Part 1: Research & Selection

After examining the Audio Deepfake Detection GitHub repository we found three approaches that seem suitable to detecting AI generated human speech, can be feasibly implemented and allow us to explore interactions between humans in real conversations.

The first and most significant approach is **RawNet2**, presented at **ICASSP** in 2020. RawNet2 is an end-to-end deep neural network as it processes the raw audio waveforms directly, utilizing residual convolutional blocks followed by a Gated Recurrent Unit (GRU). The end-to-end recognition architecture in RawNet2 constitutes features produced by time or frequency, evading trusted handcrafted features, and creates better generalization and adaptation to input audio. RawNet2 had a very low Equal Error Rate (EER) on the ASVspoof 2019 dataset, with 1.08%, thus indicating very high classification performance. The GRU is a useful component because it can account for long term dependencies in the speech data, i.e., understand conversations in an end to end manner. The main tradeoff with this approach is it takes longer to train creating a considerable amount of GPU resources, and the input audio waveforms can be impacted by non-semantic noise in real environments.

The second method is an established pipeline that employs either **LFCC (Linear Frequency Cepstral Coefficients) or CQCC (Constant-Q Cepstral Coefficients)** alongside a **Gaussian Mixture Model (GMM)**. These established machine learning methods utilize feature extraction from the spectral domain, with statistical modeling for spoofing detection. The LFCC-GMM baseline achieves an EER of about 17.55%. Although accuracy is inferior when compared to deep learning models, the approach produces an extremely lightweight model that can be applied in real-time and/or embedded systems with limited resources. Overall, its disadvantage is less generalizability to previously unseen spoofing methods, which impacts its robustness in more dynamic environments when compared to deep models.

The third and most recent method is **AASIST (Audio Anti-Spoofing using Integrated Spectro-Temporal features)**, which was introduced at **INTERSPEECH** in 2022. AASIST integrates a multi-stream architecture that incorporates 2D convolutional layers (Res2Net), along with a 1D temporal attention mechanism and a combination of the spectral features. AASIST aims to capture both short-term and long terms features present in human speech conversation. AASIST achieved state-of-the-art accuracy on

the ASVspoof 2019 dataset, which reported EER values as low as 0.63%. The new model has a modular architecture that is flexible which provides strong performance in noisy and 'real-world' conditions. However, due to the model's complexity, additional pruning or optimization may be required when deploying AASIST for effective production use.

Part 3: Documentation & Analysis

1. Implementation Process

Challenges Faced:
Lack of real audio data: In the early stages of development, we were limited to dummy data for the duration of each study, which limited the actual level of understanding we could have achieved.
Training GRU models: GRU models require sequences with a meaningful temporal structure. The training of the models continued to be unstable due to misalignment from random noise in the dummy waveforms.
How we addressed them: We built a simple alternate label dummy dataset that simulated binary classification (bonafide vs spoof).We built a smaller version of RawNet2 to mitigate overfitting and decrease the complexity.
Assumptions made: Audio samples were 1-second clips sampled at 16kHz.The distribution of the bonafide and spoof classes were equal.The model was only a prototype to show feasibility.

2. Analysis

Why RawNet2:
An end-to-end raw audio model: The model does not require handcrafted features such as spectrograms.
Well defined in the ASVspoof benchmarks: It has performed well in public deepfake detection challenges.
Temporal model with GRU: This helps in capturing speech dynamics and maintaining voice consistency.
How the model works:
Input: 1D raw audio waveform.
Conv1D layers: were used to extract low-level acoustic features.
GRU layer: is used for capturing temporal dependencies, and voice-patterns.
FC Sigmoid: outputs a binary spoof/ bonafide prediction.

Performance of Dummy Dataset:General accuracy varies depending on randomness here; potentially ~60-70%.The performance is purposively not indicative of the use of this application in the real world, however the overall pipeline works.

Noticed Strengths:

Nice design; good feel for extension and integrating APIs.Good for trialling with raw audio inputs.

Noticed Weaknesses:GRU could struggle with long/noisy sequences without attention mechanisms.Dummy dataset lacks variability and realism; limits generalisation.

Suggestions for Improvement:

Utilise real datasets: such as ASVspoof 2019 LA, WaveFake, or TIMIT spoof augmentations.

More advanced models: Consider LSTM Attention or Transformers for more robust temporal modelling.

Utilising pretrained weights: utilising some of the pretrained RawNet2 models trained on the ASVspoof.

Pipeline extensions: Add some post-processing, confidence thresholds, and noise robustness.

3. Reflection Questions

1. What were the most significant challenges in implementing this model?

Lack of access to labeled, real-world spoofed audio limited training and evaluation. Simplifying RawNet2 while maintaining key features required careful balancing.

2. How might this approach perform in real-world conditions vs. research datasets?

Real-world data contains background noise, varied accents, compression artifacts — likely reducing accuracy. Models trained only on clean datasets might fail without domain adaptation or noise augmentation.

3. What additional data or resources would improve performance?

A diverse corpus of spoofed speech from multiple synthesis techniques (TTS, vocoders, voice conversion). Augmented data covering different environments and speaking conditions. More training time and compute to enable full RawNet2 with deeper layers.

4. How would you approach deploying this model in a production environment?

Model serving: TorchScript or ONNX export for low-latency inference.
API exposure: FastAPI or Flask to handle audio uploads and return predictions.
Preprocessing: Normalize audio (volume, sampling rate) and apply VAD (voice activity detection).
Monitoring: Include confidence scores and human-in-the-loop reviews for uncertain predictions.