Source link - https://github.com/media-sec-lab/Audio-Deepfake-Detection?tab=readme-ov-file#top-repositories

Part 1- Research and selection

Paper 1-

Learning from yourself: a self-distillation method for fake speech detection

Link- https://arxiv.org/pdf/2303.01211

Used fake speech detection for synthetic speech that is used to attack automatic speech recognition systems. Used self- distillation framework- deepest segment- teacher model, shallow segments- student models. Loss functions- Hard loss, Feature loss, soft loss.

Key technical innovation that interests me –

application of self-distillation within the FSD context. The deepest network guiding the shallower networks, the model captures fine-grained information -spectrogram defects and mute segments, -important for detecting fake speech. Good for the discriminative capabilities of the model without increasing its complexity.

-Performance metrics –

Dataset- ASVspoof 2019 LA and PA.

EER of 2.5% on ASVspoof 2019 dataset.

Outperforms MFCC and spectrogram-based approaches.

-Why This Approach is Promising:

Works on raw audio waveforms, reducing manual feature engineering.

Highly adaptable to different speech synthesis models.

-Potential Limitations or Challenges:

Computationally expensive

Struggles with highly unseen AI-generated speech without fine-tuning.

Paper 2-

Towards end-to-end synthetic speech detection

Link- https://arxiv.org/pdf/2106.06341

Time-domain Synthetic Speech Detection Net (TSSDNet), an end-to-end model for synthetic speech detection -no hand-crafted feature extraction. ResNet- and Inception-style structures, showing superior performance on ASVspoof2019 and better generalization to ASVspoof2015.

-Key Technical Innovation:

No manual feature extraction.

Uses ResNet- and Inception-style CNN architectures for deep feature extraction.

-Reported Performance Metrics:

EER of 0.81% on ASVspoof2019 dataset.

Better generalization to unseen attacks compared to traditional spectrogram-based models.

-Why This Approach is Promising:

End-to-end detection , less preprocessing.

Potential Limitations or Challenges:

Computationally intensive due to raw waveform processing.

May require extensive training data for different attack types.

Paper 3-

Continual learning for fake audio detection

a method called Detecting Fake Without Forgetting (DFWF) to improve fake audio detection. The key innovation is using continual learning with knowledge distillation loss and positive sample alignment (PSA) to retain knowledge while adapting to new spoofing attacks. The approach is tested on the ASVspoof2019 dataset, showing better performance than fine-tuning by reducing the Equal Error Rate (EER).

Link- https://arxiv.org/pdf/2104.07286

-Key Technical Innovation:

Continual Learning - for fake speech attacks while preserving knowledge of previous attacks.

Uses Knowledge Distillation Loss to prevent forgetting when learning new data.

Positive Sample Alignment is introduced to minimize variations in real speech.

-Reported Performance Metrics:

ASVpoof 2019 dataset.

lower (EER) than prior fine-tuning.

Better generalization to unseen attacks.

-Why This Approach is Promising:

Learns new attacks without retraining from scratch.

long-term for evolving AI-generated speech attacks.

do not degrade on older attack types.

-Potential Limitations or Challenges:

Higher computational cost due to continual updates.

Requires diverse datasets.

ALMOST ALL OF THE PAPERS AND MODELS I TRIED READING REQUIRED HUGE GPU REUIREMENTS AND ARE COMPUTATIONALLY COSTLY.

Implementation –

Dataset used- LA (train) section of ASVpoof 2019 dataset. - Huge size of dataset. Training section used.

Mostly Implemented a part of the 2nd paper –

Residual Convolutional Neural Networks (ResNet-style blocks) With Recurrent Neural Networks (GRU - Gated Recurrent Unit)

This design is characteristic of models like RawNet2, which was originally proposed for audio classification tasks, especially speaker verification and spoof detection, using raw audio waveforms.

Why this ?

End-to-End Learning from Raw Audio (1D-CNN + GRU)

Directly trained model from raw audio forms instead of handicrafted models.

The architecture uses:

1D Convolutional Layers + Residual Connections:

- feature extractors, learning directly from raw waveforms.
- ResBlocks (inspired by ResNet) allow the network to learn deeper models without vanishing gradients, improving both performance and convergence.

GRU

- RNN-based layer captures temporal dynamics in the extracted features — very useful in audio/speech, where sequence information matters.

Fully Connected Layer:

- Converts the GRU output into final class predictions.

1. Avoid pre-processing complexity (like computing spectrograms).
2. Retain phase information, which gets lost in spectrograms.
3. Learn features directly from raw waveforms, which might contain subtler, discriminative patterns that hand-crafted features miss.

Correctness and Design Justification

- Residual Connections are implemented correctly to maintain gradient flow and support deeper architecture.
- Temporal Modeling is appropriately handled by GRU, which suits the sequential nature of audio signals.
- Feature Extraction begins directly from raw waveforms using Conv1D, allowing the model to learn relevant frequency and temporal patterns without requiring spectrogram conversion.
- Classification Layer outputs logits for two classes, suitable for training with nn.CrossEntropyLoss().

Novelty and Practical Considerations

- The model avoids using precomputed spectrograms (e.g., Mel, MFCC), instead learning representations end-to-end from raw waveform data, inspired by RawNet2.
- The combination of residual CNN blocks with GRU provides a strong inductive bias for audio classification tasks — the CNN captures local frequency features, while GRU captures long-term context.

Performance Bottlenecks

During training, performance may degrade (e.g., training only 6% after 2 hours) due to:

- Long audio sequences (e.g., 16,000 samples per second)
- Memory and time complexity of GRU processing long sequences
- Potential large batch size or lack of optimization in the data loader

Suggested Improvements:

- global average pooling before the GRU to reduce sequence length.
- batch sizes that fit comfortably in GPU memory.
- Monitoring disk I/O and CPU-GPU data transfer bottlenecks.