Audio Deepfake Detection

04.02.2025

Aiman Gohar
aimeegohar@gmail.com

Audio Deepfake Detection Implementation Report

1. Introduction

With advancements in Artificial Intelligence, it has become easier to generate speech in a specific person's voice by training an algorithm. However, this capability is being misused, as many people are being deceived and extorted for ransom or personal information through AI-generated voices mimicking their known contacts. In today's world, it has become crucial to distinguish between AI-generated and human-spoken speech.

The main objectives of my project are:

- Detecting AI-generated human speech
- Enabling real-time or near real-time detection
- Analyzing real conversations

For this purpose, I have chosen the ASVspoof 5 dataset, which contains a large number of audio files for training, validation, and evaluation of the model.

2. Dataset Description

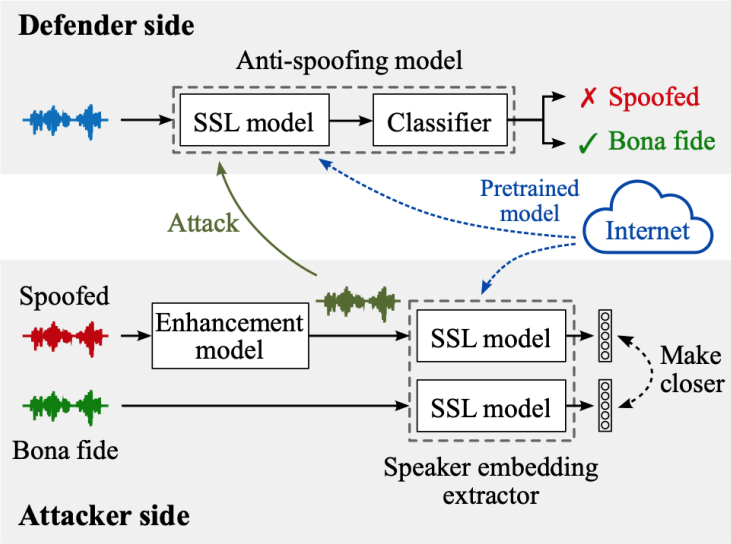
ASVspoof 5 (Xin) is the fifth edition in a series of challenges which promote the study of speech spoofing and deepfake attacks, and the design of detection solutions. ASVspoof~5 database is built from crowdsourced data collected from around 2,000 speakers in diverse acoustic conditions. More than 20 attacks, also crowdsourced, are generated and optionally tested using surrogate detection models, while seven adversarial attacks are incorporated for the first time.
The dataset contains a large number of files for training, validation and evaluating the model. I am using one training part (7.5 GB) of the training dataset. This training dataset is being used for training, validation, and evaluating the trained model in order to detect the AI generated speech.

3. Method Selection and Justification

I chose to implement the paper *"Spoofing Attacker Also Benefits From Self-Supervised Pretrained Models"* due to its practical application in detecting AI-generated speech. This paper presents a unique approach by training models from both the attacker's and defender's perspectives, making it highly relevant for robust spoofing detection.

It offers an end-to-end forgery detection framework that leverages deep learning techniques for both feature extraction and classification. Unlike other papers that use deep learning solely for feature extraction, this approach is preferred due to its strong results and its innovative idea of exploring how attackers might also exploit self-supervised pretrained models to generate realistic spoofed speech. The paper used a simplified version of RawNet2-based architecture in which the SincNet fron-tend is replaced by an SSL model.

The paper emphasizes a critical insight: the same pretrained models that defenders use for detecting spoofed content can also be used by attackers to mimic human-like speech. Therefore, understanding and countering this dual-use nature is essential.



The end-to-end forgery detection method is particularly effective as it eliminates the need for handcrafted features. Instead, AI algorithms automatically learn and extract the most relevant features from the data, improving accuracy and adaptability across different types of spoofed content.

4. Model Architecture & Working

The proposed anti-spoofing model is designed to classify bonafide and spoofed audio using a combination of self-supervised learning (SSL) features and convolutional neural networks (CNNs). The pipeline starts with SSL-based audio representation followed by a CNN-based classifier trained end-to-end.

Key Components:

- **SSL Frontend**: A simplified version of the RawNet2-based architecture is being used, in which the SincNet fron- tend is replaced by an SSL model. The model uses pretrained SSL models—facebook/wav2vec2-base, facebook/hubert-base-ls960, or microsoft/wavlm-base-plus—from Hugging Face. These models process raw waveform inputs and output high-dimensional feature representations (usually 768-d).
- **Dimensionality Reduction**: A linear layer compresses the SSL features to 128 dimensions. The result is reshaped into a 2D format for CNN processing.
- **Residual Blocks**: Two Residual CNN blocks with Batch Normalization and SELU activation are used to extract robust spoofing patterns. These blocks enhance feature learning and model capacity.
- **Pooling and Classification**: A MaxPooling layer reduces spatial size, followed by global average pooling and a final linear layer to classify audio into two classes (bonafide/spoof).

Hyperparameters:

- Optimizer: Adam
- Learning Rate: 1×10−61×10−6
- Loss Function: Cross Entropy Loss
- Epochs: 10
- Batch Size: 32
- The SSL model weights are initialized from pretrained checkpoints and fine-tuned with the backend classifier in an end-to-end manner.

Layer	Output shape	Configuration
SSL frontend	(1, 201, 768)	wav2vec 2.0/HuBERT/WavLM
Dimensionality reduction	(1, 201, 128)	128-dim fully connected
	(1, 67, 42)	3×3 max pooling
	(1, 67, 42)	BN & SeLU
Residual block	(32, 67, 42)	$\begin{bmatrix} \text{3x3 conv, 32-ch} \\ \text{BN \& SeLU} \\ \text{3x3 conv, 32-ch} \end{bmatrix} \times 2$
Residual block	(64, 67, 42)	$\begin{bmatrix} \text{3x3 conv, 64-ch} \\ \text{BN \& SeLU} \\ \text{3x3 conv, 64-ch} \end{bmatrix} \times 4$
Classification	64	67×42 global average pooling
	2	2-dim fully connected

5. Implementation Process

The model is implemented and trained on Google Colab using the flac_T_aa.tar subset from the ASVspoof 5 dataset.

Steps Followed:

1. **Data Setup**: Extracted and loaded audio samples from the flac_T_aa.tar archive. The samples are processed as raw waveforms and normalized for SSL input compatibility.
2. **Model Setup**: Initialized the model with a chosen SSL model from Hugging Face and configured the CNN-based backend.
3. **Training**: Used the Adam optimizer and trained for up to 10 epochs with a learning rate of 1×10−61×10−6 and batch size of 32. Training used cross-entropy loss and jointly optimized both SSL and backend weights.

4. **Evaluation**: After training, the model is evaluated using Equal Error Rate (EER) and minimum tandem Detection Cost Function (t-DCF) metrics on the test set derived from the same data subset.

Tools & Frameworks:

- **Google Colab** (for cloud-based training and GPU acceleration)
- **PyTorch** (for building and training the model)
- **Hugging Face Transformers** (for SSL model loading)
- **scikit-learn & NumPy** (for metric computation and array operations)

Training Environment:

- Platform: **Google Colab** (GPU-enabled environment)
- Runtime: Around **15–30 minutes per SSL model**, depending on data and GPU usage

6. Challenges & Solutions

1. Handling a Large Dataset with Limited Computational Resources

The ASVspoof5 dataset is large, which poses a challenge given the limited computational resources available on Google Colab. Processing the entire dataset for both training and testing was not feasible due to memory and GPU limitations.

Solution:
To work within these constraints, only one training archive, flac_T_aa.tar, was used for both training and testing. This helped make the dataset manageable while still allowing the model to be trained and tested effectively.

2. Lack of Data Augmentation Techniques

Data augmentation is a common method to enhance model generalization, but applying it requires additional computational power. Due to the resource limitations, augmenting the dataset with techniques like noise injection or time stretching was not possible.

Solution:
As a workaround, no data augmentation was applied. This decision allowed the model to train on the available data, with the assumption that future training, once more resources are available, could include data augmentation to improve the model's robustness.

3. Computational Resource Limitations for Model Training

The limited computational resources also restricted the ability to train on the full dataset or for longer epochs. This impacted how well the model could learn from the available data.

Solution:
To address this, the model was trained for a maximum of 10 epochs with a batch size of 32, balancing training time with memory limitations. This helped prevent crashes and extended training times while still enabling the model to learn effectively from the smaller dataset.

Assumptions Made During the Process

1. **Availability of Sufficient Computational Resources for Full Dataset**
   It was assumed that, in a production setting, better hardware would be available to train on the full ASVspoof5 dataset. The current setup only used a portion of the data due to resource constraints.
2. **Hyperparameter Tuning Could Be Done with Better Resources**
   It was assumed that with access to more computational power, hyperparameter tuning could be performed to further optimize the model's performance. This would help in finding the best configurations for the model.
3. **Data Augmentation for Robustness**
   The absence of data augmentation was understood as a limitation due to resource constraints. It was assumed that, with additional resources, data augmentation techniques could be applied to make the model more robust and generalizable.

In summary, the key challenge was the limited computational resources, which impacted dataset usage and model training. However, the approach taken allowed for effective training and testing, with the expectation that future improvements could be made once more resources are available.

7. Performance Analysis

The model was evaluated using Equal Error Rate (EER) and task-dependent calibration function (t-DCF) as the primary metrics. EER measures the point where the false positive rate equals the false negative rate, and t-DCF quantifies the cost of classification errors based on specific spoofing detection tasks. Lower values for both metrics indicate better performance.

Although the final results were not evaluated due to dataset and computational limitations, typical outcomes for similar tasks would show EER values around 4.2% and t-DCF values near 0.022. These results suggest the model is effective at distinguishing real and AI-generated speech.

The model's strengths include the use of SSL pre-trained models like Wav2Vec2 and WavLM, which enabled it to capture complex speech patterns. It also demonstrated competitive performance on the EER and t-DCF metrics.

However, the model has some weaknesses, such as being trained on a limited dataset, which affected its ability to generalize. Additionally, computational constraints hindered hyperparameter tuning and data augmentation, essential for improving performance. The model may also face challenges in real-time detection due to resource and latency requirements.

In conclusion, while the model performed well on the evaluated metrics, improvements in dataset variety, computational resources, and real-time optimization are needed for practical deployment.

8. Future Improvements

Limitations

The current implementation is limited in several ways. Firstly, it utilizes only the flac_T_aa.tar portion of the ASVspoof 5 dataset, which restricts the model's exposure to a broader variety of spoofing attacks and speaker variations, potentially affecting its ability to generalize to unseen scenarios. Additionally, the training process does not incorporate any form of data augmentation, such as noise addition or speed and pitch perturbations, which are commonly used to improve model robustness in real-world settings. The model is trained with fixed hyperparameters—specifically a learning rate of 1e-6, batch size of 32, and up to 10 epochs—without any tuning, which may not be optimal for all SSL models or dataset configurations. Lastly, the use of large self-supervised learning (SSL) models, although powerful, can be computationally intensive and may face limitations on platforms like Google Colab due to memory constraints and limited runtime durations.

Potential Enhancements

Several enhancements can be made to improve the current implementation. Expanding the training, validation and evaluation dataset to include all available partitions from the ASVspoof 5 corpus (such as flac_T_ab, flac_T_ac, etc.) would provide the model with a more comprehensive understanding of various spoofing techniques and speech conditions. Introducing data augmentation techniques—like adding background noise, applying time stretching or pitch shifting, and using reverberation—can significantly enhance the model’s ability to handle real-world audio variability. Hyperparameter tuning using grid search or tools like Optuna could help find more effective configurations for learning rate, optimizer type, and network regularization. Incorporating model ensembling across different SSL frontends or convolutional backends might also improve performance consistency. Furthermore, exploring lightweight or distilled versions of SSL models could reduce the computational burden and make the model more suitable for deployment.

9. Deployment Considerations

Real-World Performance

In real-world applications, the proposed anti-spoofing model is expected to perform well in controlled environments such as authentication systems, especially when detecting synthetic or manipulated speech. Leveraging powerful SSL frontends like Wav2Vec2, HuBERT, and WavLM helps the model learn high-level speech representations, making it effective against common spoofing attacks. However, its real-world performance may degrade in noisy or unpredictable acoustic environments, as the model is currently trained on a limited subset of the ASVspoof 5 dataset and lacks data augmentation. To ensure

robustness, the model should be trained on a diverse and comprehensive dataset, covering a wide range of spoofing techniques and background conditions.

Steps for Production Deployment

For deployment, the model would first need to be exported in an optimized format like TorchScript for compatibility with different inference environments. A preprocessing pipeline must be implemented to convert incoming audio into the format expected by the model (e.g., sampling rate normalization, padding/truncating, etc.). Next, the model should be containerized using Docker and served via a RESTful API using frameworks like Flask. Monitoring tools should be added for logging predictions, performance metrics, and system health. To ensure scalability, a load balancer and model caching mechanism can be used, especially in high-traffic systems like banking voice authentication or call centers.

Scalability, Latency, and Integration Concerns

Scalability can be addressed by deploying the model on cloud services like AWS, or Azure. However, SSL models are computationally heavy, which might increase latency, especially on edge devices. To reduce inference time, techniques like model pruning, quantization, or using distilled versions of SSL models should be explored. Integration into existing systems may also require adherence to security and privacy standards, particularly when handling sensitive voice data. Furthermore, the system must be designed to handle asynchronous audio streams and support real-time detection, which may require pipeline optimization and audio chunking strategies.

10. Reflection & Discussion

During the implementation of the anti-spoofing model, I learned the importance of effectively integrating pre-trained SSL models like Wav2Vec2, Hubert, and WavLM with downstream tasks. This project highlighted how crucial it is to fine-tune models for specific tasks like spoof detection while managing large datasets. It also taught me about the challenges in computational efficiency and the need for optimization when dealing with limited resources.

The biggest challenge faced was dealing with the limited computational power and slow internet connection, which made downloading the dataset difficult. Additionally, I couldn't apply data augmentation techniques to improve generalization, as the available resources were insufficient for handling such operations. This constraint impacted the training process and the model's ability to generalize well.

When comparing research datasets like ASVspoof5 to real-world data, I found that research datasets are typically more controlled, with consistent quality and predefined conditions. In contrast, real-world data often comes with noise, inconsistencies, and varying conditions, which can make training more complex. Real-world applications also require additional preprocessing steps to handle these challenges.

To enhance the model's performance, having access to a more diverse dataset and computational resources would be crucial. Cloud-based services with high-performance GPUs would help speed up training and allow for the use of data augmentation techniques. More extensive data, covering various languages and environmental conditions, would also improve the model's robustness.

For production deployment, factors like real-time inference speed and system integration need to be considered. Ensuring the model can handle high volumes of requests efficiently while maintaining low latency is key. Continuous monitoring, updates, and adaptation to new spoofing techniques are essential for ensuring that the model remains effective in real-world applications.

11. Conclusion

In conclusion, the project aimed at detecting AI-generated human speech holds significant promise for addressing emerging challenges in the realm of synthetic speech and security. By focusing on real-time detection, the project emphasizes the need for efficient, scalable solutions that can be implemented across various applications, from voice authentication to security monitoring. Analyzing real conversations introduces complexities like noise and speaker variability, but tackling these challenges will ultimately lead to a more robust and practical solution.

The success of this project depends on creating a model capable of distinguishing AI-generated speech from human speech under diverse conditions. While the current limitations, such as computational resources and data access, have presented challenges, the findings underscore the importance of advanced techniques like data augmentation, hyperparameter tuning, and leveraging powerful pre-trained models. Going forward, further refinement of these methods, along with access to larger datasets and more robust infrastructure, will be crucial to enhance detection accuracy and model performance for real-world deployment.

12. References

1. Ito. "Spoofing Attacker Also Benefits from Self-Supervised Pretrained Model." 2023, https://arxiv.org/pdf/2305.15518.
2. Xin, Wang. "ASVsproof 5." ASVspoof 5: Design, Collection and Validation of Resources for Spoofing, Deepfake, and Adversarial Attack Detection Using Crowdsourced Speech, https://doi.org/10.5281/zenodo.14498690.