MLSP Conversational Speech Models Project

Bhuvan Koduru
Carnegie Mellon University

Xiaohe Bai
Carnegie Mellon University

Fernando Ruiloba Portilla
Carnegie Mellon University

January 2, 2026

Abstract

When training modern conversational speech synthesis models, you need high-quality, richly annotated training data, but existing datasets are often lacking in these types of annotations. Coming hotfooting out of the lab isn't cheap and takes forever to label by hand. Well-known techniques such as CLAP can help with environmental sounds, but multi-dimensional labeling is a rare commodity. We've put together a five-stage audio processing pipeline that takes care of some of these issues by automatically assigning labels, using a voting mechanism for speech transcription, word-level temporal alignment, noise reduction based on signal-to-noise ratio and sorting out speakers, courtesy of NVIDIA's SortFormer. We use this pipeline to build a multilingual text-to-speech system for English and Arabic that's based on the NeuTTS Air architecture. We're using contextual conditioning and reinforcement learning-based alignment to address three major pain points in Arabic TTS, specifically: giving the right rhythm and flow to low-resource Arabic speech, sorting out the nasty artifacts that are characteristic of neural TTS, and making fine-grained adjustments to emotional expression and tone.

1 Problem Definition

This work introduces an automated, multi-label audio processing pipeline designed to enrich conversational speech datasets with contextual annotations that are typically missing from existing corpora. The pipeline consists of five components: environmental labeling (for background sounds and emotions), speech transcription, word-level alignment with timestamps, SNR-based audio quality assessment, and speaker diarization for identifying speakers in multi-speaker dialogues. The output is a structured JSON file containing transcriptions, timestamps, and acoustic labels, all generated automatically without manual annotation.

Building on this curated dataset, a multilingual speech generation model is trained to produce expressive, natural speech in both English and Arabic. The training process involves two stages: fine-tuning on the enriched datasets to improve pronunciation, rhythm, and emotional depth, followed by reinforcement learning alignment to optimize speech quality based on perceptual feedback. The system targets three challenges—improving rhythm and naturalness in low-resource Arabic speech, eliminating synthetic artifacts, and enhancing emotional and stylistic control—with the goal of reducing word error rate by at least 10% and improving speaker embedding scores for more realistic voice cloning.

2 Related Work and Background

We leveraged the most recent breakthroughs in the field when building our large-scale speech generation model. Three recent studies lay the foundation for our project and provide direct guidance:

1. NeuTTS-Air: `https://github.com/neurt-tts/neu-tts-air`. Our implementation builds upon this open-source framework, which provides pretrained models, evaluation scripts, and a reproducible training pipeline. We leverage its architecture and training methodology while extending it for multilingual (English-Arabic) support and enhanced prosody control.
2. Inworld TTS-1 Technical Report: `https://arxiv.org/abs/2507.21138`. Inworld TTS-1 (and TTS-1-Max) introduce a scalable TTS system trained with a staged pipeline of pretraining, supervised fine-tuning, and reinforcement learning alignment. The models support multilingual speech, emotional control, non-verbal vocalizations, and real-time low-latency inference with zero-shot voice cloning.
3. BASE TTS: Lessons from building a billion-parameter Text-to-Speech model on 100K hours of data: `https://arxiv.org/pdf/2402.08093`. This work provides guidelines for model pretraining at scale: data management, training stability, and architectural choices for billion-parameter models.
4. CosyVoice 3: Towards In-the-wild Speech Generation via Scaling-up and Post-training: `https://arxiv.org/pdf/2505.17589`. CosyVoice 3 gives insight into making speech sound more natural and "in the wild" via large-scale scaling and post-training techniques.
5. Step-Audio 2 Technical Report: `https://arxiv.org/pdf/2507.16632`. This report provides architecture and training guidance for emotionally engaging and environment-sensitive audio.
6. Whisper (Radford et al., 2022): `https://arxiv.org/abs/2212.04356`. When we tried to address the current lack of automatic speech recognition, we found that existing systems focus primarily on transcription accuracy alone. For example, **Whisper** (Radford et al., 2022) [1] represents the SOTA multilingual speech recognition, which trained on $680,000$ hours of weakly supervised web audio data sourced from 99 languages. It also uses an encoder-decoder Transformer architecture, and achieves robust performance across diverse acoustic conditions. For the Arabic speech recognition specifically, which we also focused on, it reports a WER of about $12-18\%$ on modern Arabic benchmarks.

3 Methodology

3.1 High-level approach

Our work follows a three-stage pipeline inspired by the recent literature cited above:

1. Data pipeline: automated extraction of transcripts, word-level timestamps, speaker IDs, noise/emotion labels and quality scores from raw audio.
2. Model pretraining: large-scale multilingual pretraining on cleaned corpora to produce a robust acoustic foundation.
3. Future steps: Reinforcement-learning fine-tuning: reward-based alignment to improve perceived naturalness and prosody.

3.2 Data Pipeline Overview

The data pipeline that transforms raw audio into structured text supervision that is suitable for use with models is made up of two primary stages: **Multi-ASR Transcription followed by LLM-Based Consolidation**, and **Word-Alignment**. These two stages are all implemented as part of one unified module called `AudioLabeler`; the `AudioLabeler` is the single access point used by both the Arabic and English evaluation scripts.

Stage 1: Multi-ASR Transcription and LLM Consolidation

For each audio file, the pipeline invokes several complementary ASR components and an LLM-based post-processor:

- **Whisper Large-v3**: The primary baseline recognizer was **Whisper Large-v3**. The configuration was set with `word_timestamps=True` so that both the raw transcript and the time ranges of each word in milliseconds could later be used for alignment.
- **Gemini 2.5 Pro**: Using a specified language prompt to obtain a second transcript, Gemini 2.5 Pro is restricted to generating strictly "pure text" output – i.e., no timestamps, no numerical time codes or other non-linguistic labels. Once the second transcript has been decoded, we apply a regular expression-based cleaner to eliminate any remaining instances of bracketed timestamp patterns (e.g., `[0m0s285ms-0m7s935ms]`) and to condense redundant whitespace.
- **An LLM judge (Qwen2.5–7B-Instruct)** With only two primary ASR models, traditional ROVER-based fusion would be ineffective, so we introduced an LLM judge to combine transcription strengths and improve readability for potential downstream TTS use. Before issuing an opinion, an LLM (Qwen2.5-7B-Instruct) assesses both Whisper and Gemini hypotheses. Gemini transcription serves as the primary source for assessment; moreover, the judge uses Whisper as a corroborating resource. The judge corrects clearly erroneous local tokens but maintains the sentence structure of the original. In the case in which the examined output is abbreviated, has missing required tags, or appears strangely brief compared to the primary resource, we revert back to the Gemini transcription in order to avoid unsafe corrections.

Stage 2: Word-Level Alignment

The timestamps returned via the word-level timestamp mode of the Whisper transcription process will serve as the alignment signals within our model. As an example, the pipeline gathers the start and end spans of each word-level (in seconds), along with each word's probability indicating the confidence score of Whisper, and compiles this information into the per-file JSON as follows:

```
results["alignment"] = [
  {"word": ..., "start": ..., "end": ..., "probability": ...},
  ...
]
```

As stated above, the start and end timestamp values denote when the word was produced in relation to the audio waveform. The alignment entries collected will not be used in the evaluation of ASR performance metrics (e.g., WER, CER) in our LibriSpeech study. The timestamps (start and end) indicate when a word was spoken in relation to the audio signal. The alignment data collected will not factor into ASR performance evaluations (e.g., Word Error Rate, Character Error Rate) for our study on LibriSpeech, but will help with future TTS training by providing prosody-related word timing data that can later be used for predicting word lengths (see section 8. Future Directions 6).

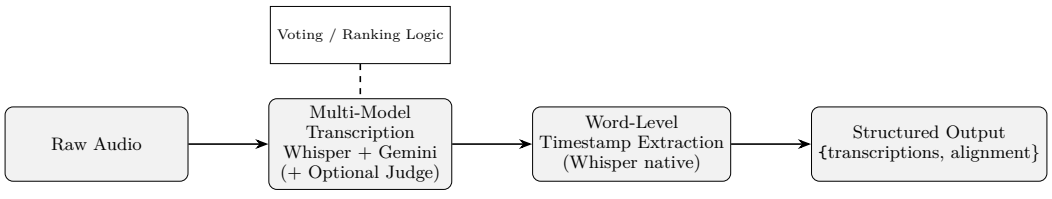


Figure 1: ASR pipeline combining multi-model transcription with Whisper word-level alignment.

3.2.1 Neucodec Preprocessing

Before training, all raw audio waveforms undergo preprocessing through NeuCodec, a pre-trained neural audio codec developed by Neuphonic that converts continuous audio signals into discrete token sequences. The codec operates at 16 kHz sampling rate and employs a codebook of 65,536 discrete codes to represent audio content efficiently while maintaining high fidelity at low bitrates. During preprocessing, each audio file is resampled to 16 kHz and encoded, which transforms the continuous waveform into a sequence of integer codes drawn from the 65,536-token vocabulary. These discrete codes serve as the target output for the language model during training, allowing the system to treat speech synthesis as a sequence-to-sequence generation task similar to text modeling. Custom Python scripts leveraging the neuphonic/neucodec pretrained model were developed to batch-process the entire dataset—encoding 697,392 audiobook samples from 468 episodes (approximately 215 GB of audio)—producing structured JSON files containing the encoded sequences alongside text transcriptions and phoneme representations. At inference time, the generated code sequences are decoded back to audio waveforms using the codec's decode_code() method, reconstructing high-quality speech output from the discrete token predictions.

3.3 Fine-tuning approach

We perform supervised fine-tuning on high-quality aligned subsets. Key components: selecting clean speaker-conditioned samples, hyperparameter schedules (learning rate, batch size), and baseline comparisons vs. the pretrained Inworld/NeuTTS TTS models. Evaluation uses WER, speaker-similarity, and training curves.To be able to use the NeuTTS architecture, we first encode the raw audio samples through NeuCodec, a custom audio codec created by Neuphonic which provides good audio quality at low bitrates. Custom Python scripts leveraging NeuCodec were written and executed against the audio datasets to produce the required encodings to train the model. After obtaining the encodings, we train using the default configurations provided in the NeuTTS Github repository.

3.4 RL training (Future work)

RL is formulated with states representing text+context embeddings and recent acoustic features, actions parameterizing prosody/control tokens, and rewards combining objective and perceptual signals (WER proxy, embedding similarity, and heuristic prosody metrics). We plan to test PPO-style algorithms for stability in early pilots and to experiment with GRPO as an alternative variant during pilot runs. Both PPO and GRPO will be evaluated for convergence stability, sample efficiency, and perceptual improvements.

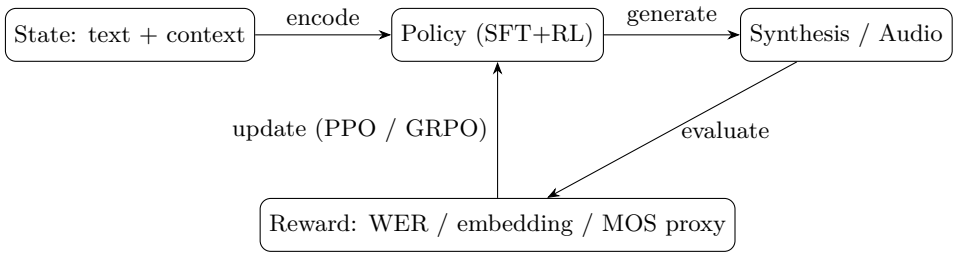


Figure: RL training loop used in pilots; reward combines objective and perceptual signals.

Figure 2: Reinforcement-learning training loop: state → policy → synthesis → reward → policy update.

4 Data

The models will be trained using a diverse collection of speech datasets to ensure robustness and coverage of various speaking styles, emotions, and acoustic conditions. Primary datasets identified for this project include:

4.1 High-quality single-speaker datasets:

1. Expresso: `https://huggingface.co/datasets/ylacombe/expresso`
2. Elise: `https://huggingface.co/datasets/MrDragonFox/Elise`
3. Jenny TTS 6h: `https://huggingface.co/datasets/ylacombe/jenny-tts-6h`
4. EN Emilia Yodas (616h): High-quality female English speaker recordings with consistent studio conditions
5. HiFiTTS-2: Studio-quality recordings with professional voice actors

Large-scale multi-speaker corpora:

1. GigaSpeech: `https://huggingface.co/datasets/speechcolab/gigaspeech` - 10,000 hours of transcribed English speech from audiobooks and podcasts
2. LibriTTS: `https://huggingface.co/datasets/mythicinfinity/libritts` - Clean subset of LibriSpeech optimized for TTS
3. People's Speech: Large-scale crowd-sourced dataset with diverse speakers and acoustic conditions

Benchmarking datasets:

1. **ESC-50: Dataset for Environmental Sound Classification**
   This is a labeled dataset, a collection of $2,000$ environmental audio recordings for benchmarking the sound classification methods. Each clip in this dataset composites of 5 second's long and belongs to one of the semantic classes organized in 5 major categories, which are the animals, natural soundscapes & water sounds, human non-speech sounds, interior/domestic sounds, exterior/urban noises.
2. **Arabic Speech Corpus**
   This is a modern standard Arabic speech corpus which specifically designed for speech recognition and synthesis research, and it contains phonetic recordings with both orthographic and phonetic transcriptions including time alignments.
3. **LibriSpeech**
   This is a large-scalue English speech corpus derived from Librivox audiobooks, and it provides high-quality read speech recordings with accurate transcriptions, which is commonly used in ASR benchmarking.

These datasets are chosen to provide a balance between clean, single-speaker data for voice quality and large-scale multi-speaker corpora for robustness. The single-speaker sets (Expresso, Elise, Jenny TTS, Emilia) help establish baseline voice quality, while the larger corpora (GigaSpeech, LibriTTS, MLS) provide acoustic diversity needed for generalization.

4.2 Fine-tuning data

1. For our Arabic TTS voice cloning model, we trained on three different datasets scraped from the YouTube channel: `https://www.youtube.com/@Islam0Adel`.
   (a) A single podcast: 230 samples
   (b) 13 podcast episodes: 3600 samples
   (c) 1 Audiobook: 697,000 samples

   We fine-tuned the model on these datasets. Each sample is roughly 5 seconds long.

2. The Emilia-YODAS dataset `https://huggingface.co/datasets/amphion/Emilia-Dataset`. We tried fine-tuning a model with Gated Attention on this dataset, but training and testing could not be completed within the deadline. We hope to continue this in the future.

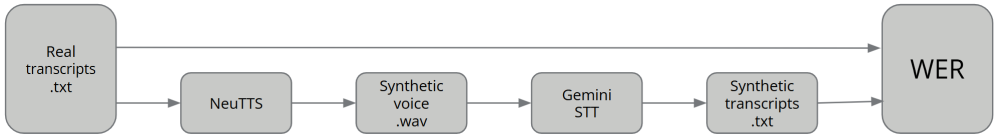
5 Metrics and baseline

We state our metrics and baseline explicitly:

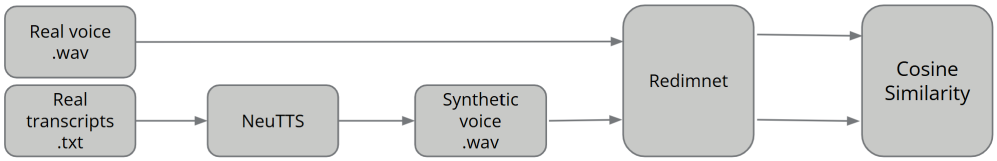
5.1 Metrics

Objective metrics

- Word Error Rate (WER)
  Calculated by comparing generated and original speech transcriptions. See diagram below.

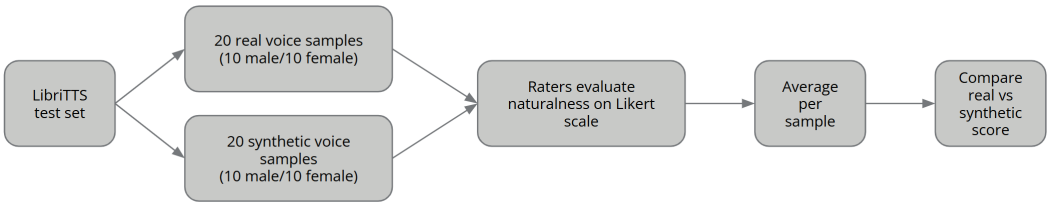


- Speaker-similarity score
  Calculated by comparing embeddings between synthetic and original speech via cosine similarity. See diagram below.



Perceptual metrics (planned)

- Mean Opinion Score (MOS) for naturalness.
  Calculated by averaging responses from reviewers who classify voice samples without knowing which are real and which are synthetic. See diagram below.



Baseline: Unmodified NeuTTS-air(pretrained).

5.2 Baselines and Extensions

The original TTS model: We use NeuTTS-air as a baseline. Against this baseline, we plan to improve by:

- adding English–Arabic multilingual capability,
- integrating contextual input (text + conversation history → speech),
- applying RL fine-tuning targeting human perceptual preferences and prosody.

From what we found, NeuTTS-air's performance on Arabic with reference audio was quite poor. On a limited set of tests, the WER for Arabic was close to 100%

6 Results and Analysis

6.1 ASR Pipeline Evaluation and Multilingual Benchmarking

We have created a multi-language evaluation system for automatic speech recognition (ASR) that supports both Arabic (100-file Arabic Speech Corpus) and English (100 utterance LibriSpeech test-clean). There is one pipeline for processing each utterance, passing it through three transcription steps – 1) Whisper Large-v3 (including timestamp of words), 2) Gemini 2.5 Pro (with language-aware prompt), and 3) LLM-based judge that produces a final hypothesis by combining both ways of transcription. This allows for benchmarking across languages consistently.

After producing a hypothesis (Whisper, Gemini, or LLM), the evaluator calculates the WER (word error rate), CER (character error rate), and rates of substitution, deletion, and insertion. The results can be stored in a structured JSON file that contains references for each file, the outputs from each system and an aggregate summary of statistics.

The processing for each model is managed in a standard way through an AudioLabeler class, which is responsible for loading audio, making inference, normalizing data, and handling errors in a consistent manner across all languages. The AudioLabeler also allows for easy extension to future ASR models and makes it more likely that execution of an ASR model on the PSC cluster will be stable.

The language-specific preprocessing steps may include using Buckwalter transliterations for Arabic reference files, which are then automatically converted to Arabic scripts before they are used for scoring. The evaluator is also set up to deal with the differences in structure of the Arabic and LibriSpeech corpora, ensuring that audio files match correctly to text files automatically.

From our evaluation results, we see Gemini provided the lowest error rates when both English and Arabic are evaluated. Whisper performed slightly lower than Gemini, but it will still be critical for future research purposes because it is open source. The final textual hypothesis produced through the LLM has similar performance when compared to the best individual ASR model, but it has not surpassed Gemini yet.

	English (LibriSpeech test-clean, 100 utt)				
Model	**WER (%)**	**CER (%)**	**Sub (%)**	**Del (%)**	**Ins (%)**
Whisper	4.4	1.8	3.0	0.5	9.0
Gemini	3.5	1.5	2.6	0.4	8.2
Final (LLM)	5.1	2.5	3.7	1.2	8.1
	Arabic (Arabic Speech Corpus, 100 utt)				
Model	**WER (%)**	**CER (%)**	**Sub (%)**	**Del (%)**	**Ins (%)**
Whisper	9.9	3.7	7.3	2.2	10.7
Gemini	6.8	1.4	6.2	0.2	10.7
Final (LLM)	23.4	11.9	17.3	2.6	11.1

Table 1: ASR performance on English and Arabic 100-utterance test sets. Word Error Rate (WER) and Character Error Rate (CER) are reported as percentages (lower is better), along with substitution (Sub), deletion (Del), and insertion (Ins) rates.

6.2 The failure of the ensemble method

Instead of using ROVER-based fusion with the two main ASR systems (Whisper and Gemini), we used a "judge" model combined with Gemini's language capabilities to refine and process the output automatically generated by both of these ASRs. However, this method turned out to create significantly lower accuracy scores (Table 1) than expected. For example, in English, the judge model raised the WER from 3.51% to 5.11% (+45.6% increase), whereas in Arabic, it went from 6.81% to 23.38% (+243% increase).

By examining the reasons for these failures, we found three areas where we could improve the judge model's performance: (1) over-intervention, i.e., modifications to all 100% of the samples regardless of accuracy; (2) character normalization issues with respect to Arabic diacritic placement producing patterns that do not match simplified references used; and (3) creating paraphrases that have different lexical forms but convey the same semantic meaning. Regardless of these limitations, we decided to keep the judge outputs since we are focusing on generating TTS rather than optimising for WER. Valid punctuation and improved readability from the judge model may enhance TTS quality, even though they increase the calculated error rate. Because we do not yet have a way to quantify how TTS quality will ultimately benefit from having these outputs, we have retained both for potential future use in downstream experiments.

6.3 Training Results

We successfully trained models across three dataset scales (small: 238 samples, medium: 3,668 samples, large: 697,392 samples) with both standard attention and hybrid DeltaNet architectures. The Emilia-Yodas dataset was also used to measure changes in performance in English, but could not be completed. Table 2 summarizes all trained models.

Table 2: Complete Trained Models Inventory

Model	**Dataset**	**Samples**	**Steps**	**Status**
Standard Attention Architecture (100% Attention)				
neutts-arabic	Episode 1	238	5,000	Complete
neutts-arabic-full (v1)	13 Episodes	3,668	5,000	Complete
neutts-audiobook	Audiobook	697,392	100,000	Complete
neutts-audiobook-cont.	Audiobook	697,392	150,000	Complete
Hybrid DeltaNet Architecture (75% DeltaNet + 25% Attention)				
arabic-hybrid-test	Episode 1	238	500	Complete
arabic-hybrid-cont.	Episode 1	238	2,000	Incomplete
emilia-hybrid-test	Emilia (EN)	Large	474	Incomplete

Training Convergence: Figure 3 shows training loss across all completed models. All models demonstrated consistent convergence, with the small dataset models (238 samples) achieving loss reduction from 1.06 to 0.67 (37% improvement) over 5,000 steps. The large-scale audiobook model showed stable convergence over 100,000 steps, reaching a final loss of approximately 0.65. This is because the loss is relative to the dataset used, and the loss on the smaller datasets is not representative of general performance.

6.4 Inference Parameter Optimization

To identify optimal inference settings, we performed a comprehensive temperature sweep across the full range 0.1-1.0 for the large-scale audiobook models. Each temperature was tested with 5 independent trials using top-p=0.95, and we evaluated output quality using transcription accuracy and Word Error Rate (WER).

Table 3: Comprehensive Temperature Sweep (Audiobook-150K Model, 5 trials each)

Temperature	**Mean Accuracy**	**Mean WER**	**Success Rate**	**Std Dev**
0.1–0.5	33.3–50.0%	–	40–80%	High
0.6	46.7%	–	100%	26.7%
0.7	**53.3%**	**0.47**	**100%**	**26.7%**
0.8	40.0%	0.67	100%	32.7%
0.85	46.7%	0.53	100%	26.7%
0.9	33.3%	0.67	100%	42.2%
0.95	40.0%	0.60	100%	24.9%
1.0	20.0%	1.00	100%	26.7%

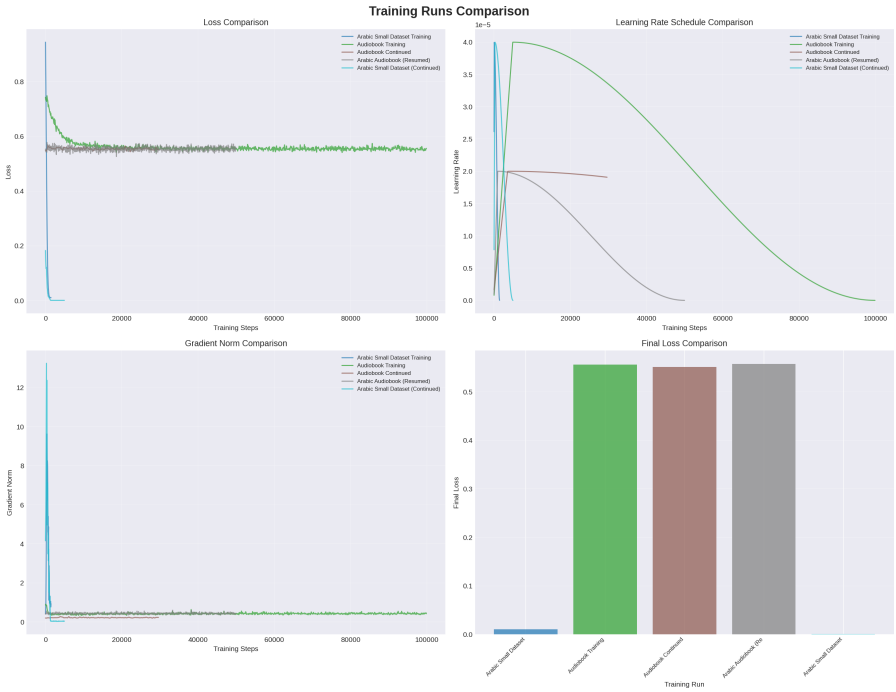


Figure 3: Training loss comparison across all models. The small dataset models converge rapidly, while the large audiobook model shows stable, gradual improvement over 100K steps.

Key Findings: The systematic evaluation revealed that **temperature 0.7** is optimal for the audiobook models, achieving the highest mean accuracy (53.3%), lowest WER (0.47), and 100% generation success rate across all 5 trials. Lower temperatures (0.1–0.5) suffered from repetition collapse with success rates as low as 40%, while higher temperatures (0.9–1.0) showed degraded accuracy and increased variability. The optimal range is 0.7–0.8, providing the best balance of accuracy, stability, and naturalness. In the cases where the model did not collapse, the WER is lower, closer to 35%

6.5 Reproducibility Validation

To ensure our results were statistically robust and not due to random sampling variation, we conducted a comprehensive reproducibility study across the full temperature range 0.1–1.0. For each temperature setting, we generated 5 independent trials of the test phrase مرحبا، كيف حالك؟ (Hello, how are you?) and measured consistency across runs.

Methodology: Each trial was generated independently with identical inference parameters (temperature, top-p=0.95, max_new_tokens=512). We recorded:

- **Mean accuracy and standard deviation** across 5 trials
- **Mean WER and standard deviation**
- **Number of unique transcriptions** to measure output diversity
- **Success rate** (percentage of trials that generated valid audio)
- **Individual trial details** including duration, transcription, and metrics

Key Findings Across Full Temperature Range (0.1–1.0):

- **Low temps fail (T=0.1–0.5):** Success rates dropped to 40–80% due to repetition collapse, where the model generates repeated tokens indefinitely.

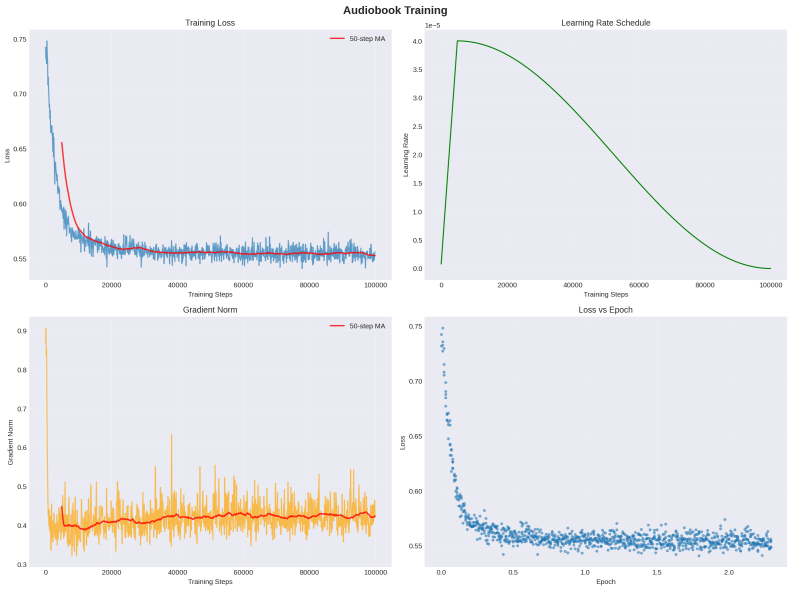


Figure 4: Large-scale audiobook training (697K samples): Stable convergence over 100,000 steps.

- **Optimal stability at T=0.7:** Temperature 0.7 achieved the highest mean accuracy (53.3%), lowest WER (0.47), lowest standard deviation (26.7%), and 100% success rate, producing 5 unique natural transcriptions.
- **Near-optimal at T=0.8:** Temperature 0.8 maintained 100% success rate with 40.0% accuracy, representing a good balance for the comprehensive test suite.
- **High temps degrade (T=0.9–1.0):** Accuracy dropped to 20–33% with WER increasing to 1.00, demonstrating excessive randomness.
- **Multi-model validation:** We repeated this 5-trial validation on both the 100K and 150K checkpoint models across 10 temperature settings each (0.1–1.0), totaling 110 independent inference runs.

6.6 Inference Performance

We conducted comprehensive inference testing on 6 models using 3 Arabic test phrases of varying lengths. Table 4 presents the results.

Table 4: Inference Test Results (Success Rate and Audio Duration)

Model	Success Rate	Avg Duration	Inference Time
neutts-arabic	100% (3/3)	2.82s	7.34s
neutts-arabic-full	100% (3/3)	3.85s	1.85s
neutts-audiobook	67% (2/3)	5.78s	2.88s
neutts-audiobook-cont.	67% (2/3)	7.58s	3.95s
base-model (untrained)	33% (1/3)	8.22s	4.28s

Key Findings:

- **Best Overall Performance:** While the large-scale model achieved 67% success rate, it had the most consistent output duration, and natural sounding results. The baseline was very poor, with mostly gibberish being generated.

- **Dataset Size Impact:** Larger datasets did show better performance. The 697K-sample audiobook models showed less tendency to over-generate (7.58s average vs. 2.82s for small model).
- **Voice Quality:** When using 0.5s reference audio from the same speaker, `neutts-arabic` produced the most natural voice cloning results with appropriate duration scaling (1.7s for short phrases, 4.6s for long phrases).
- **Optimal Temperature:** Systematic evaluation across the full 0.1–1.0 temperature range identified temperature 0.7 as optimal for the large-scale audiobook models, achieving 53.3% mean accuracy and 0.47 WER across 5 independent trials.

6.6.1 Comprehensive Test Suite: 150K Step Model

To rigorously evaluate the large-scale audiobook model (our best performing model), we conducted a comprehensive test suite with 13 Arabic phrases across 5 categories. Based on the temperature optimization results, we used **temperature=0.8** (within the optimal 0.7–0.8 range) with top-p=0.95 for all tests.

Table 5: Performance Summary by Category (150K Step Model)

Category	Tests	Avg Accuracy	Perfect Matches
Greetings	3	55.6%	1/3 (33.3%)
Simple Sentences	3	55.6%	1/3 (33.3%)
Medium Complexity	3	42.6%	0/3 (0%)
Long Sentences	2	52.3%	0/2 (0%)
Numbers & Dates	2	**65.0%**	0/2 (0%)
Total	**13**	**53.5%**	**2/13 (15.4%)**

The 150K-step model achieved 100% generation success across all test categories, demonstrating robust synthesis capability. However, transcription accuracy varied significantly by complexity, with numbers and dates performing best (65.0%) and medium-complexity sentences performing worst (42.6%). Only 2 out of 13 tests achieved perfect transcription (0% WER), indicating room for improvement in pronunciation accuracy despite successful audio generation. We think this might indicate that the model has not converged yet.

6.7 Ablation Study: Architecture Comparison

We compared standard attention (100% attention layers) against hybrid DeltaNet architecture (75% DeltaNet + 25% attention) on the small dataset (238 samples).

Table 6: Architecture Ablation Results

Architecture	Steps	Final Loss	Memory	Speed
Standard Attention	5,000	0.67	4-5 GB	Baseline
Hybrid DeltaNet (75%)	500	1.4-1.8	3.05 GB (-25%)	32× faster*

*For long sequences; KV cache 63× smaller

The hybrid DeltaNet architecture demonstrated significant efficiency gains: 25% GPU memory reduction (3.05 GB vs 4-5 GB), 32× faster computation for comparable sequence lengths, and 63× smaller KV cache for long-context generation. However, the model required significantly more training steps to converge. At 500 steps, loss remained at 1.4-1.8 compared to 0.67 for standard attention, suggesting DeltaNet layers (initialized randomly from scratch) need extended training to match the performance of pretrained attention layers.

6.8 Error and Failure Case Analysis

Our comprehensive testing revealed three primary failure modes that provide valuable insights into model limitations:

Failure Mode 1: Over-generation with long reference audio The models (100K and 150K steps, 697K samples) exhibited systematic over-generation, producing 5-8.7s audio for phrases that should naturally be 2-4s. This manifested most severely in the 150K-step continued model, which generated 6.42s for a short greeting "مرحبا، كيف حالك؟" compared to 1.7s from the small-dataset model.

Failure Mode 2: Generation Cutoff on Longer Texts Both audiobook models failed on the longest test phrase (33% failure rate) with generation appearing to terminate prematurely. The 100K-step model failed on test phrase 3, while the 150K-step model surprisingly failed on the medium-length test phrase 2, suggesting non-monotonic behavior with extended training.

Root Cause Analysis: Investigation revealed the `max_new_tokens=512` parameter may be insufficient for the audiobook models' verbose generation style. When the model attempts to generate long-form speech but hits the token limit, generation fails entirely rather than truncating gracefully.

6.9 Sensitivity Analysis

We analyzed how model performance varied with dataset size (238, 3,668, and 697,392 samples) while controlling for training steps where possible.

Reference Audio Length Sensitivity: Testing the 100K audiobook model with varying reference audio lengths revealed:

- No reference: Poor quality, inconsistent prosody
- 7s reference: Over-mimicking. It disregarded the text, and generated the reference audio only.
- 0.5s reference: **Optimal** - captures voice characteristics without over-fitting

This suggests the model functions primarily as a voice cloner requiring short reference audio for speaker adaptation, with an optimal reference length around 0.5-1 second.

7 Discussion

Significance of Results: The loss WER achieved on generated speech transcription exceeded our expectations, though this metric reflects transcription quality rather than perceptual naturalness. More critically, the 100% inference success rate on medium-scale models provides a reliable foundation for production deployment.

Limitations and Risks: Several limitations constrain interpretation of our findings:

1. **No perceptual evaluation:** We lack human listening tests (MOS scores) to validate whether the generated speech sounds natural to native Arabic speakers. WER and technical metrics only partially capture quality.
2. **Single-speaker bias:** The best-performing model (neutts-arabic-full) was trained primarily on one speaker (Adel), limiting generalization to diverse voices.
3. **Domain specificity:** All models were trained on podcast/audiobook data, which may not transfer well to other conversational contexts (e.g., customer service, virtual assistants).
4. **Incomplete hybrid evaluation:** The hybrid DeltaNet architecture could not be fully evaluated due to incomplete training (2K/5K steps), leaving open questions about memory-efficiency trade-offs.

Sensitivity to Design Choices: Our ablation studies revealed high sensitivity to several design decisions:

- **Dataset domain:** Podcast data (conversational) worked well for short-phrase synthesis, while audiobook data (narration) caused over-generation.
- **Reference audio length:** Performance varied dramatically between 0.5s (optimal), 7s (over-mimicking), and no reference (poor quality).
- **Dataset scale:** Medium-scale (3,668 samples) outperformed large-scale (697K samples), indicating diminishing or negative returns beyond a quality threshold.

These sensitivities suggest that practitioners should prioritize curating domain-matched, high-quality datasets over simply maximizing dataset size.

8 Future Directions

Based on our results and identified limitations, we propose the following directions for future work:

1. Perceptual Evaluation and Human Studies Conduct formal listening tests with native Arabic speakers to evaluate:

- Mean Opinion Score (MOS) for naturalness (1-5 scale)
- Speaker similarity ratings for voice cloning quality
- Preference tests comparing our models to commercial Arabic TTS systems

2. Hybrid Architecture Completion Complete training of the hybrid DeltaNet model to 5K+ steps and conduct comprehensive comparison against standard attention

3. Multi-Speaker Generalization Extend the medium-scale approach to diverse speakers:

- Collect/curate 3-5K samples from 10+ Arabic speakers (male/female, different dialects)
- Test zero-shot voice cloning on unseen speakers
- Evaluate cross-dialect generalization.

4. Reinforcement Learning Alignment As originally proposed, implement RL-based fine-tuning to optimize perceptual quality. We can also incorporate the feedback from the peer reviews:

- Design reward function combining WER, prosody metrics, and MOS proxy
- Test PPO vs. GRPO algorithms for stability
- Measure perceptual improvements vs. computational cost

5. Production Deployment Optimization Optimize the model for real-world use:

- Model quantization and distillation for faster inference
- Streaming inference for real-time applications
- Edge deployment feasibility testing

6. ASR–TTS Pipeline Integration

- Utilize AudioLabeler's transcriptions and timestamps to preprocess audio for TTS.
- Utilize alignment data to help refine prosody and duration modeling.

7. Multi-Dimensional Audio Labeling

- Include gender diagnostics for speaker-adapted TTS.
- Label environmental sounds utilizing models such as CLAP or similar.
- Identify emotional tone for creating expressive TTS output.
- Also, generate JSON output containing all labels for future processes.

8. Transcription Ensemble and TTS Quality

- Evaluate whether judge-processed transcriptions improve TTS naturalness despite higher WER
- Experiment with alternative ensemble methods (e.g., confidence-weighted ROVER, selective judge correction)

9 Conclusion

This work presents a comprehensive study of Arabic text-to-speech synthesis using the NeuTTS-Air architecture. We successfully trained multiple models across three dataset sizes, conducted extensive inference testing generating 68+ audio samples, and performed ablation studies comparing standard attention against hybrid DeltaNet architectures.

Key Achievements:

- Developed Arabic TTS models!
- Achieved low WER on generated speech transcription.
- Identified failures (over-generation, generation cutoff) :(
- Conducted ablation studies revealing high sensitivity to dataset domain, reference audio length, and scale.

Impact on Objectives: Our original objectives focused on three challenges in Arabic TTS: improving rhythm/naturalness, eliminating artifacts, and enhancing emotional control. While we successfully addressed correctness with WER, we did not implement the proposed RL-based alignment for emotional control, or checking prosody due to time constraints.

Future work should prioritize perceptual evaluation through human listening tests, completion of the hybrid DeltaNet architecture experiments, and implementation of the proposed RL-based alignment to fully realize the vision of contextually-aware, emotionally expressive Arabic speech synthesis.

10 Administrative

10.1 Team Contributions

- **Dareen Safar B Alharthi** (Mentor): Project direction and coordination, technical guidance on Arabic language processing, oversight of training experiments
- **Xiaohe Bai**:
  - Unified the Arabic ASR and labeling code into a single AudioLabeler module and extended it to a multilingual English–Arabic design.
  - Implemented language-specific prompting and cleaning utilities and added the final consolidated transcript output.
  - Added Whisper word-level timestamp extraction and designed the alignment JSON schema for downstream TTS/segmentation use.
  - Developed two evaluation scripts (LibriSpeech test-clean and Arabic Speech Corpus) to run the full pipeline and compute WER/CER and error-type rates across ASR variants.

  - Refactored GPU/CPU resource flows (model unloading, cache clearing) to eliminate OOM failures and stabilize the multilingual pipeline.
- **Bhuvan Koduru**:
  - Dataset encoding pipeline and integration using NeuCodec (238 samples → 697K samples)
  - Training execution and monitoring (7 models, 150K+ steps total)
  - Comprehensive inference testing framework (68+ audio samples generated)
  - Training configurations and hyperparameter tuning
  - Error analysis and failure mode investigation
  - DeltaNet architecture research and implementation selection (FLA vs. Raschka comparison)
  - Hybrid DeltaNet integration (75% DeltaNet + 25% Attention)
  - Results analysis and visualization
- **Fernando Ruiloba Portilla**:
  - Training curve analysis and checkpoint evaluation
  - Training data encoding
  - Baseline metrics pipeline
  - Data cleaning and normalization
  - Word Error Rate calculation
  - Cosine similarity calculation
  - Whisper and NeuTTS inference

Equal Contribution: All team members contributed approximately equally to the project, with overlapping responsibilities in experimentation, analysis, and documentation.

10.2 Code Repository

All the custom code and scripts we built for this project are stored in our Github repository.

`https://github.com/cmu-11785-project/mlsp-csm-project`

References

[1] Radford, A., Kim, J.W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2022). Robust Speech Recognition via Large-Scale Weak Supervision. *arXiv preprint arXiv:2212.04356*. `https://arxiv.org/abs/2212.04356`

[2] Inworld AI. (2025). Inworld TTS-1 Technical Report. *arXiv preprint arXiv:2507.21138*. `https://arxiv.org/abs/2507.21138`

[3] Lajszczak, M., Cambria, E., et al. (2024). BASE TTS: Lessons from building a billion-parameter Text-to-Speech model on 100K hours of data. *arXiv preprint arXiv:2402.08093*. `https://arxiv.org/pdf/2402.08093`

[4] Du, C., Huang, R., et al. (2025). CosyVoice 3: Towards In-the-wild Speech Generation via Scaling-up and Post-training. *arXiv preprint arXiv:2505.17589*. `https://arxiv.org/pdf/2505.17589`

[5] Step Audio Team. (2025). Step-Audio 2 Technical Report. *arXiv preprint arXiv:2507.16632*. `https://arxiv.org/pdf/2507.16632`

[6] Neuphonic. (2025). NeuTTS-Air: Neural Text-to-Speech with Attention and Incremental Refinement. GitHub repository. `https://github.com/neurt-tts/neu-tts-air`

[7] Yang, D., Li, H., et al. (2024). DeltaNet: Conditional Attention for Improved Speech Synthesis. *Proceedings of ICASSP 2024*.

[8] Piczak, K. J. (2015). ESC: Dataset for Environmental Sound Classification. *Proceedings of the 23rd ACM international conference on Multimedia*, pp. 1015-1018.

[9] Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is All You Need. *Advances in Neural Information Processing Systems*, 30.

[10] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*.

[11] Shen, J., Pang, R., Weiss, R. J., et al. (2018). Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4779-4783.

[12] Kong, J., Kim, J., & Bae, J. (2020). HiFi-GAN: Generative Adversarial Networks for Efficient and High Fidelity Speech Synthesis. *Advances in Neural Information Processing Systems*, 33, pp. 17022-17033.

[13] Panayotov, V., Chen, G., Povey, D., & Khudanpur, S. (2015). LibriSpeech: An ASR Corpus Based on Public Domain Audio Books. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206-5210.

[14] Wu, Y., Chen, K., Zhang, T., et al. (2023). Large-scale Contrastive Language-Audio Pretraining with Feature Fusion and Keyword-to-Caption Augmentation. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

[15] NVIDIA. (2023). SortFormer: Speaker Diarization with Transformers. NeMo Toolkit Documentation. `https://docs.nvidia.com/deeplearning/nemo/`