

RuHuman: A Resilient Multimodal AI-Powered Audio Verification System

Timothy Do

Friday, December 15th, 2023, 7:00 PM PST

ECE202A Embedded Systems @ UCLA



Motivation and Objectives

- Audio deep fakes are becoming more easily spoofable with the advent of generative AI.
 - Growing number in deep fake crimes
- RuHuman evaluates existing audio liveness detection systems that exploit multiple audio modalities.
- The Objectives of this RuHuman Are:
 - Evaluating different audio liveness detector architectures against state of the art voice cloners.
 - Developing a user interface to make gateway devices (e.g. phone) easily verify with their microphones



Previous/Related Work

- ASVSpooF is a dedicated challenge that to develop audio spoofing classifiers. Three main challenges are PA, LA, and DF (RuHuman focuses on LA).
- Researchers take advantage of various encodings (STFT, MFCC, CQCC) to augment the training resolution of the audio sample.
- UCLA NESL developed ResNet models with different embeddings and performed model fusion to minimize ASV/CM loss metrics in ASVSpooF 2019.

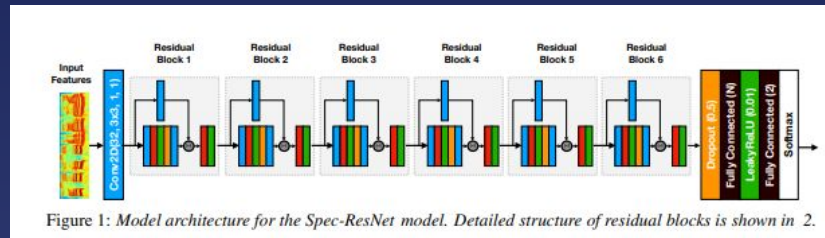


Figure 1: Model architecture for the Spec-ResNet model. Detailed structure of residual blocks is shown in 2.

Novelty of Project

- Datasets used for training in ASVSpooof2019 LA challenge were in a controlled environment, insusceptible to 'out in the wild' samples
- RuHuman investigates the performance of different detector architectures when additive noise is mixed in with the suspected audio sample.
- Additionally, RuHuman implements an interactive user interface that allow for a wide range of computing (from a cell phone to a computer workstation) to access multiple detector models with just an HTTP Post Request.

RuHuman Audio Verification System

Check if Your Audio is Produced by an AI:

Option 1: Record a Live Sample

Record

Option 2: Upload a Recording

Choose File No file chosen

Verify Audio Reset

Choose a Detector Model: Tortoise



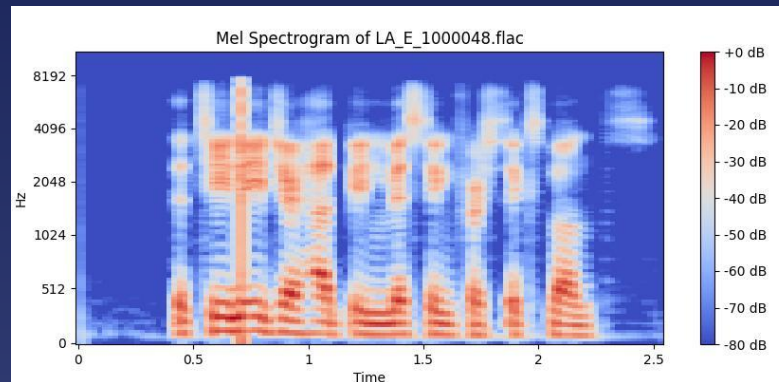
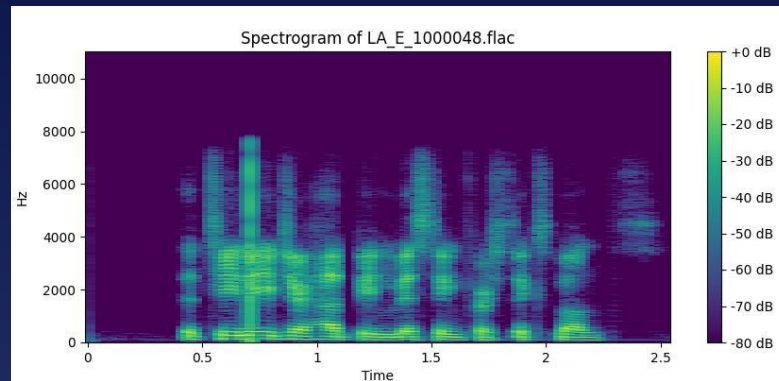
Background: Audio Feature Extractions

STFT (Short-Time Fourier Transform): Compute FFT over a window, use log to plot Spectrogram.

LFCC (Linear-Frequency Cepstral Coefficients): Map frequency powers through a linear filter bank then take DCT of log of filter coefficients

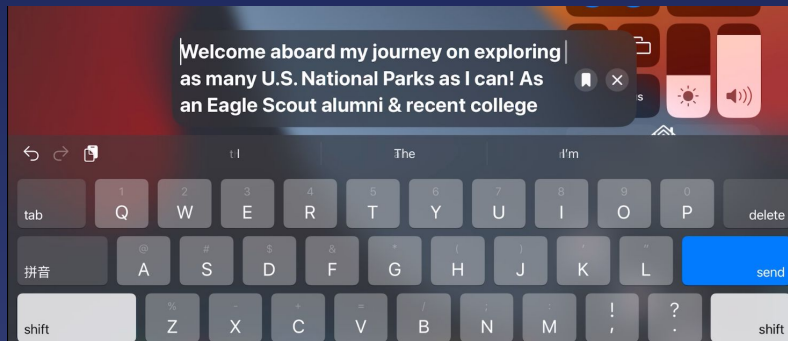
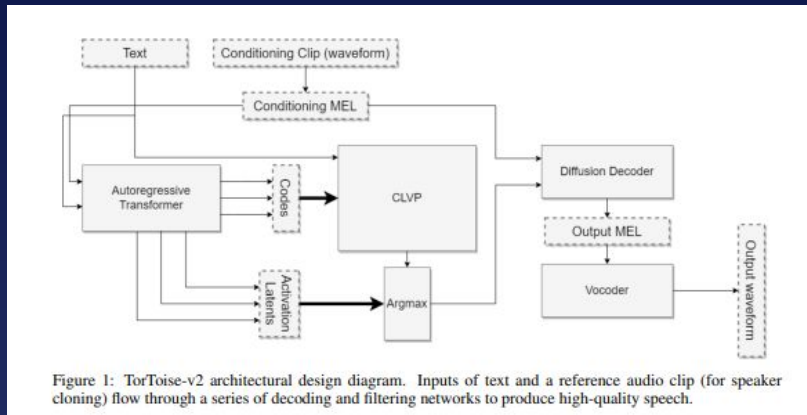
MFCC (Mel-Frequency Cepstral Coefficients): Map frequency powers through a mel filter bank (tuned for humans), then take DCT of coefficients.

CQCC (Constant Q Cepstral Coefficients): Uses CQT basis for cepstrum. More spectro/temporal resolution at low/high frequencies.



Overview of Voice Cloners

- **Tortoise:** A voice cloner using Autoregressive Diffusive decoder with DDPM noise schedulers.
 - Pre-trained on 49,000 hours of audio clips from podcasts/videos.
- **Apple Personal Voice:** A proprietary personalized voice synthesizer that is trained by using 15 mins of live utterance
 - Trained locally on iOS device, use text to speech to synthesize audio.
 - Aside: Not easily exportable, must use screen recorder to grab audio.



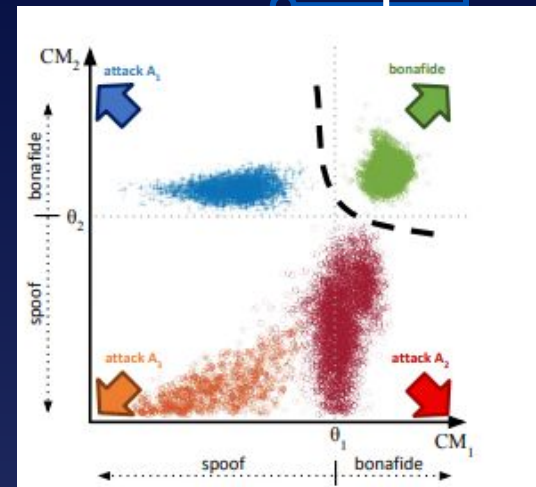
Overview of Voice Detectors

Baseline ASVSpooof Models:

- **LFCC-GMM:** A Two-Class Gaussian Mixture Model with LFCC input. Outputs LLR between bonafide & spoof.
- **CQCC-GMM:** A Two-Class Gaussian Mixture Model with CQCC input. Outputs LLR between bonafide & spoof.
- **RawNet2:** An End to End Neural Network using mel sinc filters. Outputs two-class likelihood between bonafide and spoof.
- All Baseline models trained with ASVSpooof2019 Training Set (~25380 Samples, 20/107 Distinct Speakers from total set)

In the Wild Models:

- **Tortoise:** An Audio Mini Encoder Classifier. Outputs probability on whether an audio sample was generated by TortoiseTTS.
 - Trained on same data used by detector (49,000 hours of podcasts/videos from internet).



Layer	Input: 64000 samples	Output shape
Fixed Sinc filters	Conv(129, 1, 128)	(21290, 128)
	Maxpooling(3)	
	BN & LeakyReLU	
Res block	BN & LeakyReLU	× 2 (2365, 128)
	Conv(3, 1, 128)	
	BN & LeakyReLU	
	Conv(3, 1, 128)	
	Maxpooling(3)	
Res block	FMS	× 4 (29, 512)
	BN & LeakyReLU	
	Conv(3, 1, 512)	
	BN & LeakyReLU	
	Conv(3, 1, 512)	
GRU	GRU(1024)	(1024)
	FC	(1024)
Output	1024	2

Overview of User Interface

- Web Server Implemented in Python Flask to serve many gateway devices that have internet access.
- Users can verify their audio sample in 2 ways:
 - **1.** Users record a live sample client-side by using `getUserMedia()` from Javascript, which then is saved to a blob.
 - **2.** Users upload a pre-recorded audio file from their device to the HTML form to post.
- Users upload data via HTTP request post to server.
- File is stored and pass through detector via refactored callable Python functions (GMMs: `matlab.engine`)
- Dashboard displays uploaded audio (verifying it's on the server), in addition to visualizing it's encodings with *librosa* and outputting metrics of verification.

RuHuman Audio Verification System

Check if Your Audio is Produced by an AI:

Option 1: Record a Live Sample

Record

Option 2: Upload a Recording

Choose File No file chosen

Verify Audio Reset

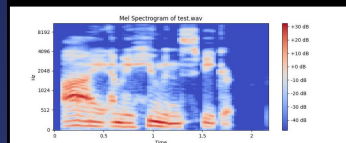
Choose a Detector Model: Tortoise

Your Uploaded Audio Sample: (test.wav)

0:00:00



Features:



Detector Model: Tortoise (Detected in 0.16 seconds)

Probability of Being Spoofed by Generative AI via Tortoise Model: 99.98%

[Return to Homepage](#)

Metrics of Evaluation

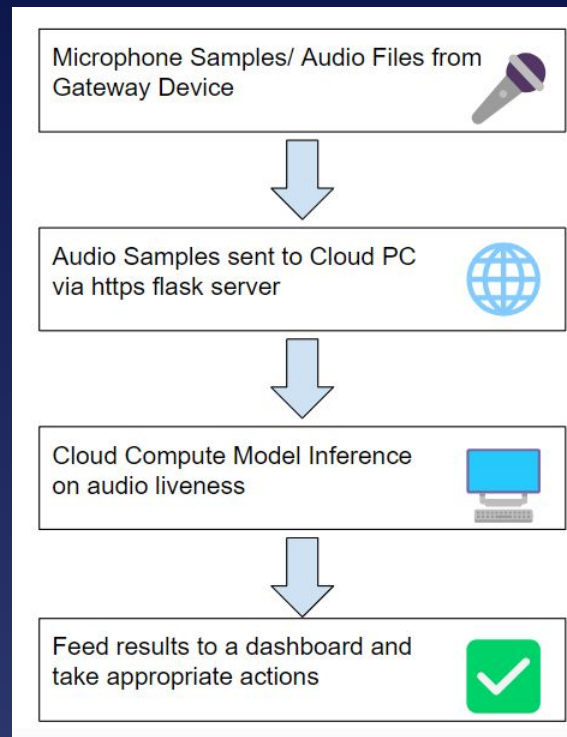
- **t-DCF:** A custom cost function that weighs between predefined ASV scores (from organizers), speaker priors, and CM scores provided by the detector architecture.
- **EER:** The likelihood threshold at which the miss rate and the false alarm rate equate.
- **Computation Time:** The duration for a single audio sample to be verified.
 - Ideally in a live setting should be less than duration of the sample.
- Objective is to minimize all metrics for accuracy and efficiency in live settings.

$$\begin{aligned} \text{t-DCF}(s, t) = & C_{\text{miss}}^{\text{asv}} \cdot \pi_{\text{tar}} \cdot P_a(s, t) \\ & + C_{\text{fa}}^{\text{asv}} \cdot \pi_{\text{non}} \cdot P_b(s, t) \\ & + C_{\text{fa}}^{\text{cm}} \cdot \pi_{\text{spooof}} \cdot P_c(s, t) \\ & + C_{\text{miss}}^{\text{cm}} \cdot \pi_{\text{tar}} \cdot P_d(s). \end{aligned}$$

- $C_{\text{miss}}^{\text{asv}}$ – cost of ASV system rejecting a target trial.
- $C_{\text{fa}}^{\text{asv}}$ – cost of ASV system accepting a nontarget trial.
- $C_{\text{miss}}^{\text{cm}}$ – cost of CM rejecting a human trial.
- $C_{\text{fa}}^{\text{cm}}$ – cost of CM accepting a spoof trial.

Technical Approach

- **Phase 1: Investigation of Additive Noise Sources** (ASVSpooof Submissions Only)
 - Work with each baseline detector to produce metrics on the ASVSpooof2021 Evaluation Set (181,566 samples, 67 Distinct Speakers, 37 F, 30 M).
 - Inject each audio sample in the evaluation set with additive white gaussian noise (SNR = 50).
 - Recompute metrics with infected evaluation set and compare scores.
- **Phase 2: Deployment of User Interface**
 - Send/receive audio from client/gateway with multiple upload options.
 - Refactor detectors to be callable functions with audio path argument.
 - Show audio features and results on dashboard.



Primary Metric Metric: t-DCF & EERs

Detector Architecture (Normal)	Pooled t-DCF	Pooled EER (%)
Baseline LFCC-GMM (MATLAB)	0.5758	19.30
Baseline CQCC-GMM (MATLAB)	0.4964	15.62
Baseline RawNet2 (Python)	0.4255	9.49

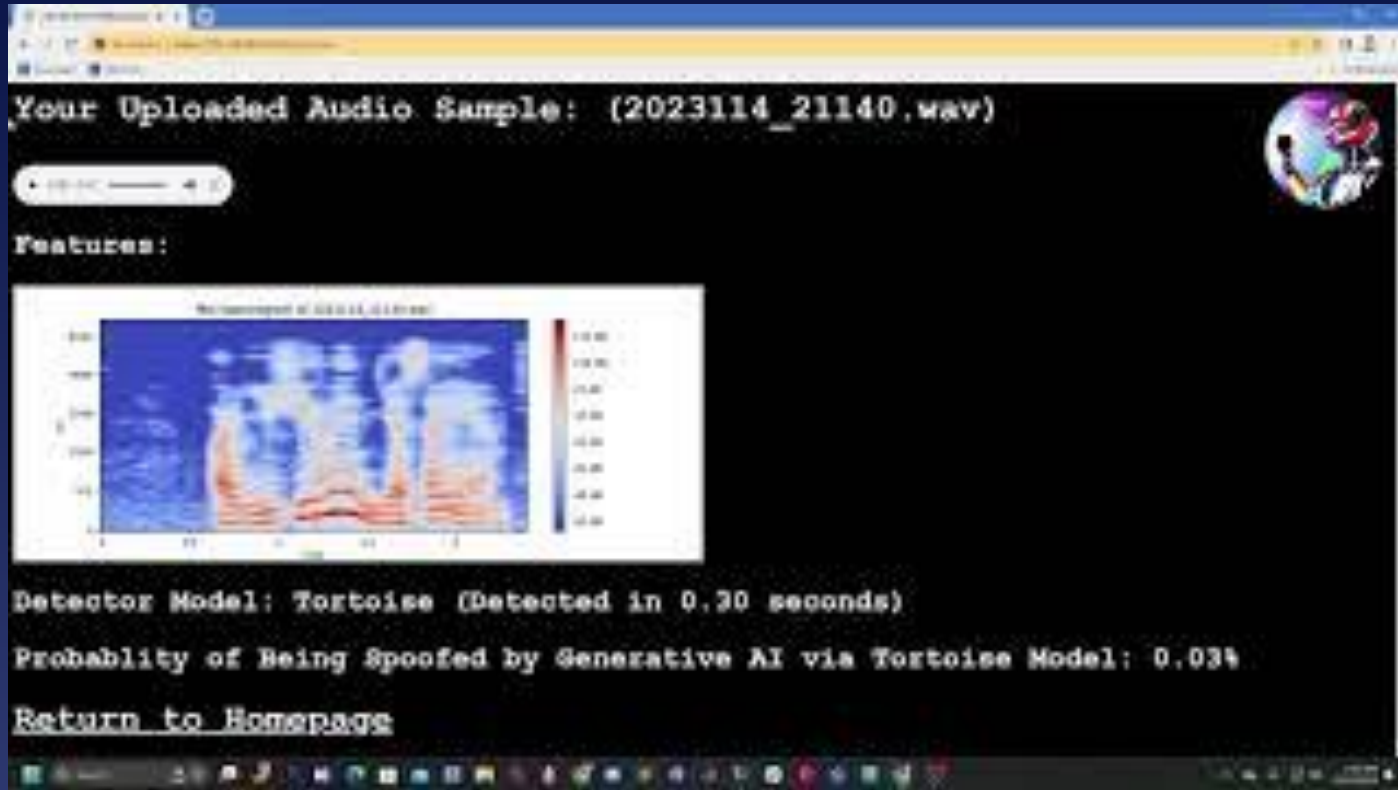
Detector Architecture (AWGN)	Pooled t-DCF	Pooled EER (%)
Baseline LFCC-GMM (MATLAB)	0.8527	41.33
Baseline CQCC-GMM (MATLAB)	0.7609	31.99
Baseline RawNet2 (Python)	0.3317	7.23

Secondary Metric: Computation Time

Detector Architecture	Average Computation Time per Sample (s)
Tortoise Audio Mini Encoder (Python)	0.22
Baseline LFCC-GMM (MATLAB)	0.18
Baseline CQCC-GMM (MATLAB)	0.27
Baseline RawNet2 (Python)	0.30

**on average each sample was approximately 3-5 seconds long*

Demo



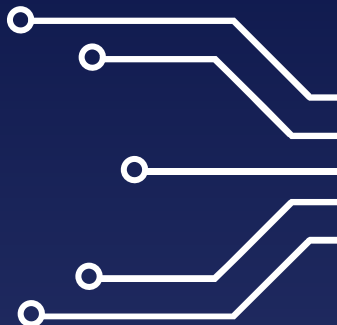

Conclusion & Future Directions

Conclusion:

- Primary evaluation from phase 1 indicates that additive noise can be injected to enhance a plausible spoofing attack to being successful.
- All detectors were able to detect an audio samples within a second, making them usable for liveness detection.
- User Interface was deployed and easily accessible from gateway devices.

Future Directions:

- Preprocess audio samples with denoising to isolate speaker from noise before being sent to spoof detection model.
- Audio spoofing detection can be fused with other modalities of liveness detection (i.e. video deep fake detection) for a complete system. May experiment with this pipeline over winter break on a separate branch.



Thanks for Listening!
Any Questions?

