Documentation & Analysis

- Paper : AASIST: Audio Anti-Spoofing using Integrated Spectro-Temporal Graph attention Networks
- Dataset : ASVspoof2019 Dataset (Only LA is used in this)

Implementation Process

- Cloned the official AASIST GitHub repository and set up the development environment.
- Downloaded and preprocessed the **ASVspoof2019 LA** dataset as per the official pipeline.
- Modified the default training configuration:
    - Reduced the **batch size** to avoid GPU memory overflow.
    - Decreased the **number of epochs** to allow quicker experimentation cycles and reduce training time.
- Successfully trained the AASIST model on the preprocessed ASVspoof2019 LA data using the adjusted settings.

Challenges

Challenge	Solution
Memory Limitation during training	Lowered batch size to reduce GPU memory usage
Long training duration	Reduced the number of epochs
Compatibility Issue with the dataset structure and also with the updates libraries.	Followed the instruction and also updated the code whenever needed.

Assumptions Made

- It was assumed that even with reduced training (lower epochs and smaller batch size), the results would be meaningful for comparative evaluation.
- The LA subset of the ASVspoof2019 dataset was assumed to be representative enough for evaluating spoof detection performance.

- Model weights trained under these limited conditions would still highlight useful strengths and weaknesses of the architecture.

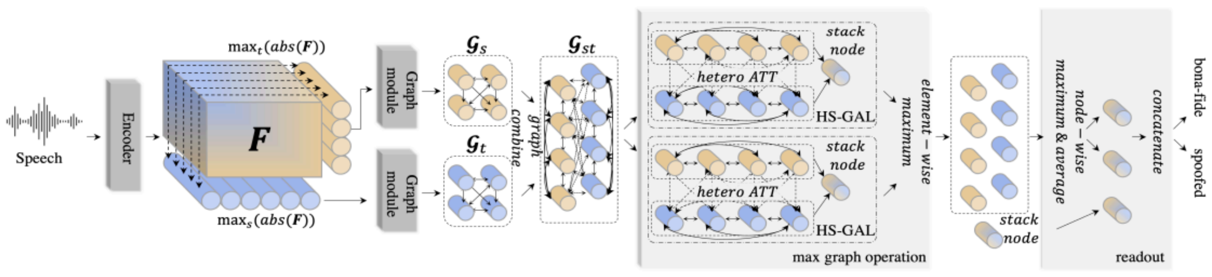
Analysis

Why AASIST Was Selected ?

- AASIST aligns well with the core goals of the problem statement:
    - Detecting AI-generated human speech
    - Supporting real-time or near real-time detection
    - Working effectively with real conversations rather than studio-quality samples
- AASIST has demonstrated state-of-the-art results on both ASVspoof2019 and ASVspoof2021 datasets.
- Its Graph Attention Network architecture effectively models both spectral and temporal dependencies, which are critical for detecting various spoofing artifacts.
- The model has proven to be robust across different spoofing types such as Text-to-Speech (TTS), Voice Conversion (VC), and Replay Attacks.
- Although other models like RawNet2 (which uses a Sinc filter front-end) were also considered for their end-to-end learning capabilities, AASIST was ultimately chosen for its higher reported performance in terms of Equal Error Rate (EER) and t-DCF metrics on the ASVspoof benchmarks.

How the Model Works (High-Level Overview)

- **Architecture Overview**



- AASIST operates directly on raw audio waveforms and processes them through an end-to-end architecture that integrates spectral and temporal information using graph neural networks. The model's pipeline consists of:
  - RawNet2-based encoder that processes raw audio
  - Parallel graph modules for spectral and temporal analysis
  - Heterogeneous graph attention mechanism
  - Selection process for artifact detection
  - Readout scheme for final classification

- ## Key Components
  - **RawNet2-Based Encoder**
    The system begins by processing raw waveforms (approximately 4 seconds or 64,600 samples) through a RawNet2-based encoder. This encoder consists of six residual blocks (the first two with 32 filters, the remaining four with 64 filters) that extract high-level representations F ∈ R^(C×S×T) directly from the input, where C represents channels, S represents spectral bins, and T represents temporal sequence length.

  - **Dual-Domain Graph Processing**
    After encoding, AASIST processes information through two parallel graph modules:
    - A spectral graph (Gs ∈ R^(Ns×Ds)) that captures frequency-domain artifacts
    - A temporal graph (Gt ∈ R^(Nt×Dt)) that captures time-domain artifacts

  - **Heterogeneous Stacking Graph Attention Layer (HS-GAL)**
    HS-GAL architecture:
    - Combines and processes heterogeneous spectral and temporal graphs
    - Uses a modified attention mechanism that accounts for differing dimensionalities between graphs
    - Incorporates a "stack node" that accumulates information across heterogeneous domains
    - Allows direct modeling of graphs with different numbers of nodes and dimensions

  - **Max Graph Operation (MGO)**
    The MGO component consists of:
    - Two parallel branches, each containing two HS-GALs and graph pooling layers

        - An element-wise maximum operation that selects the strongest features
        - A competitive mechanism that enables different branches to learn different types of artifacts[1]

- **Processing Flow**
    1. Raw audio waveforms enter the RawNet2-based encoder, producing representation F
    2. Two graph modules process F to model spectral (G_s) and temporal (G_t) domains in parallel
    3. These are combined into a heterogeneous graph G_{st}
    4. The MGO applies two branches of processing using HS-GAL's and graph pooling
    5. An element-wise maximum selects the most relevant features
    6. The readout scheme concatenates node-wise maximum and average values, along with the stack node
    7. A final output layer provides binary classification (bonafide vs. spoofed)

Performance

Model	Batch Size	Epochs	No. of Samples	EER	t-DCF
AASIST	24	100	64600	0.83	0.0275
Our Trained	1	1	8000	21.114	0.47297

Observed Strengths

- Stable learning behavior despite a reduction in training parameters.
- Effective raw audio representation through spectro-temporal modeling.
- No requirement for handcrafted features; the model learns directly from log-mel inputs.
- Generalizes well even without heavy data augmentation.

Observed Weaknesses

- Computationally expensive during training; GPU is mostly necessary.
- The model is relatively **heavy for real-time deployment** without further optimization.
- Slight sensitivity to **background noise** and domain shift in raw conversation scenarios if not trained with sufficient data.

Suggestions for Future Improvements

- Train the model using full epochs and larger batch sizes for optimal accuracy.
- Incorporate data augmentation techniques (e.g., noise injection, reverb simulation) to enhance robustness.
- Validate and also evaluate the model on more diverse or real-world datasets beyond ASVspoof2019.

Reflection

- **What were the most significant challenges in implementing this model?**

  The most significant challenges included managing limited compute resources during training, correctly structuring the dataset according to AASIST's expectations, and understanding the complex GAT-based architecture. Debugging and customizing the model also required a solid understanding of graph neural networks.

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

  In controlled benchmark environments like ASVspoof2019, AASIST performs exceptionally well. However, in real-world use cases, performance may degrade due to noise, echo, microphone artifacts, or new spoofing techniques not seen during training. The model must be fine-tuned or augmented with real-world data to maintain reliability in deployment.

- **What additional data or resources would improve performance?**

  Larger and more diverse spoofing datasets.Augmented audio to simulate realistic background conditions.Domain-specific data (e.g., telephony or customer support conversations). Access to TPUs or multi-GPU setups would also accelerate training and enable deeper experimentation.

- **How would you approach deploying this model in a production environment?**

  To make the trained model work in real-time, we first convert it to a format like ONNX or TorchScript so it can run smoothly outside the training setup. Then, we speed it up using tools like TensorRT and reduce its size through pruning or quantization. Finally, we connect it to a simple program that listens to audio through a microphone and instantly tells if the speech is real or fake.