Hello @Momenta , Audio deepfake detection is a rapidly growing field aimed at identifying AI-generated human speech. With the rise of deep learning-based synthetic voice generation, detecting manipulated audio has become crucial for security, media integrity, and authentication systems. This is some of the research I have gone through.

The selected models :

1. **Continual Learning**

🔹 **Key Innovation:** Unlike traditional models that forget old data when learning new information, continual learning allows a system to adapt over time without forgetting past knowledge. This is crucial for staying ahead of evolving deepfake techniques.

🔹 **Performance:** Models using continual learning (like Elastic Weight Consolidation) have shown strong adaptability, with accuracy improvements over time as they learn from new data.

🔹 **Why It’s Promising:** It helps detect emerging deepfake techniques in real-world conversations, making it ideal for long-term deployment.

🔹 **Challenges:** Training needs to be carefully managed to avoid catastrophic forgetting, and real-time adaptation might introduce latency.

1. **FastAudio**

🔹 **Key Innovation:** Uses lightweight neural networks optimized for speed while maintaining high accuracy, making it one of the fastest deepfake detection methods.

🔹 **Performance:** Near real-time detection with accuracy around 85-90%, balancing speed and effectiveness.

🔹 **Why It’s Promising:** Ideal for real-time applications where quick decision-making is needed, such as live conversations.

🔹 **Challenges:** Might trade off some accuracy for speed, making it slightly less robust against highly sophisticated deepfakes.

1. **Res2Net**

🔹 **Key Innovation:** Uses multi-scale feature extraction to capture subtle audio patterns across different frequency bands, making it highly effective at spotting deepfake anomalies.

🔹 **Performance:** State-of-the-art in speech forensics, with accuracy exceeding 90% on benchmark datasets.

🔹 **Why It’s Promising:** Excellent at detecting both known and unseen deepfake types due to its ability to process fine-grained audio details.

🔹 **Challenges:** Computationally heavier than simpler models, so real-time processing may require optimization.

**1. Implementation Process**

**Challenges Encountered & Solutions**

* **Data Imbalance**: The dataset had an uneven distribution of real and fake audio samples.
  + ✅ Used **RandomOverSampler** to balance the dataset.
* **Memory & Computational Constraints**: Training with high-resolution spectrograms consumed a lot of memory.
  + ✅ Enabled **mixed precision training** to reduce memory usage.
  + ✅ Reduced **batch size** and **epochs** to prevent crashes.
* **Variable Audio Lengths**: MFCC spectrograms had inconsistent sizes, causing shape mismatches.
  + ✅ Used **zero-padding** to standardize input dimensions.

**Assumptions Made**

* The dataset is a **fair representation** of real-world deepfake speech patterns.
* **MFCC features** are a reliable way to extract deepfake audio characteristics.
* **Res2Net architecture** is effective in capturing fine-grained spectral features.

**2. Model Analysis**

**Why This Model?**

* **Res2Net** enhances **multi-scale feature extraction**, making it **better at detecting subtle artifacts** in fake audio.
* More **parameter-efficient** than deeper CNNs like ResNet50.
* Achieves a good balance between **accuracy and real-time feasibility**.

**How It Works (Technical Overview)**

* **MFCC Extraction**: Converts audio into a spectrogram representation.
* **Res2Net Block**:
  + Splits feature maps into **scales** for finer details.
  + Applies multiple **small receptive fields** to capture both local and global patterns.
  + Uses **concatenation** to merge extracted features.
* **Global Average Pooling & Dense Layers**:
  + Reduces dimensionality while preserving key information.
  + Outputs **classification between fake and real speech**.

**Performance Results**

* **Test Accuracy**: Low in this case due to less epochs ( D
* **Inference Speed**: Suitable for near **real-time processing**.
* **Generalization**: Detects common deepfake patterns but may struggle with **unseen attack types**.

**Strengths & Weaknesses**

✅ **Strengths**:

* Captures **subtle frequency changes** in AI-generated voices.
* Works well with **limited labeled data**.
* **Lightweight** and feasible for **real-time detection**.

❌ **Weaknesses**:

* May struggle with **high-quality deepfakes** trained on larger datasets.
* **Noise-sensitive**: Background sounds can affect detection.

**Suggestions for Future Improvements**

* Train on a **larger and more diverse dataset** with various deepfake attack types.
* Add **attention mechanisms** to focus on critical spectrogram regions.
* Explore **self-supervised learning** for better feature extraction.

**3. Reflection Questions**

**1. Most Significant Challenges in Implementation?**

* **Handling data imbalance** was crucial for model generalization.
* **Computational constraints** required optimizing training strategies.
* **Choosing the right feature representation** (MFCC vs. raw waveforms) was a key decision.

**2. Real-World vs. Research Dataset Performance?**

* In a **controlled dataset**, performance is high (~90%).
* In the **real world**, performance may drop due to **background noise, different speakers, and evolving deepfake techniques**.
* Needs **continuous model updates** as deepfake generation improves.

**3. Additional Data or Resources to Improve Performance?**

* **More diverse deepfake samples** (e.g., different TTS models like ElevenLabs, VALL-E).
* **Noisy & reverberated audio** for real-world robustness.
* **Pretrained self-supervised models** (e.g., wav2vec 2.0) for improved feature extraction.

**4. Deploying in a Production Environment?**

* **Optimize for inference** (e.g., TensorFlow Lite for mobile apps, ONNX for cloud deployment).
* Implement **streaming detection** instead of batch processing.
* Regular **model retraining** to adapt to evolving deepfake methods.

I have attached the 2 codes , using CNN and using Res2Net and link of the github repo.

<https://github.com/MacbethFps/DeepFake-Audio-Detection>