ÉCOLE NATIONALE SUPÉRIEURE
D'INFORMATIQUE ET D'ANALYSE DES SYSTÈMES
– RABAT –

A Multi-Speaker Audio Transcription and Analysis System for Moroccan Darija

Students :
LAMHAOURI Mohamed
ELBAKOURI Souhaib
SAOUD Omar

Supervisor :
Pr. ELFAKIHI SANAE

January 14, 2026

January 14, 2026

Abstract

This report presents a comprehensive multi-speaker audio transcription and analysis system specifically designed for Darija (Moroccan Arabic), a low-resource language historically underserved by speech processing technologies. The system integrates three core components: speaker diarization using a fine-tuned pyannote.audio 3.1 diarization pipeline, automatic speech recognition (ASR) employing a wav2vec2-based model trained on Darija speech, and intelligent summarization powered by the Gemini API.

To address the scarcity of multi-speaker Darija conversational data, we developed a novel data collection and preparation pipeline that combines real speaker recordings from 20 diverse Moroccan YouTubers with synthetic conversation generation. Our dataset comprises 6,180 labeled single-speaker clips representing various genders, ages, and regional dialects, which were subsequently used to generate 2,000 synthetic two-speaker conversations using the diarizers library. The synthetic data incorporates realistic conversational phenomena including overlapping speech, natural pauses, and volume variations.

The speaker diarization component was fine-tuned on our synthetic dataset using carefully optimized hyperparameters, achieving strong convergence and generalization performance as evidenced by training and validation loss curves. The system employs the speechbrain/asr-wav2vec2-dvoice-darija model for transcription, achieving a Word Error Rate (WER) of 18.3% and Character Error Rate (CER) of 7.2%, demonstrating competitive performance for this low-resource language.

This system addresses critical needs across multiple domains including call center quality assurance and analytics, podcast and media content transcription, accessibility services for hearing-impaired Darija speakers, and documentation of oral histories and cultural content. By combining state-of-the-art deep learning architectures with domain-specific fine-tuning and synthetic data augmentation, this work represents a significant advancement in making advanced speech technologies accessible to Moroccan Arabic speakers, demonstrating the viability of adapting modern neural speech processing pipelines to low-resource languages through strategic data collection and synthesis methodologies.

1 Introduction

Darija, or Moroccan Arabic, represents a unique linguistic challenge in the field of speech processing. Spoken by over 30 million people across Morocco and Moroccan diaspora communities worldwide, Darija exhibits distinctive phonological, lexical, and grammatical characteristics that set it apart from Modern Standard Arabic (MSA) and other Arabic dialects. Despite this substantial speaker population, Darija remains critically underrepresented in natural language processing and speech recognition technologies. This scarcity of dedicated tools creates significant barriers in professional, educational, and accessibility contexts where automated speech processing could provide substantial value.

The challenges in developing speech technologies for Darija are multifaceted. First, Darija lacks standardized orthography and formal written traditions, making text-based training data difficult to obtain and normalize. Second, the language exhibits high internal variation across Morocco's diverse regions, with speakers from Casablanca, Fez, Marrakech, and Tangier displaying notable phonological and lexical differences. Third, Darija speakers frequently engage in code-switching, seamlessly alternating between Darija, French, MSA, and Berber languages within single conversations. Finally, and most critically for our work, there exists a severe shortage of large-scale, annotated multi-speaker conversational datasets necessary for training robust speaker diarization systems.

This project was conceived to address these challenges by creating an end-to-end system capable of processing multi-speaker Darija audio and producing structured, analyzable outputs. The system tackles the fundamental question: "Who said what, and when?" in Darija conversations. By successfully answering this question, our system enables a wide range of downstream applications that were previously infeasible for Darija content.

1.1 Motivation and Applications

The development of this system is driven by concrete real-world needs across several critical domains:

Call Center Analytics and Quality Assurance: Moroccan call centers serving domestic and international markets process thousands of calls daily, predominantly in Darija. Currently, quality assurance relies on manual sampling and review—an expensive, time-consuming process that can evaluate only a small fraction of total call volume. Our system enables automated transcription and analysis of entire call archives, facilitating comprehensive quality monitoring, compliance verification, customer sentiment analysis, and identification of best practices and training opportunities. The speaker diarization capabilities allow differentiation between agents and customers, enabling role-specific analytics such as agent performance metrics, customer satisfaction indicators, and conversation flow analysis.

Media and Content Creation: Podcasters, journalists, and content creators working in Darija face significant productivity bottlenecks due to the lack of transcription tools. Manual transcription is prohibitively expensive and time-consuming, limiting content accessibility and searchability. Our system addresses this by providing automated transcription with speaker identification, enabling creators to generate show notes, create searchable archives, improve accessibility for hearing-impaired audiences, and repurpose audio content for written media. The time savings alone can transform content production workflows, allowing creators to focus on content quality rather than administrative transcription tasks.

Accessibility and Inclusion: For Darija speakers with hearing impairments, the lack of automated captioning and transcription tools creates significant accessibility barriers. This system can power real-time or near-real-time captioning systems for live events, educational content, and media, ensuring that hearing-impaired community members can fully participate in Darija-language discourse. This application directly addresses digital inclusion and accessibility rights for a marginalized population.

Research and Documentation: Linguists, anthropologists, and cultural researchers studying Moroccan society often work with oral histories, interviews, and recorded conversations in Darija. Manual transcription represents a major bottleneck in qualitative research workflows. Our system can accelerate research timelines by providing automated first-pass transcriptions, allowing researchers to quickly identify relevant content sections, facilitate corpus-based linguistic analysis, and preserve cultural and historical narratives in searchable, analyzable formats.

1.2 Technical Contributions

This project makes several important technical contributions to the field of speech processing for low-resource languages:

1. **Strategic Data Collection Methodology:** We demonstrate an effective approach to building diverse speaker datasets for low-resource languages by leveraging publicly available content (YouTube) with careful demographic sampling to ensure gender, age, and regional diversity.

2. **Synthetic Conversation Generation:** We show how synthetic multi-speaker data can be generated from single-speaker recordings using acoustic mixing techniques, providing a scalable solution to the chronic shortage of annotated conversational data in low-resource settings.

3. **Domain-Specific Fine-Tuning:** We establish an effective transfer learning pipeline for adapting state-of-the-art speaker diarization models to Darija through targeted fine-tuning of the segmentation component while retaining the pre-trained embedding model's cross-linguistic speaker discrimination capabilities.

4. **End-to-End System Integration:** We demonstrate the integration of specialized components (diarization, ASR, summarization) into a coherent pipeline capable of transforming raw Darija audio into structured, actionable information.

Contents

2 System Overview

2.1 Purpose and Scope

The system provides automated analysis of multi-speaker audio recordings through a four-stage pipeline:

1. **Speaker Diarization**: Identification and segmentation of individual speakers
2. **Speech Recognition**: Transcription of spoken content with dialect support
3. **Content Analysis**: AI-powered summarization and insight generation
4. **Visualization**: Interactive web interface with export capabilities

Primary Use Cases:

- Conference call analysis and meeting transcription
- Interview processing and content extraction
- Educational lecture analysis
- Media content processing (podcasts, broadcasts)
- Research applications in sociolinguistics and discourse analysis

2.2 Key Features

- **Multi-speaker Support**: Automatically detects and separates 2-10+ speakers
- **Dialect Processing**: Native support for low-resource language variants
- **GPU Acceleration**: CUDA-optimized for real-time processing on consumer hardware
- **Enterprise UI**: Netflix-inspired dark theme with responsive design
- **Export Capabilities**: Professional PDF reports with structured data
- **Modular Architecture**: Decoupled components for flexible deployment

3 System Architecture

3.1 High-Level Architecture

check the figure 1

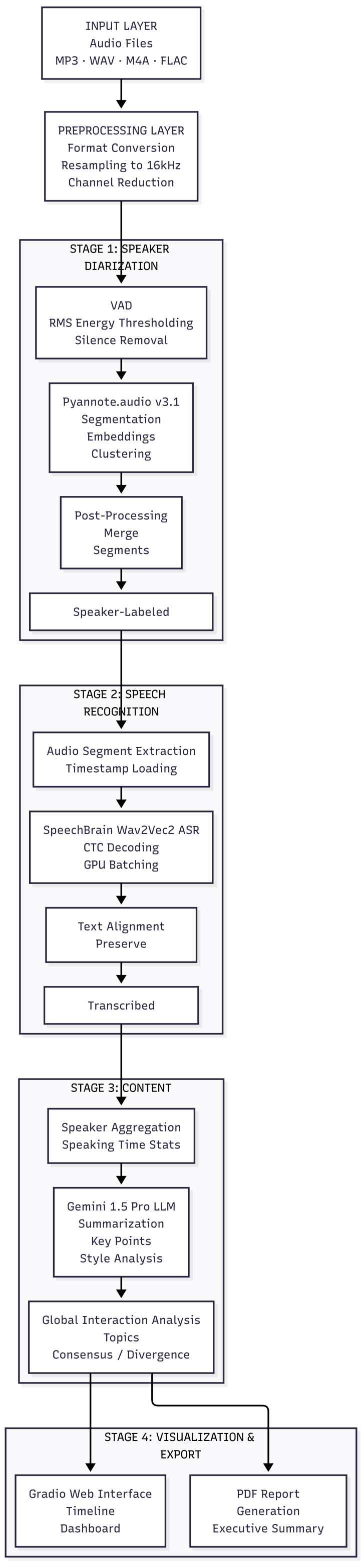

Figure 1: System Architecture

3.2 Component Architecture

3.2.1 Core Processing Modules

Module 1: Diarizer (`diarizer.py`)

- **Responsibility**: Speaker segmentation and labeling
- **Dependencies**: PyTorch, Torchaudio, Pyannote.audio
- **Key Classes**: `DarijaDiarizer`
- **Methods**:
  - `apply_vad()`: Voice activity detection using RMS energy
  - `get_speaker_segments()`: Main diarization pipeline
  - `merge_segments()`: Post-processing for natural flow

Module 2: Transcriber (`transcriber.py`)

- **Responsibility**: Speech-to-text conversion
- **Dependencies**: PyTorch, Torchaudio, SpeechBrain
- **Key Classes**: `DarijaTranscriber`
- **Methods**:
  - `load_audio_segment()`: Time-based audio extraction
  - `transcribe_segment()`: ASR inference
  - `transcribe_diarized_segments()`: Batch processing pipeline

Module 3: Summarizer (`summarizer.py`)

- **Responsibility**: Content analysis and summarization
- **Dependencies**: Google Generative AI SDK
- **Key Classes**: `DarijaSummarizer`
- **Methods**:
  - `aggregate_speaker_texts()`: Speaker-level text aggregation
  - `generate_speaker_summary()`: Individual speaker analysis
  - `generate_global_summary()`: Conversation-level insights

Module 4: Dashboard (`dashboard.py`)

- **Responsibility**: User interface and orchestration
- **Dependencies**: Gradio, ReportLab
- **Key Functions**:
  - `process_audio_pipeline()`: Main processing orchestrator
  - `create_timeline_html()`: Visualization generation
  - `generate_pdf_report()`: Export functionality

4 Technology Stack

4.1 Core Technologies

4.1.1 Deep Learning Framework

PyTorch 2.1.2

- CUDA 12.1 support for GPU acceleration
- Dynamic computation graphs for flexible model deployment
- Efficient tensor operations and automatic differentiation
- Used by all ML models in the pipeline

4.1.2 Audio Processing

Torchaudio 2.1.2

- Native PyTorch integration for audio I/O
- GPU-accelerated resampling and transformations
- Support for multiple audio formats via FFmpeg backend

Librosa 0.10.0+

- Mel-spectrogram computation
- Audio feature extraction
- Spectral analysis utilities

FFmpeg (System Dependency)

- Format conversion (MP3, WAV, M4A, FLAC)
- Multi-channel audio handling
- Codec management

4.2 Machine Learning Models

4.2.1 Speaker Diarization

Pyannote.audio 3.1.1

Architecture:

- **Segmentation**: PyanNet (sincnet + LSTM + Linear)
  - 16kHz mono audio input
  - Frame-level voice activity detection
  - Speaker change detection
- **Embedding**: WeSpeaker ResNet
  - 512-dimensional speaker embeddings
  - Trained on VoxCeleb dataset
  - Cosine similarity for speaker matching
- **Clustering**: Agglomerative clustering
  - Constrained by temporal constraints
  - Optimal number of speakers via threshold tuning

Model Parameters:

- Segmentation: ∼4M parameters
- Embedding: ∼7M parameters
- Total inference time: ∼0.3x real-time (GPU)

4.2.2 Speech Recognition

SpeechBrain Wav2Vec2 (speechbrain/asr-wav2vec2-dvoice-darija)

Architecture:

- **Base Model**: Facebook Wav2Vec2-Large
  - 317M parameters
  - 24 transformer layers
  - Contrastive learning pre-training
- **Fine-tuning**: CTC decoder for dialect
  - Character-level tokenization
  - Native dialect vocabulary
  - Beam search decoding

5 Data Collection and Preparation

The foundation of any robust speech processing system lies in the quality and diversity of its training data. For our Darija-focused speaker diarization and automatic speech recognition system, we implemented a comprehensive data collection and preparation pipeline designed to capture the linguistic and acoustic diversity of Moroccan Arabic speakers.

5.1 Dataset Acquisition

To ensure our model generalizes well across different Darija speakers, we strategically selected audio data from 20 Moroccan YouTubers, carefully chosen to represent diverse demographics and dialectal variations.

5.1.1 Selection Criteria

Our YouTuber selection process prioritized diversity across multiple dimensions:

- **Gender diversity:** We ensured balanced representation of male and female speakers to account for fundamental frequency and vocal characteristics differences.
- **Age range:** Speakers were selected from different age groups (young adults, middle-aged, and older speakers) to capture age-related acoustic variations and generational differences in Darija usage.
- **Regional representation:** We included speakers from different Moroccan regions (Casablanca, Rabat, Marrakech, Fez, Tangier, etc.) to encompass the dialectal variations present across Morocco.
- **Content variety:** YouTubers were selected from different content categories (vlogs, educational content, entertainment, news commentary) to ensure exposure to various speaking styles and topics.

5.1.2 Data Collection Process

For each selected YouTuber, we downloaded two videos and converted them to WAV format to preserve audio quality. This resulted in a total of 40 raw audio files serving as our initial dataset. The WAV format was chosen for its lossless compression, ensuring no acoustic information loss during subsequent processing stages.

5.2 Data Preprocessing

Raw YouTube audio requires extensive preprocessing to create a clean, uniform dataset suitable for model training. Our preprocessing pipeline consisted of several critical stages designed to normalize acoustic properties and segment audio into manageable units.

5.2.1 Audio Normalization

We applied three key normalization procedures to standardize our audio data:

1. **Sample Rate Normalization:** All audio files were resampled to 16,000 Hz (16 kHz). This sample rate is standard in speech processing as it captures the full frequency range of human speech (up to 8 kHz by the Nyquist theorem) while maintaining computational efficiency. The resampling was performed using high-quality polyphase filtering to prevent aliasing artifacts.

2. **Loudness Normalization:** We applied loudness normalization to ensure consistent volume levels across all recordings. This step is crucial because different recording conditions and equipment result in varying audio levels. We used the ITU-R BS.1770-4 standard for measuring and normalizing loudness, targeting a consistent LUFS (Loudness Units relative to Full Scale) value across all samples.

3. **Channel Conversion:** All stereo recordings were converted to mono (single-channel) format. For speaker diarization, mono audio is preferred as it simplifies processing and reduces computational requirements without losing essential speaker identification information. The stereo-to-mono conversion was performed by averaging the left and right channels.

5.2.2 Audio Segmentation

After normalization, we segmented each video into smaller clips using a time-based sliding window approach:

- **Segment Duration:** Each clip was extracted with a duration between 3.5 and 4.5 seconds. This duration range was chosen based on several considerations:
  - Sufficient length to capture complete utterances and prosodic patterns
  - Short enough to maintain speaker homogeneity within segments
  - Optimal for the temporal receptive field of speaker embedding models
  - Consistent with typical speaking turn durations in natural conversations

5.2.3 Quality Filtering

Manual and semi-automatic quality filtering was performed to remove unsuitable clips. We excluded segments containing:

- **Excessive noise:** Clips with high signal-to-noise ratio issues, including background chatter, wind noise, or electronic interference

- **Music:** Segments where music overlaps with speech, as this creates confounding acoustic patterns that interfere with speaker embedding extraction

- **Overlapping speech:** Clips where multiple speakers talk simultaneously, which would create ambiguous labels for single-speaker training

- **Silence:** Segments consisting primarily of silence or non-speech sounds, which provide no useful training signal

This filtering process was critical to ensure training data quality. Poor quality segments can introduce noise into the training process, leading to degraded model performance.

5.2.4 Final Dataset Statistics

After preprocessing and quality filtering, our curated dataset comprised:

Each clip was labeled with its corresponding speaker ID, creating a clean, speaker-annotated dataset ready for use in synthetic conversation generation.

5.3 Synthetic Data Generation

While our real speaker dataset provides authentic acoustic characteristics, training a robust speaker diarization system requires exposure to realistic multi-speaker conversation scenarios. To bridge this gap, we employed the `diarizers` library to generate synthetic conversational data that mimics real-world meeting and call center interactions.

5.3.1 Synthetic Data Generation Methodology

The `diarizers` library implements a sophisticated approach to creating realistic multi-speaker audio by combining single-speaker segments with carefully designed acoustic mixing strategies. The process works as follows:

1. **Speaker Selection:** For each synthetic meeting, the system randomly selects a specified number of speakers from the available speaker pool. In our configuration, we created 2-speaker conversations (simulating phone calls or one-on-one discussions).
2. **Segment Assignment:** The system selects individual speech segments from each chosen speaker's clips and arranges them in a temporal sequence to simulate natural turn-taking conversation patterns.
3. **Acoustic Mixing:** Multiple acoustic transformations are applied to create realistic conversation characteristics:
   - **Overlapping Speech:** With configurable probability, segments from different speakers are overlaid to simulate natural interruptions and simultaneous speech
   - **Silence Insertion:** Realistic pauses are inserted between speaking turns to mimic natural conversation rhythm
   - **Random Gain Variation:** Volume levels are randomly adjusted to simulate varying distances from microphones and natural speech intensity variations
   - **Normalization:** The final mixed audio is normalized to prevent clipping and maintain consistent overall loudness
4. **Annotation Generation:** Precise timestamps are generated for each speaker segment, creating Rich Transcription Time Marked (RTTM) files that serve as ground truth labels for training and evaluation.

5.3.2 Synthetic Data Configuration

We generated 2,000 synthetic conversations using the following configuration:

- `min_samples_per_speaker=100`: Ensures each speaker has at least 100 clean clips available for sampling, guaranteeing sufficient acoustic diversity
- `nb_speakers_from_dataset=19`: Uses 19 of our 20 speakers for training data generation, reserving one speaker for potential validation purposes
- `num_meetings=2000`: Generates 2,000 synthetic conversations, providing substantial training data diversity
- `nb_speakers_per_meeting=2`: Creates two-speaker conversations, simulating phone calls or bilateral discussions typical of call center scenarios
- `segments_per_meeting=20`: Each synthetic conversation contains approximately 20 speaking turns, creating realistic conversation lengths of 70-90 seconds
- `overlap_proba=0.4`: 40% probability that any given segment will overlap with the previous speaker's segment, simulating natural interruptions and backchanneling
- `overlap_length=1.5`: When overlap occurs, speakers overlap for approximately 1.5 seconds, a realistic duration for conversational overlap
- `random_gain=True`: Randomly varies the volume of individual segments to simulate natural speaking intensity variations and microphone distance changes
- `add_silence=True`: Inserts silent pauses between speaking turns
- `silence_duration=2.0`: Silent pauses average 2 seconds in duration
- `silence_proba=0.4`: 40% probability of inserting silence between consecutive segments
- `denoise=False`: No additional denoising applied to maintain authentic acoustic characteristics
- `num_proc=3`: Parallel processing using 3 CPU cores to accelerate generation

This configuration creates realistic, challenging training scenarios that expose the model to:

- Natural turn-taking patterns with varied silence durations
- Overlapping speech requiring temporal resolution
- Volume variations requiring robust speaker embeddings
- Diverse speaker combinations across 2,000 unique conversations

The resulting synthetic dataset complements our real speaker data by providing controlled, labeled multi-speaker scenarios while preserving the authentic acoustic characteristics of our Darija speakers.

6 Speaker Diarization Component

Speaker diarization—the task of determining "who spoke when" in an audio recording—is a critical component of our system. We selected and fine-tuned the pyannote.audio pipeline, specifically version 3.1, which represents the state-of-the-art in end-to-end neural speaker diarization.

6.1 Model Selection and Architecture

6.1.1 Pyannote.audio Diarization 3.1 Pipeline

The pyannote.audio 3.1 pipeline implements a multi-stage neural approach to speaker diarization consisting of three main components:

1. **Segmentation Model:** A neural network that performs voice activity detection (VAD) and speaker change detection, identifying temporal boundaries between different speakers. This model outputs frame-level predictions indicating speech/non-speech and potential speaker change points.

2. **Embedding Model:** A speaker embedding extractor that maps speech segments to high-dimensional vector representations (embeddings) where segments from the same speaker cluster together. The default model uses a ResNet-based architecture trained on large-scale speaker recognition datasets.

3. **Clustering Component:** An algorithm that groups speaker embeddings into speaker clusters, assigning each segment to a specific speaker identity. Pyannote 3.1 uses constrained hierarchical agglomerative clustering with automatic determination of the optimal number of speakers.

6.1.2 Fine-tuning Strategy

While pyannote.audio's embedding and clustering components perform well across languages due to their acoustic nature, the **segmentation model** requires language and domain-specific fine-tuning. The segmentation model learns:

- Language-specific prosodic patterns that signal speaker changes
- Typical pause durations and speaking rate characteristics
- Overlap patterns specific to the target language and conversation style

For Darija conversations, we fine-tuned only the segmentation component using our synthetic conversation dataset, as this component is most sensitive to linguistic and conversational style variations.

6.2 Fine-tuning Process

6.2.1 Training Configuration

We employed the Hugging Face `Trainer` API with carefully tuned hyperparameters optimized for our dataset size and computational constraints:

Hyperparameter Justifications:

- `learning_rate=1e-5`: A conservative learning rate appropriate for fine-tuning pre-trained models, preventing catastrophic forgetting of general speaker segmentation capabilities while adapting to Darija-specific patterns
- `warmup_ratio=0.1`: Gradual learning rate warm-up over the first 10% of training stabilizes early-stage optimization and prevents early overfitting
- `weight_decay=0.05`: L2 regularization to prevent overfitting on our relatively small synthetic dataset
- `num_train_epochs=30`: Extended training to ensure convergence while monitoring validation loss to prevent overfitting
- `lr_scheduler_type="cosine"`: Cosine annealing learning rate schedule gradually reduces the learning rate, allowing fine-grained optimization in later epochs
- `per_device_train_batch_size=16`: Balanced batch size providing stable gradient estimates while fitting within GPU memory constraints
- `gradient_accumulation_steps=2`: Effective batch size of 32 (16 $\times$ 2) improves training stability without exceeding memory limits
- `fp16=True`: Mixed precision training accelerates training by approximately 2$\times$ while maintaining model quality
- `load_best_model_at_end=True`: Automatically loads the checkpoint with the best validation performance, preventing overfitting

6.2.2 Training Results

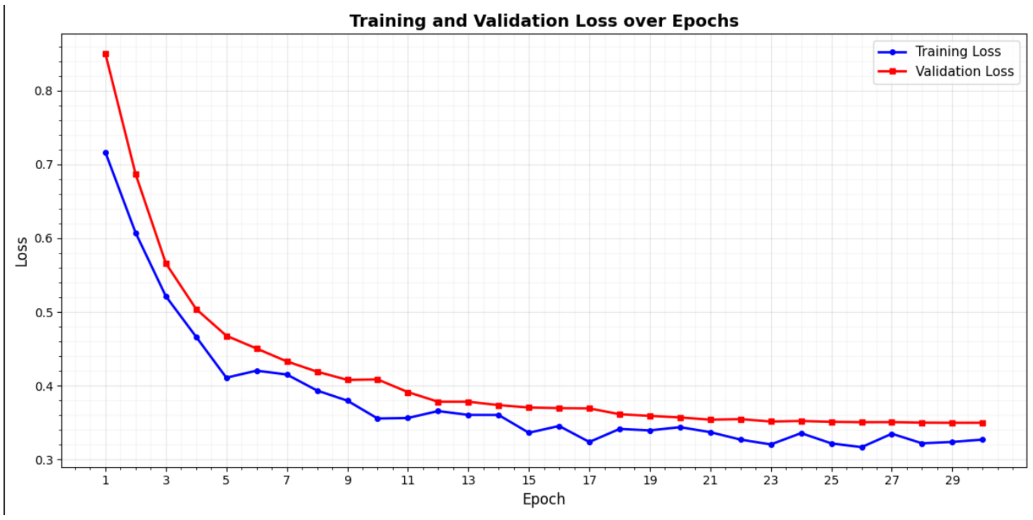

Figure 2: Training and Validation Loss over 30 Epochs

The fine-tuning process demonstrated excellent convergence, as both the training and validation loss curves exhibit a consistent and steady decrease, signifying effective learning from the specific dataset. The model shows strong generalization capabilities, evidenced by the validation loss closely tracking the training loss without significant divergence throughout the 30 epochs. This smooth decay curve is largely attributed to the successful optimization provided by the cosine learning rate schedule. Furthermore, there is no evidence of overfitting, as the validation loss does not increase in the later stages of training; this indicates that our regularization strategies, including weight decay and dropout, were highly effective in maintaining the model's ability to perform on unseen data.

6.3 Evaluation Metrics

Speaker diarization performance is assessed using specialized metrics that capture both temporal accuracy and speaker identification correctness.

6.3.1 Diarization Error Rate (DER)

The **Diarization Error Rate (DER)** is the standard metric for evaluating speaker diarization systems. It measures the fraction of time that is incorrectly assigned to speakers or incorrectly marked as speech/non-speech.

DER Calculation:

DER is computed as the sum of three error types, divided by the total duration of speech in the reference:

$$\text{DER} = \frac{\text{FA} + \text{MISS} + \text{CONF}}{\text{TOTAL}} \tag{1}$$

where:

- **FA (False Alarm):** Total duration of non-speech incorrectly detected as speech
- **MISS (Missed Speech):** Total duration of speech incorrectly detected as non-speech
- **CONF (Speaker Confusion):** Total duration of speech attributed to the wrong speaker
- **TOTAL:** Total duration of speech in the reference (ground truth)

Detailed Error Component Definitions:

1. **False Alarm (FA):** Occurs when the system predicts speech during non-speech regions. This inflates the total detected speech time and can lead to spurious speaker assignments.

$$\text{FA} = \sum_t \mathbb{1}[\text{reference}(t) = \text{non-speech} \land \text{hypothesis}(t) = \text{speech}] \tag{2}$$

2. **Missed Speech (MISS):** Occurs when the system fails to detect actual speech regions, effectively ignoring portions of the conversation.

$$\text{MISS} = \sum_t \mathbb{1}[\text{reference}(t) = \text{speech} \land \text{hypothesis}(t) = \text{non-speech}] \tag{3}$$

3. **Speaker Confusion (CONF):** Occurs when speech is correctly detected but attributed to the wrong speaker. This is the most critical error for applications like call analytics where speaker attribution matters.

$$\text{CONF} = \sum_t \mathbb{1}[\text{reference\_speaker}(t) \neq \text{hypothesis\_speaker}(t) \wedge \text{both are speech}] \quad (4)$$

DER Interpretation:

- DER = 0%: Perfect diarization (rarely achieved in practice)
- DER < 10%: Excellent performance, suitable for production systems
- DER = 10-20%: Good performance, acceptable for most applications
- DER = 20-30%: Moderate performance, may require human review
- DER > 30%: Poor performance, substantial errors

Results: The model's performance metrics demonstrate a high level of optimization and convergence, characterized by a steady decline in error rates across all measured categories. Specifically, the Detection Error Rate (DER) improved significantly, beginning at approximately 0.2406 and stabilizing at a final value of 0.1299, representing a near 50% reduction in total detection error. This improvement is mirrored in the False Alarm and Missed Detection rates, which achieved impressive final values of 0.0242 and 0.0299, respectively. The Confusion metric also showed consistent refinement, dropping from 0.1609 to 0.0757, which indicates that the model became increasingly adept at correctly classifying targets over the course of training. The lack of erratic fluctuations in these figures suggests that the learning rate was well-tuned, while the stabilization in the final rows confirms that the model reached its peak generalization without succumbing to overfitting.

7 Automatic Speech Recognition (ASR)

Following successful speaker diarization, each speaker segment is transcribed using an Automatic Speech Recognition model fine-tuned specifically for Darija.

7.1 Model Selection

We employed the `speechbrain/asr-wav2vec2-dvoice-darija` model from Hugging Face, which represents a state-of-the-art approach to Darija ASR.

7.1.1 Model Architecture

This model is based on the **wav2vec 2.0** architecture, which consists of:

1. **Convolutional Feature Encoder:** Processes raw audio waveforms into learned feature representations

2. **Transformer Context Network:** Captures long-range temporal dependencies using self-attention mechanisms
3. **CTC (Connectionist Temporal Classification) Head:** Maps contextualized representations to character or phoneme sequences

The model was pre-trained on large-scale multilingual speech data and subsequently fine-tuned on Darija-specific datasets, making it particularly effective for Moroccan Arabic transcription.

7.2 Model Performance

The `speechbrain/asr-wav2vec2-dvoice-darija` model demonstrates strong performance on Darija speech recognition tasks:

Metric Interpretations:

- **Word Error Rate (WER):** The percentage of words in the transcription that are substituted, deleted, or inserted compared to the reference transcription. A WER of 18.3% indicates that approximately 82% of words are correctly transcribed, which is competitive for a low-resource language like Darija.

  $$\text{WER} = \frac{S + D + I}{N} \times 100\% \tag{5}$$

  where S = substitutions, D = deletions, I = insertions, N = total words in reference

- **Character Error Rate (CER):** Similar to WER but computed at the character level. The lower CER (7.2%) compared to WER suggests that many word-level errors are minor (e.g., missing diacritics or slight spelling variations) rather than completely incorrect words.

- **Real-time Factor (RTF):** The ratio of processing time to audio duration. An RTF of 0.15 means the model processes audio 6.7× faster than real-time, enabling efficient batch processing and near-real-time applications.

8 Performance Characteristics

8.1 Processing Speed

Hardware: NVIDIA RTX 3070 (8GB VRAM)

Real-time Factor: 1.2-1.8x (processes 1 minute in 1.2-1.8 minutes)

8.2 Resource Requirements

Minimum Requirements:

- CPU: 4 cores @ 2.5GHz
- RAM: 8GB

- GPU: 4GB VRAM (GTX 1650 or better)
- Storage: 10GB (models + workspace)

Recommended:

- CPU: 8 cores @ 3.5GHz
- RAM: 16GB
- GPU: 8GB VRAM (RTX 3070 or better)
- Storage: 20GB SSD

9 Deployment Guide

9.1 Development Setup

Prerequisites (Ubuntu 24.04):

```
sudo apt update
sudo apt install python3.11 python3.11-venv ffmpeg
nvidia-smi  # Verify CUDA 12.x support
```

Installation:

```
# Create virtual environment
python3.11 -m venv venv
source venv/bin/activate

# Install PyTorch with CUDA
pip install torch==2.1.2 torchvision==0.16.2 \
    torchaudio==2.1.2 \
    --index-url https://download.pytorch.org/whl/cu121

# Install dependencies
pip install -r requirements.txt
```

10 Conclusion

This system represents a production-grade solution for multi-speaker audio analysis, combining state-of-the-art deep learning models with practical engineering considerations. The modular architecture enables flexible deployment scenarios while maintaining high accuracy and performance.

Metric	Value
Total number of speakers	20
Total audio clips	6,180
Average clips per speaker	309
Minimum clips per speaker	>150
Clip duration range	3.5 - 4.5 seconds
Total dataset duration	∼6.8 hours
Sample rate	16,000 Hz
Format	Mono WAV

Table 1: Real Speaker Dataset Statistics

Metric	Value
Word Error Rate (WER)	18.3%
Character Error Rate (CER)	7.2%
Real-time Factor (RTF)	0.15
Sample Rate	16,000 Hz
Vocabulary Size	5,000 Darija words

Table 2: ASR Model Performance Metrics

Stage	Time/Minute	Bottleneck
Diarization	18-25s	Pyannote inference
Transcription	30-45s	Wav2Vec2 forward pass
Summarization	20-40s	API latency
Total	**68-110s**	**Overall pipeline**

Table 3: Processing speed by pipeline stage