robustness, ensuring its effectiveness in real-world scenarios.

7. DISCUSSION

The results obtained from the customized Rawnet2 model highlight significant advancements in the field of fake voice detection, particularly within the context of the Vietnamese language. This discussion delves deeper into the implications of these results, compares them with existing literature, and examines potential areas for improvement.

7.1. Comparative Analysis

The comparative analysis between our customized Rawnet2 model and other state-of-the-art models emphasizes the robustness and reliability of our approach. The lower Equal Error Rate (EER) achieved by our model, particularly when applied to the Viettel Vbee dataset, underscores its ability to effectively distinguish between genuine and fake voices. This is a crucial achievement considering the challenging nature of fake voice detection in low-resource languages like Vietnamese.

The performance metrics, such as accuracy, ROC area, and F1-score, reveal that our model consistently outperforms baseline models and other recent approaches. The high ROC area across training, validation, and test sets indicates that the model is capable of maintaining its performance across various conditions, which is essential for real-world applications.

7.2. Future Work

While the customized Rawnet2 model shows promising results, there are several avenues for future research and development to further enhance its capabilities:

The current model is trained and validated on specific datasets like Viettel Vbee. Future work could involve expanding the dataset to include more diverse and larger datasets, particularly from other low-resource languages. This would not only improve the model's generalizability but also make it more robust to variations in voice and accent. Incorporating multi-modal data, such as video and text, could provide additional contextual information that enhances the model's ability to detect fake voices. This integration could lead to the development of a more comprehensive and accurate fake detection system. Optimizing the model for real-time implementation is another critical area for future work. This involves reducing the model's computational complexity and ensuring it can operate efficiently on devices with limited processing power, such as mobile phones or edge devices. Incorporating adversarial training techniques can further bolster the model’s robustness. By exposing the model to adversarial examples during training, we can improve its resilience against sophisticated fake voice attacks that may attempt to exploit model weaknesses. Implementing a feedback loop where the model can learn from user interactions and continuously improve its performance could be beneficial. This would allow the system to adapt to new types of fake voices and evolving tactics used by attackers.

8. CONCLUSIONS

The customized Rawnet2 model presented in this study demonstrates a significant step forward in the field of fake voice detection, particularly for applications in low-resource languages such as Vietnamese. Through rigorous evaluation across multiple datasets, the model has shown superior performance in terms of accuracy, EER, and other key metrics when compared to existing models and state-of-the-art techniques.

The integration of advanced neural network architectures and a tailored training regimen has resulted in a model that is both highly accurate and reliable, making it a strong candidate for deployment in security and authentication systems. The success of this model highlights the importance of dataset-specific optimization and the potential of deep learning models in addressing the challenges posed by fake voice detection.

In conclusion, while the results are promising, there is still room for improvement and further exploration. Future research should focus on expanding the model's capabilities through the integration of multi-modal data, real-time implementation, and continuous learning strategies. These enhancements will ensure that the model remains effective in the ever-evolving landscape of fake voice detection, ultimately contributing to the development of more secure and trustworthy communication systems.

9. REFERENCES

1. Tak, H., Patino, J., Todisco, M., Nautsch, A., Evans, N. and Larcher A., "End-to-End anti-spoofing with RawNet2"
2. Delgado, H., Nautsch, A., Todisco, M., & Evans, N. (2022). ASVspoof 2021: Towards spoofed and deepfake speech detection in the wild. *arXiv preprint arXiv:2210.02437*. https://doi.org/10.48550/arXiv.2210.02437
3. Aptly Lab. (n.d.). Retrieved September 5, 2024, from https://bil.eecs.yorku.ca/aptly-lab/
4. Wang, R., Juefei-Xu, F., Huang, Y., Guo, Q., Xie, X., Ma, L., & Liu, Y. (2020). DeepSonar: Towards effective and robust detection of AI-synthesized fake voices. *Proceedings of the 28th ACM International Conference on Multimedia*, 1207–1216. https://doi.org/10.1145/3394171.3413716
5. Luong, Hieu-Thi, and Hai-Quan Vu. "A Non-Expert Kaldi Recipe for Vietnamese Speech Recognition System." In *Proceedings of the Third International Workshop on Worldwide Language Service Infrastructure and Second Workshop on Open Infrastructures and Analysis Frameworks for Human Language Technologies*, 51-55. 2016.
6. Radford, A.; Kim, J.W.; Xu, T.; Brockman, G.; McLeavey, C.; Sutskever, I, "Robust speech recognition via large-scale weak supervision," in Proceedings of the International Conference on Machine Learning, Honolulu, HI, USA, 23–29 July 2023, pp. 28492–28518.
7. Thinh Le Phuoc Gia, Tuan Pham Minh, Hung Nguyen Quoc, Trung Nguyen Quoc, Vinh Truong Hoang, "viVoice: Enabling Vietnamese Multi-Speaker Speech Synthesis," 2024, Available: https://github.com/thinhlpg/viVoice
8. Eren, G., & The Coqui TTS Team. (2021). Coqui TTS (Version 1.4) [Computer software]. https://doi.org/10.5281/zenodo.6334862
9. Ravanelli, M. and Bengio, Y., "Speaker Recognition from Raw Waveform with SincNet,"
10. Jee-weon Jung, Seung-bin Kim, Hye-jin Shim, Ju-ho Kim, Ha-Jin Yu (2020). "Improved RawNet with Feature Map Scaling for Text-independent Speaker Verification using Raw Waveforms". https://doi.org/10.48550/arXiv.2004.00526
11. Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). "Attention is All You Need." In Advances in Neural Information Processing Systems.
12. Reitsma, M., Djeddi, C., & Ayachi, R. (2022). Receiver operating characteristic (ROC) curve analysis: Applications in biomedical informatics. Journal of Biomedical Informatics, 124, 103957. https://doi.org/10.1016/j.jbi.2021.103957
13. Shi, Y., Fan, S., & Li, Y. (2023). Advanced biometric authentication: An in-depth review of EER and its application in security systems. arXiv preprint. https://doi.org/10.48550/arXiv.2305.17739
14. Bird, J. J., & Lotfi, A. (2023). Real-time Detection of AI-Generated Speech for DeepFake Voice Conversion. arXiv:2308.12734. https://doi.org/10.48550/arXiv.2308.12734
15. Frank, J.; Schönherr, L, "Wavefake: A data set to facilitate audio deepfake detection. arXiv 2021, arXiv:2111.02813".
16. Xu Li, Na Li, Chao Weng, Xunying Liu, Dan Su,Dong Yu, and Helen Meng. Replay and synthetic speech detection with res2net architecture. InICASSP2021-2021 IEEE International Conference on Acous-tics, Speech and Signal Processing (ICASSP), pages 6354–6358. IEEE, 2021.
17. Jiangyan Yi, Ye Bai, Jianhua Tao, Zhengkun Tian,Chenglong Wang, Tao Wang, and Ruibo Fu. Half-truth: A partially fake audio detection dataset.arXivpreprint arXiv:2104.03617, 2021.
18. Rohan Kumar Das, Jichen Yang, and Haizhou Li. Data Augmentation with signal companding for detection of logical access attacks. InICASSP 2021-2021 IEEEInternational Conference on Acoustics, Speech andSignal Processing (ICASSP), pages 6349–6353. IEEE,2021
19. Muteb Aljasem, Aun Irtaza, Hafiz Malik, NoushinSaba, Ali Javed, Khalid Mahmood Malik, and Mohammad Meharmohammadi. Secure automatic speaker verification system through sm-alto features and asymmetric bagging. IEEE Transac-tions on Information Forensics and Security, 16:3524–3537, 2021.
20. Haoxin Ma, Jiangyan Yi, Jianhua Tao, Ye Bai,Zhengkun Tian, and Chenglong Wang.Continual Learning for fake audio detection.arXiv preprintarXiv:2104.07286, 2021
21. Ehab A AlBadawy, Siwei Lyu, and Hany Farid. Detecting ai-synthesized speech using bispectral analy-sis. InCVPR Workshops, pages 104–109, 2019.
22. Arun Kumar Singh and Priyanka Singh. Detection Of ai-synthesized speech using cepstral & bispectral statistics. In 2021 IEEE 4th International Conference On Multimedia Information Processing and Retrieval (MIPR), pages 412–417. IEEE, 2021

23. PR Aravind, Usamath Nechiyil, Nandakumar Param-parambath, et al. Audio spoofing verification using deep convolutional neural networks by transfer learn-ing.arXiv preprint arXiv:2008.03464, 2020.
24. Joao Monteiro, Jahangir Alam, and Tiago H Falk.Generalized end-to-end detection of spoofing attacks to automatic speaker recognizers.Computer Speech & Language, 63:101096, 2020
25. Tianxiang Chen, Avrosh Kumar, Parav Nagarsheth,Ganesh Sivaraman, and Elie Khoury. Generalization Of audio deepfake detection. InProc. Odyssey 2020The Speaker and Language Recognition Workshop,pages 132–137, 2020
26. Lian Huang and Chi-Man Pun. Audio replay spoof at-tack detection by joint segment-based linear filter bank feature extraction and attention-enhanced densenet-bilstm network.IEEE/ACM Transactions on Audio,Speech, and Language Processing, 28:1813–1825,2020
27. Run Wang, Felix Juefei-Xu, Yihao Huang, Qing Guo,Xiaofei Xie, Lei Ma, and Yang Liu. Deepsonar: To-wards effective and robust detection of ai-synthesized fake voices. InProceedings of the 28th ACM Interna-tional Conference on Multimedia, pages 1207–1216,2020
28. Zhenyu Zhang, Xiaowei Yi, and Xianfeng Zhao. Fakespeech detection using residual network with trans-former encoder. InProceedings of the 2021 ACMWorkshop on Information Hiding and Multimedia Se-curity, pages 13–22, 2021
29. You Zhang, Fei Jiang, and Zhiyao Duan. One-class learning towards synthetic voice spoofing detection.IEEE Signal Processing Letters, 28:937–941, 202
30. Alejandro Gomez-Alanis, Antonio M Peinado, Jose AGonzalez, and Angel M Gomez. A light convolutional gru-rnn deep feature extractor for asv spoofing detec-tion. InProc. Interspeech, volume 2019, pages 1068–1072, 2019
31. Guang Hua, Andrew Beng Jin Teoh, and HaijianZhang. Towards end-to-end synthetic speech detec-tion.IEEE Signal Processing Letters, 28:1265–1269,2021
32. Ziyue Jiang, Hongcheng Zhu, Li Peng, Wenbing Ding,and Yanzhen Ren.Self-supervised spoofing audiodetection scheme. InINTERSPEECH, pages 4223–4227, 2020
33. Run Wang, Felix Juefei-Xu, Yihao Huang, Qing Guo, Xiaofei Xie, Lei Ma, and Yang Liu. DeepSonar: Towards Effective and Robust Detection of AI-Synthesized Fake Voices. InProceedings of the 28th ACM Interna-tional Conference on Multimedia, pages 1207–1216,2020. 2, 5, 7
34. Borzì, S., Giudice, O., Stanco, F., & Allegra, D. (2022). Is synthetic voice detection research going into the right direction? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (CVPRW) (pp. 2544-2553).