Unified Neural Pipeline

Unified Neural Pipeline for Target Speaker Identification and Multispeaker ASR

Version 1.3 | Release: Nov 2025

1. Objective

Design and implement a **Target Speaker Diarization and ASR System** capable of isolating a specific speaker's voice from a multi-speaker conversation, transcribing all participants and supporting real-time streaming input.

- Separate the **target speaker's voice** using a short reference clip.
- Perform **speaker diarization**, **ASR** and **punctuation restoration**.
- Support both **offline batch processing** and **streaming mode**.
- Output structured JSON with speaker labels, timestamps and transcriptions.

2. Input and Output Specifications

Input Files:

- `mixture_audio.wav` – multi-speaker recording.
- `target_sample.wav` – 3–10 s reference clip of the target speaker.

Expected Output:

- `target_speaker.wav` – clean separated voice of the target speaker.
- `diarization.json` – per-speaker transcription and timestamps.

Example JSON Output:

```
[
  {"speaker": "Target", "start": 0.45, "end": 5.62,
   "text": "Hello, how are you?", "confidence": 0.97},
  {"speaker": "Speaker_B", "start": 5.63, "end": 10.25,
   "text": "I'm doing well, thank you.", "confidence": 0.95}
]
```

3. System Architecture Overview

An end-to-end fusion of open-source SOTA models for noise suppression, speaker separation, diarization, recognition and post-processing. The design must be modular, configurable and maintainable.

Stage	Model / Description
Endpoint Detection	CAM++ Diarization — detects speech boundaries.
Overlap Detection	PyAnnote Diarization — handles simultaneous speakers.
Audio Denoising	UVR-MDX-Net — removes noise and reverberation.
Voice Activity Detection	FSMN-Monophone VAD — segments active speech regions.
Speech Separation	MossFormer2 (fine-tuned) — isolates mixed voices.
Audio Restoration	Apollo — restores and enhances degraded audio.
Speaker Recognition	ERes2NetV2-Large — matches target speaker embeddings.
ASR	Paraformer / Whisper / SenseVoice — high-accuracy transcription.
Punctuation Restoration	CT-Transformer — adds punctuation and casing.

4. Main Tasks

1. **System Design:** Design a modular, multi-model pipeline defining data flow from raw audio to text.
2. **Implementation:** Build modular Python code (e.g., `TargetDiarization.py`, `ASRModule.py`) using GPU and async operations.
3. **Per-Speaker Output:** Produce timestamped transcripts with confidence scores and (if multilingual) language tags.
4. **Web/API Interface:** Implement REST and WebSocket endpoints supporting offline and streaming inference.
5. **Visualization:** Create a timeline view of speaker turns and transcriptions.

5. Evaluation Criteria

Category	Evaluation Parameters
System Design & Architecture	Modularity, logical structure, scalability.
Code Quality & Documentation	Readability, maintainability, logging practices.
Performance & Accuracy	Diarization, separation, and ASR precision under noise.
Innovation & Optimization	GPU utilization, latency, embedding logic, real-time handling.
Deployment Readiness	REST/CLI usability, efficiency, reproducibility.

6. Bonus Challenge — Real-Time Streaming Diarization

Extend your system to operate in **real time:**

- Accept microphone or streamed audio input over WebSocket.
- Perform continuous target-speaker detection, diarization and ASR with sub-second latency.
- Maintain RTF ≤ 1.0 and handle multiple concurrent sessions.

Thank you for your effort and creativity. We look forward to your submission.