Outline - Check-in #3

Title: Is This Really You? Identifying Key Features for Deepfake Audio Detection

Who: Samuel Liu (szliu); Andrew Kim (akim199); Sami Nourji (snourji); Ghali Maata (gmaata)
szliu, akim199, snourji, gmaata
Meeting date: 04/25/2025, 5:30pm

Github link: https://github.com/saminourji/DeepfakeAudioDetection

Introduction

Since our last meeting, we've adjusted our primary goal away from simply re-implementing the research paper (where we would be conducting regular deepfake audio classification) to now working towards contributing to the field of audio deepfake classification. Our new goal is to conduct a comparative study evaluating the relative importance of different extracted audio features from our datasets. Specifically, our project relies on Mel-Frequency Cepstral Coefficients (MFCC) extracted from the audio, but MFCC itself contains 13 features; the first feature is the "energy" or "log-energy" term, the second feature is timbre, etc. We want to understand which of these features is the most informative for deepfake detection.

Our idea emerged during development, when we ran into compute limitations and realized that our model was processing all 13 MFCC features, even though only a subset may meaningfully contribute to classification performance. If the top 5 features can deliver results comparable to using all 13, we can reduce inference time significantly while preserving accuracy. This directly aligns with our original goal of enabling real-time deepfake detection, where low latency and lightweight models are essential. For example, in scenarios like deceptive phone call detection where the task is to determine whether a voice is genuine or generated by deepfake technology, real-time and accurate decisions are critical.

This is thus a comparative study of the performance of different features on a binary classification task.

A 2023 real-time deepfake detection study (which achieved 99.3% average classification accuracy on real-time classification), confirms that MFCCs are pivotal for deepfake detection; their analysis identifies MFCC 2 and MFCC 18 as the most discriminative features, while MFCC 5 shows no significant contribution. However, their findings are based on a dataset they generated themselves using retrieval-based voice conversion, which may not reflect the complexity or diversity of real-world deepfakes. In contrast, our study applies a similar analysis

to ASVspoof 2019, a widely used benchmark in audio deepfake detection. Evaluating feature importance on a standardized and trusted dataset allows for more generalizable insights and better comparison across future research.

Another paper published in March 2025 introduces an interesting architecture for (non-real-time) deepfake audio classification. It leverages a Siamese CNN with a new "StacLoss" function and self-attention modules to achieve 98% accuracy on the more robust AVSspoof2019 deepfake dataset. We use this architecture as the foundation of our own work. However, instead of focusing on raw classification, we adapt it to perform a comparative analysis of MFCC features by systematically removing them and evaluating their individual contributions through counterfactual experiments.

Challenges: What has been the hardest part of the project you've encountered so far?

One of the main challenges we encountered was translating the paper into functioning code, as many implementation-level details were left unspecified. A key example of this was figuring out how to implement the paper's description of "batch hard triplet mining." While the concept is clearly explained – selecting the hardest positive and negative for a given anchor – it wasn't immediately clear where and how to compute this in practice. Should the hardest triplets be computed dynamically within each batch (online), or should the epoch be pre-processed into hard batches and stored for repeated access during training (offline)? This led us to explore the trade-offs between the two approaches. Offline hard-triplet lists are easy on both code and GPU resources, requiring no additional computation and working with any batch size, but they can go stale as the model improves, limiting final accuracy. In contrast, online batch-hard mining keeps triplets challenging throughout training and typically leads to better performance, but at the cost of increased GPU memory and time. Specifically, online mining requires computing a full $B \times B$ pairwise distance matrix each step, adding $O(B^2d)$ flops and B^2 floats of GPU memory, while offline triplets avoid this overhead entirely. Ultimately, we chose online hard triplet mining, as we determined that the benefits of continuously challenging the model outweighed the computational overhead and would lead to faster convergence.

Insights: Are there any concrete results you can show at this point?

- In the training dataset from 2019 ASVspoof, we have found that there are 18,452 bonafide audio samples, 163,114 spoof audio samples, and 67 unique speakers. (Shown in Figure 1).
- Defined a class TripletAudioDataset, inheriting from PyTorch's Dataset class, that naively/randomly creates triplets of tensors following the format (anchor tensor, positive

tensor, negative tensor). Each triplet's tensors are from the same speaker so that stac_loss distances are properly comparable. Some of the tensors are printed in Figure 2.

- The results of our preprocessing (See Figure 2) allowed us to verify that each triplet indeed consisted of three valid tensor segments – anchors, positives, and negatives – all of shape (1, 13, 400). These outputs validated the tensor structure post-MFCC preprocessing and confirmed that triplets were loading properly into memory for training. This step was essential before implementing more complex logic like hard triplet mining, as it ensured the pipeline was functionally sound, and the data was correctly formatted and accessible.
- We have completed the model architecture, specifically the attention, stac_loss, and siamese_cnn modules. We also built the training loop, which takes the triplets and does a forward pass through the model described in the Siamese CNN paper. Our validation mode is set to return the average loss per epoch.

Figure 1 - Summary statistics from the ASVspoof 2019 dataset after preprocessing. The dataset includes 18,452 bonafide (real) samples, 163,114 spoofed (fake) samples, and 67 unique speakers.

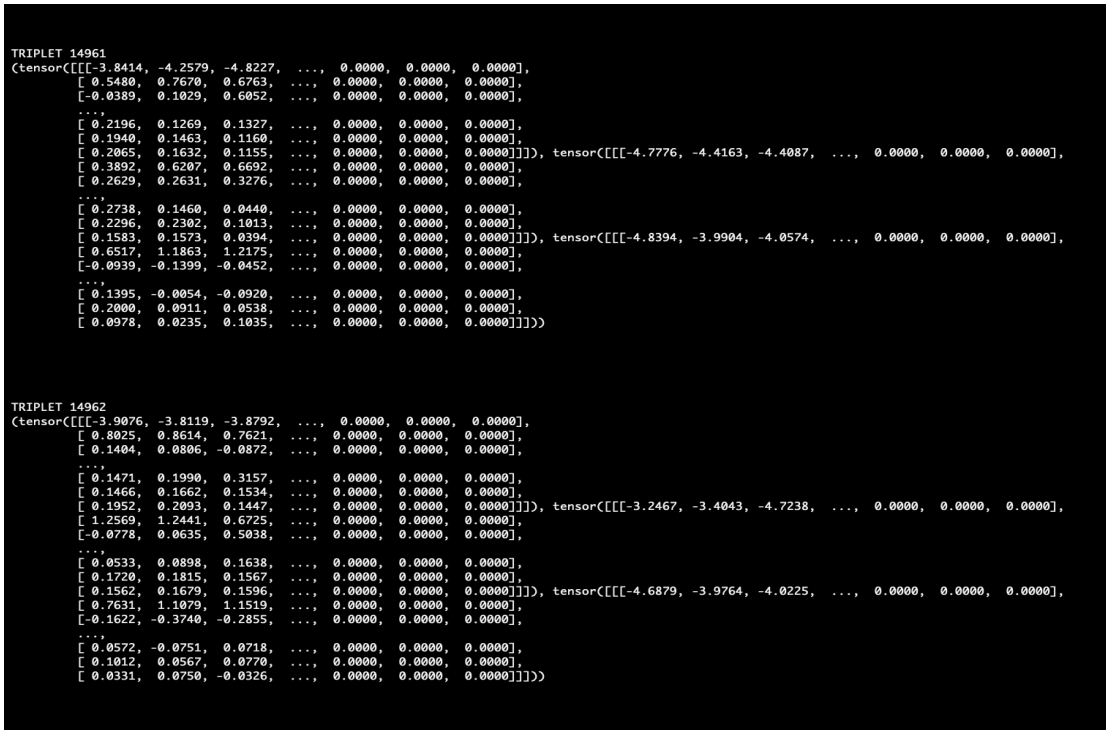

Figure 2 - Sample outputs from the TripletAudioDataset class. Each triplet consists of tensors representing an anchor, a positive (same speaker as anchor), and a negative (different speaker). This confirms that MFCC-processed segments are correctly structured and loaded into memory for training.

Plan: Are you on track with your project?

We're on track with our project. As discussed during our last check-in, our goals of completing preprocessing and nearly finish building full architecture have been achieved. The last step in building our architecture is implementing consistent feature removal (for our counterfactual experiments) before passing the data through the rest of the pipeline.

- **What do you need to dedicate more time to?**

Here are our tasks for the following days, and we are confident that we will finalize them by the April 30th pester presentation and May 2nd final project deadline:

- Finalize the architecture to support feature removal
- Begin training multiple models (one where each feature is removed) and gather data to ensure statistical significance
    - Compute precision, recall, and analyze the mean and variance for each setup
- Compile results into a comparative table and summarize key findings
- Design a poster covering our motivation, model architecture, the paper that inspired us, the methodology for comparison, and the final results

- Draft a paper-style report including the same findings and analysis

- **What do you need to dedicate more time to?**

Our current priority is training. We want to allocate much time as possible to tune hyperparameters and iterate between experiments and result interpretation. To scale this process, we're splitting training tasks among the team. Rather than relying on a single device, we're distributing different versions of our comparative study across our four available devices. We will use OSCAR as part of the training, and all team members already have requested and received access to the platform.
As part of this parallel training process, we're ensuring that our local environments are standardized and that all models are trained on the same dataset. Consistent version control and strong communication are essential, and we're prioritizing both to streamline training.

- **What are you thinking of changing, if anything?**

As mentioned earlier, we've already shifted away from simply reimplementing the original paper. Instead, we've reframed our project around a comparative analysis of MFCC feature importance. This change was driven by practical compute constraints and the broader goal of enabling lightweight, real-time deepfake detection. By identifying and ranking the most predictive MFCC coefficients, we aim to inform the design of more efficient models. This feature-level understanding also opens up new possibilities for interpretability in audio deepfake detection—specifically, by highlighting which aspects of the signal are most informative to differentiate deepfake generation.