Part 3: Documentation & Analysis

Devam Tanwani

1. Implementation Process

Challenges Encountered

Long Processing Time: It was very time consuming for my laptop to go through all the dataset points (audio files).

Thus, I reduced the number to 2000 (1000 bonafide and 1000 spoof).

Feature extraction from 2000 .wav files using Wave2Vec2 was time-consuming since each file needed to be loaded, resampled (if needed), processed through a deep model, and pooled.

How Challenges Were Addressed

- **Progressive Extraction**: We used a loop with `try/except` blocks to handle exceptions gracefully and continued extracting features from valid files.
- **Resampling**: We added a `torchaudio.transforms.Resample()` function to ensure all audio was converted to 16kHz before feeding into Wave2Vec2.
- **Sampling**: We reduced the dataset to a balanced set of 1000 bona-fide and 1000 spoof files to keep processing time manageable.

Assumptions Made

- Each `.wav` file contains a single utterance.
- The labels in the metadata file (`bona-fide` or `spoof`) are reliable.
- Pretrained Wave2Vec2 embeddings are suitable without additional fine-tuning.
- Mean pooling on the final hidden state of Wave2Vec2 captures enough signal for classification.

2. Analysis

Model Selection Rationale

- **Wave2Vec2**: Chosen for its ability to generate high-quality, self-supervised audio embeddings without requiring labeled data. It captures both phonetic and speaker-level

features.

- **SVM**: Selected as a lightweight and interpretable classifier, especially effective for binary tasks and well-suited for small to mid-sized datasets.

How the Model Works

- **Wave2Vec2**:
    - A transformer-based model trained on raw speech to extract embeddings.
    - Converts raw audio waveforms into dense hidden representations using a convolutional encoder followed by a transformer.
    - We applied **mean pooling** to generate a fixed-size feature vector for each audio file.
- **SVM**:
    - A supervised binary classifier that learns a decision boundary (hyperplane) to separate two classes.
    - We used the RBF kernel to allow for nonlinear separation between spoof and bona-fide samples.

Performance Results

- **Classifier Used**: `SVC(kernel='rbf', C=1, gamma='scale')`
- **Evaluation Metrics**: Classification report and confusion matrix.
- For 200 points, accuracy was 0.68
- For 2000 points, accuracy went up to 0.78.
  The total number of dataset points are 31,776 (19,963 bonafide and 11,816 spoof), the trend suggests accuracy could go higher with full data usage.

Observed Strengths

- Strong binary classification performance without end-to-end training.
- The pretrained Wave2Vec2 model helped extract powerful and generalizable features.

- SVM reduces overfitting due to its regularization properties.

Observed Weaknesses

- Long inference time due to Wave2Vec2's heavy architecture.
- Not scalable for real-time or large-scale streaming without optimization.
- Feature extraction requires a significant amount of RAM and GPU if done in parallel.

Suggestions for Future Improvements

- Replace SVM with a small feedforward neural network (MLP) for potentially better decision boundaries.
- Fine-tune the Wave2Vec2 model end-to-end with a classification head on the spoof dataset.
- Introduce audio data augmentation (noise injection, speed variation) to improve robustness.
- Optimize inference using model distillation or export to ONNX for real-time use.

3. Reflection

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

Handling the compute cost and memory consumption of Wave2Vec2 during feature extraction.
Ensuring audio was preprocessed correctly (resampling, mono channel) to avoid model errors.
Managing I/O efficiently while processing thousands of audio files.

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

Research datasets are often clean, with minimal background noise and clear labels.
In real-world environments, performance may degrade due to:

1. Background noise
2. Accents and speaking styles not present in training data

New types of spoofing attacks (e.g., text-to-speech, voice cloning with unseen models)

3. What additional data or resources would improve performance?

More spoof samples from various attack types (voice cloning, TTS, replays).
Labeled metadata like speaker gender, age, or emotion could help improve generalization.

Domain-specific finetuning of Wave2Vec2 on speech samples from the same environment (e.g., call centers, public interviews).

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

Preprocess audio to 16kHz and convert to mono channel on-the-fly. Export the model as a pipeline (Wave2Vec2 + SVM) using `TorchScript` or `ONNX`. Use a queue-based audio processing system to batch inputs and perform inference in mini-batches.