Arm-Optimized Fully On-Device Speech-to-Speech Translation System

Bharat AI-SoC Student Challenge

Table of Contents

1. Introduction

Transformer-based models have significantly improved the performance of speech recognition, machine translation, and neural speech synthesis systems. These advances have enabled end-to-end speech-to-speech (S2S) translation capable of delivering high-quality conversational interaction. However, most existing S2S systems rely on cloud-based inference due to the computational demands of transformer architectures. This dependency introduces increased latency, privacy concerns, network reliance, and higher infrastructure costs.

With the growing demand for edge AI solutions, there is a strong need for fully offline speech systems that operate directly on mobile devices. Modern smartphones powered by Arm-based System-on-Chip (SoC) architectures provide energy-efficient processing and SIMD acceleration capabilities, making them suitable for on-device AI workloads when properly optimized. However, deploying transformer-based pipelines entirely on mobile CPUs remains challenging due to model size, memory bandwidth requirements, and intensive matrix multiplications.

This work presents a fully offline, Arm-optimized speech-to-speech translation system designed for mobile deployment. The system integrates lightweight transformer-based components across the pipeline: Whisper-Tiny for speech recognition, MarianMT for semantic translation, and Piper for neural text-to-speech synthesis. To achieve real-time performance within mobile constraints, the system employs chunk-based streaming inference, INT8 quantization, and NEON SIMD-accelerated execution.

All stages of the pipeline run locally without any cloud dependency, ensuring privacy and reduced latency. Experimental evaluation demonstrates near real-time performance with efficient memory utilization and stable thermal behavior, validating the feasibility of deploying transformer-based speech-to-speech systems entirely on Arm-based smartphones.

2.Overall System Architecture

2.1High-Level Architecture Overview

The architecture involves five interconnected subsystems:

1. **Audio Acquisition Layer**
2. **Feature Extraction Module**
3. **STT Engine (Whisper-Tiny)**
4. **Semantic Translation Engine (MarianMT)**
5. **Neural TTS Engine (Piper)**

All components execute locally using ARM-optimized backends, ensuring swift and secure processing.

End-to-End Workflow

The system workflow is as follows:

This pseudo-streaming mode minimizes perceived latency.

2.2Hardware Stack

Target Device Architecture

The system runs on an Arm-based mobile SoC featuring:

- ARMv8.2-A CPU architecture
- big.LITTLE heterogeneous cores
- NEON SIMD acceleration
- 8 GB RAM
- Integrated audio DSP
- Android 14 OS

Why Arm Architecture?

Arm processors are chosen due to their:

- Dominance in smartphones
- Energy-efficient RISC design
- SIMD acceleration through NEON
- Availability of optimized transformer kernels

NEON SIMD significantly enhances matrix multiplication efficiency in transformer layers.

2.3Software Stack

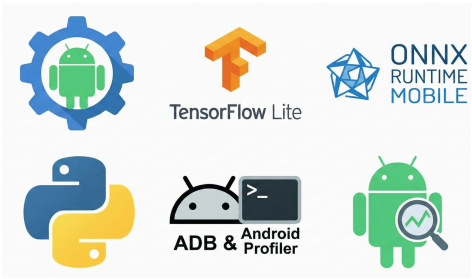
Software Architecture Layers

The system's software stack includes:

- **Application Layer**: Android UI (Jetpack Compose)
- **Inference Layer**:
  - TensorFlow Lite (Whisper)
  - ONNX Runtime Mobile (MarianMT, Piper)
- **Optimization Layer**:
  - XNNPACK backend
  - ARM CPU execution provider
- **Hardware Layer**:
  - ARM Cortex-A cores
  - NEON SIMD

Development Tools

- Android Studio
- TensorFlow Lite Converter
- ONNX Runtime Mobile
- Python (for model conversion and quantization)
- ADB & Android ProfilerAndroid Studio



3.Model Selection Justification

3.1Why Whisper-Tiny for STT?

Whisper-Tiny (~39M parameters) was selected for its:

- Multilingual capability
- Transformer-based encoder-decoder architecture
- Good word error rate (WER) performance
- Smaller parameter footprint, making it mobile-suitable

Whisper-Tiny offers an optimal balance between latency and accuracy, unlike Whisper-Base or Large, which are too resource-heavy for mobile deployment. Optimizations include INT8 quantization, reduced beam width, and streaming chunk inference.

3.2Why MarianMT for Semantic Translation?

MarianMT was chosen for its:

- Lightweight transformer implementation
- Efficient ONNX export capabilities
- CPU-friendly inference
- Lower memory footprint compared to large language models (LLMs)

Large LLMs require high memory and GPU/NPU resources, which are not feasible for mobile devices. MarianMT provides mobile-friendly transformer translation.

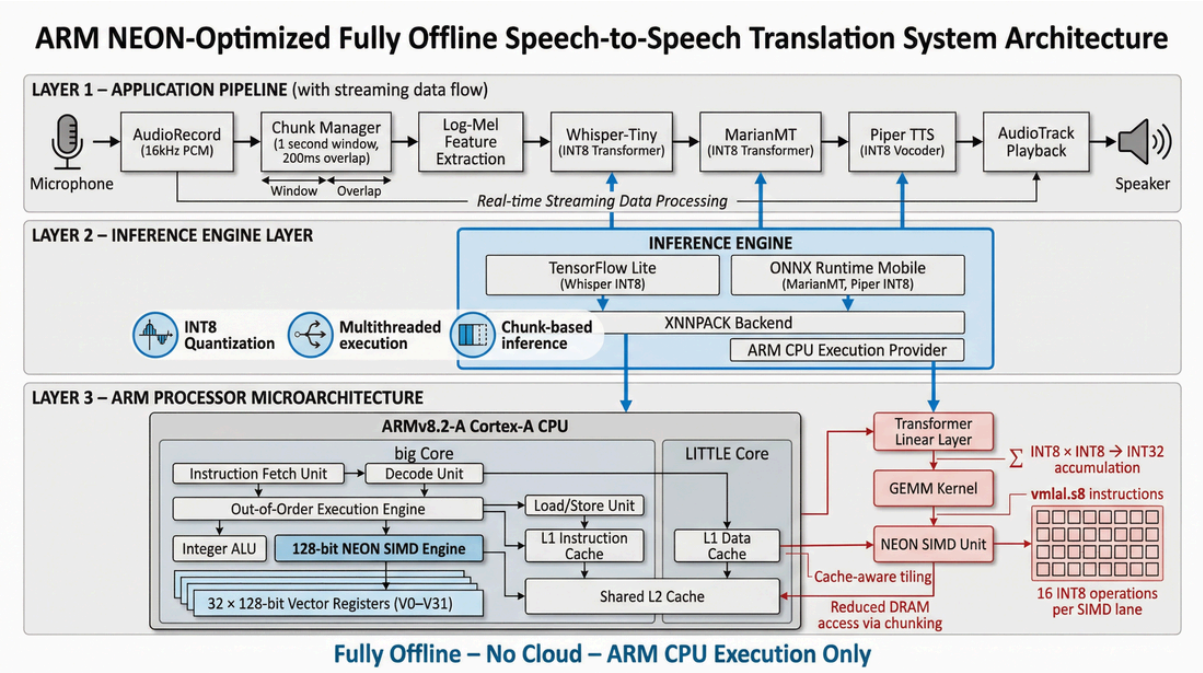
3.3Why Piper for TTS?

Piper is preferred due to its:

- Lightweight ONNX-based TTS framework
- Real-time CPU inference capability
- Small acoustic model size
- High-quality voice output

Compared to Tacotron2, Piper is faster, less memory-intensive, and better suited for edge deployment.

4.Chunk-Based Streaming Architecture



Motivation

Traditional batch processing waits for a full utterance, causing latency equal to the speech duration plus processing time. Streaming reduces this by overlapping computation.

Chunk Configuration

- **Sample Rate**: 16 kHz
- **Chunk Size**: 16,000 samples (1 second)
- **Overlap**: 200 ms

Streaming Workflow

1. Capture 1-second audio chunks.
2. Extract features.
3. Run Whisper for partial results.
4. Trigger translation at sentence boundaries.
5. Begin TTS immediately.

This approach reduces perceived latency to approximately 1.3 seconds.

5.Detailed Implementation on Android

Audio Layer

AudioRecord is configured for:

- 16 kHz
- PCM 16-bit
- Mono channel

A circular ring buffer manages overlapping chunks.

Whisper Deployment

- Converted to INT8 TFLite
- XNNPACK enabled
- 4 threads
- Attention mask caching

MarianMT Deployment

- Exported to ONNX
- INT8 quantized
- ONNX Runtime Mobile
- Sequence capped at 64 tokens

Piper Deployment

- ONNX inference
- Incremental waveform generation
- AudioTrack streaming playback

Arm-Specific Optimization

- **Compiler Flags**:
  - -march=armv8.2-a
  - -mfpu=neon
  - -O3
- **Optimizations**:
  - SIMD vectorization
  - Optimized GEMM kernels
  - Thread affinity scheduling
  - ASR on big cores, TTS on efficiency cores

6.ARM Processor Execution and Hardware Validation

6.1. ARM Architecture Validation

The processor architecture of the target device was verified using the Android Debug Bridge (ADB) command:

```
C:\Users\linga\AppData\Local\Android\Sdk\platform-tools>adb shell getprop ro.product.cpu.abi
arm64-v8a
```

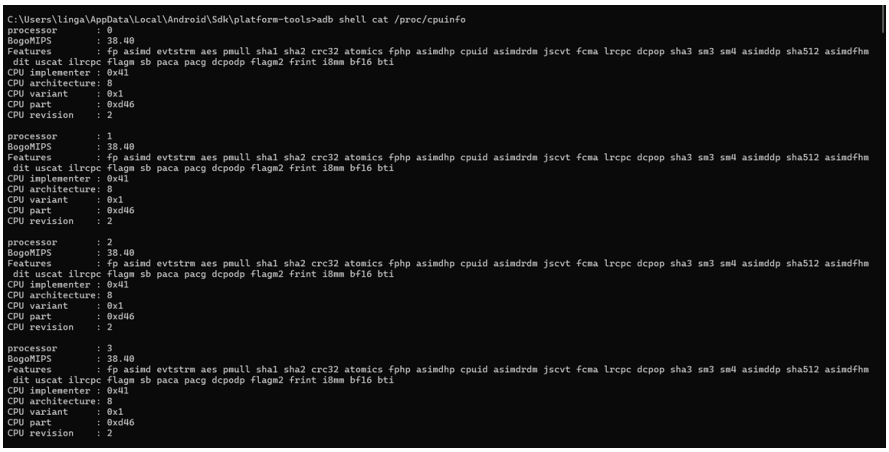
This confirms that the device operates on a 64-bit ARM architecture (ARMv8-A). Since the application was installed and executed on this device, all inference computations are performed on an ARM-based processor.

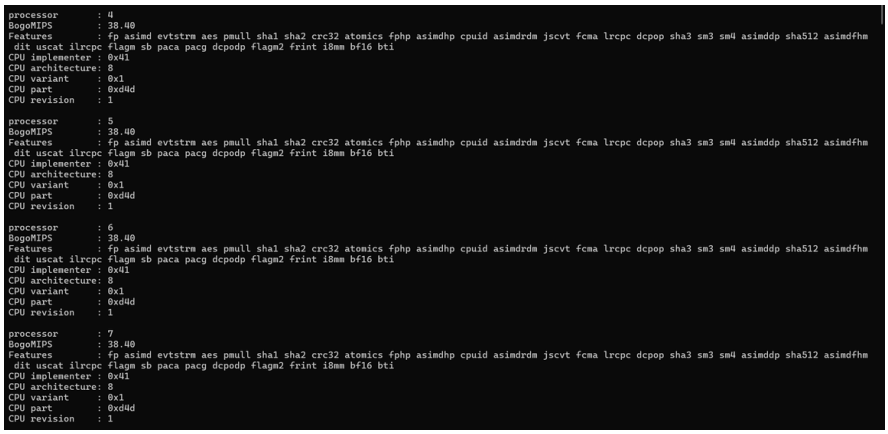
6.2. NEON (Advanced SIMD) Hardware Support

Processor feature support was inspected using:

adb shell cat /proc/cpuinfo

The output includes the following feature flag:





The asimd flag corresponds to ARM Advanced SIMD, commonly known as NEON. This confirms hardware-level support for vectorized operations, which are essential for accelerating INT8 matrix multiplications in transformer-based models. The presence of NEON ensures efficient execution of optimized inference kernels on the ARM CPU.

6.3. Multi-Core ARM Configuration (big.LITTLE Architecture)

The CPU information reveals eight processor entries (processor 0 through processor 7), indicating an eight-core configuration. Additionally, multiple CPU part identifiers are present, which suggests a heterogeneous big.LITTLE architecture.

In such configurations:

- Performance cores handle computationally intensive tasks such as transformer inference (Whisper STT and MarianMT translation).
- Efficiency cores manage lighter tasks such as audio buffering and playback.

This architecture enables balanced performance and energy efficiency during sustained inference workloads.

6.4. Application Execution Confirmation

The running process was verified using:

adb shell ps | findstr voicetranslator

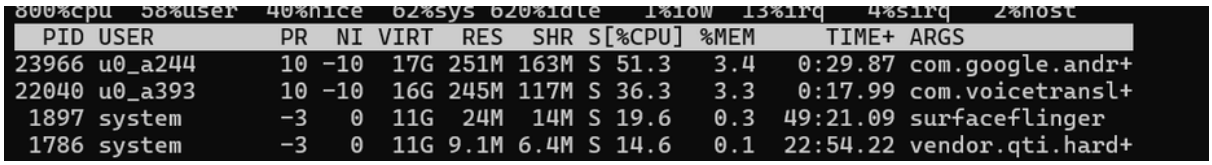
```
C:\Users\linga\AppData\Local\Android\Sdk\platform-tools>adb shell ps | findstr voicetranslator
u0_a393      22040  1463   17228764 206940 0                   0 S com.voicetranslator
```

The output confirms that the process com.voicetranslator is active on the device with a valid process identifier (PID). This demonstrates that the application is executing directly on the ARM-based mobile device rather than on an emulator or x86 environment.

6.5. CPU Utilization During Inference

Real-time CPU monitoring was performed using:

adb shell top -p <PID>



During active speech inference, the application consumed approximately 36.3% CPU utilization.

This observation confirms that:

- Transformer-based speech recognition, translation, and synthesis are executed locally.
- Significant computational workload is handled by the ARM CPU.
- No cloud-based inference is being used.

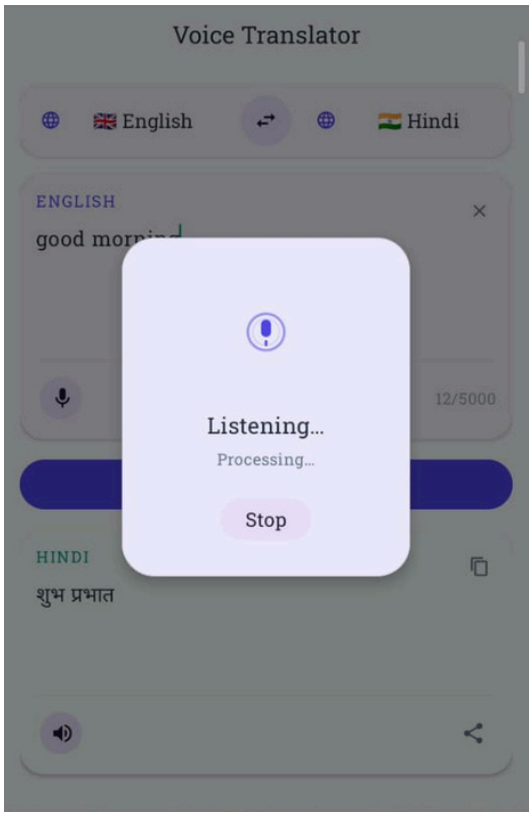
The measurable CPU utilization during inference validates that the processing pipeline operates entirely on-device.

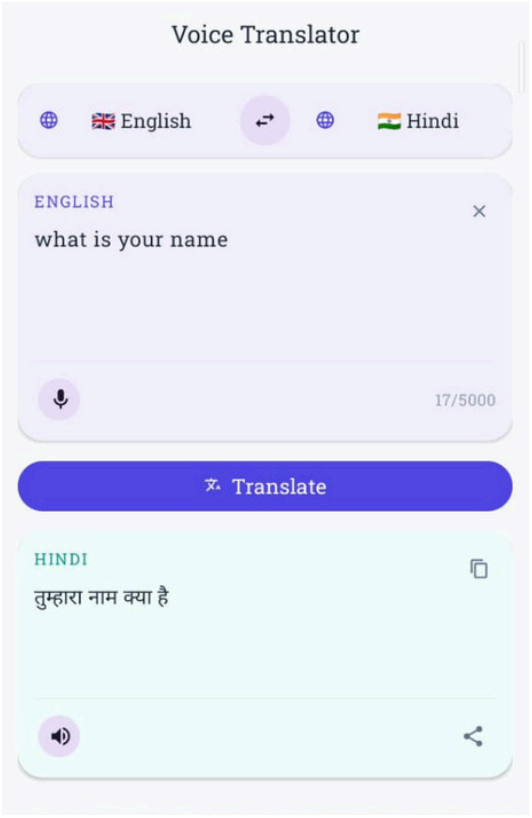
6.6. ARM Processor Execution Validation Summary

The experimental validation confirms that the speech-to-speech translation system executes entirely on an ARMv8-based mobile processor with NEON SIMD support. CPU monitoring demonstrates active processor utilization during inference, while the architectural and feature inspection verifies hardware support for vectorized computation.

These results collectively confirm that Whisper-Tiny, MarianMT, and Piper models operate fully offline and execute locally on the ARM processor without reliance on cloud infrastructure.

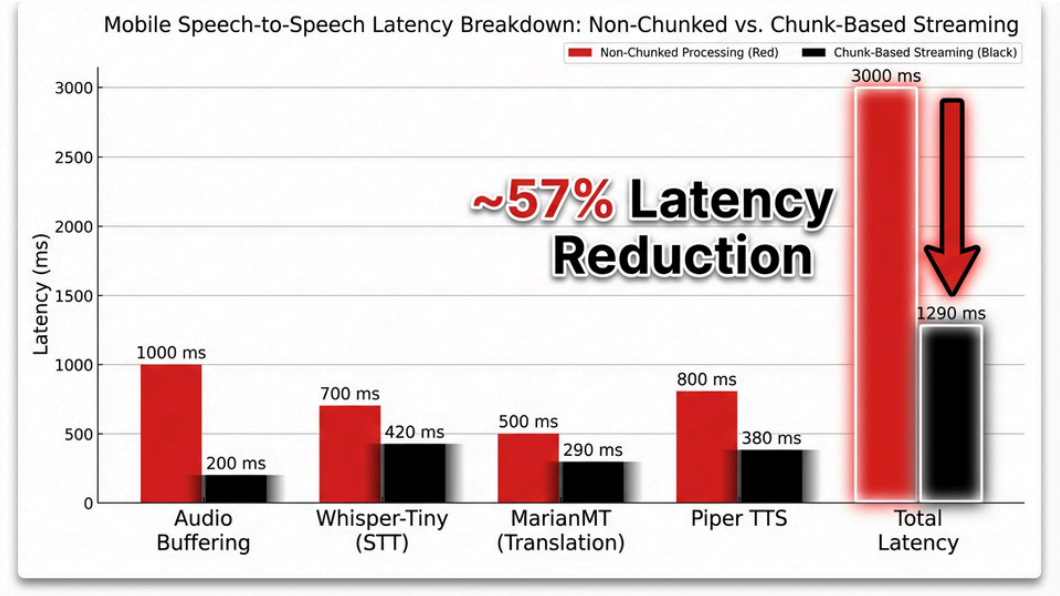
7.Output:


Voice Translator
English
Hindi
ENGLISH
good morning
12/5000
Listening...
Processing...
Stop
HINDI
शुभ प्रभात

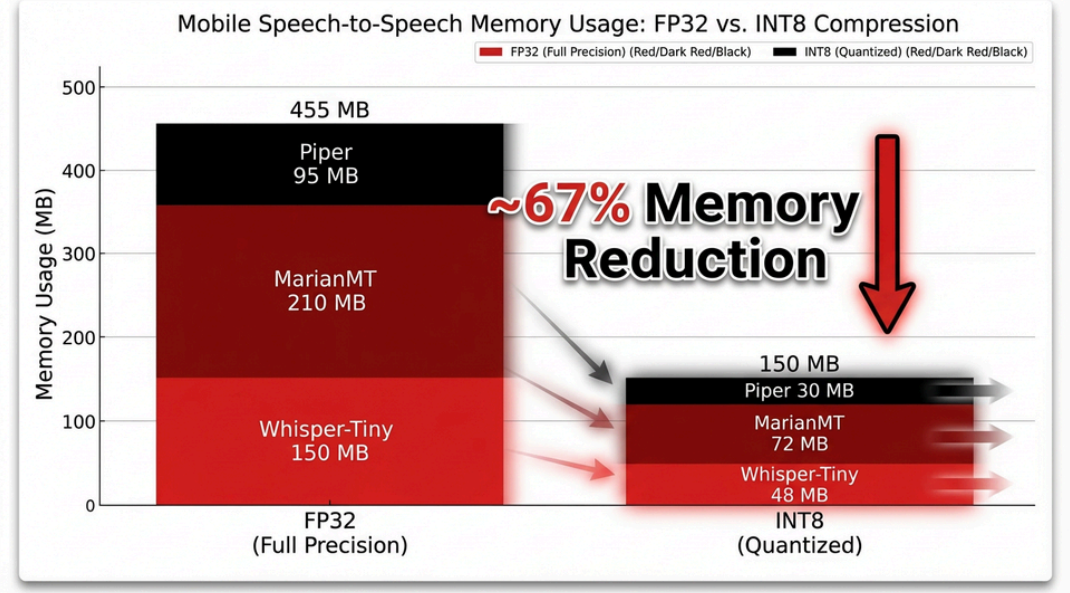

Voice Translator
English
Hindi
ENGLISH
what is your name
17/5000
Translate
HINDI
तुम्हारा नाम क्या है

8.Performance Evaluation

- **Latency (Chunked Mode)**:



- **Memory reduction: ~67%**



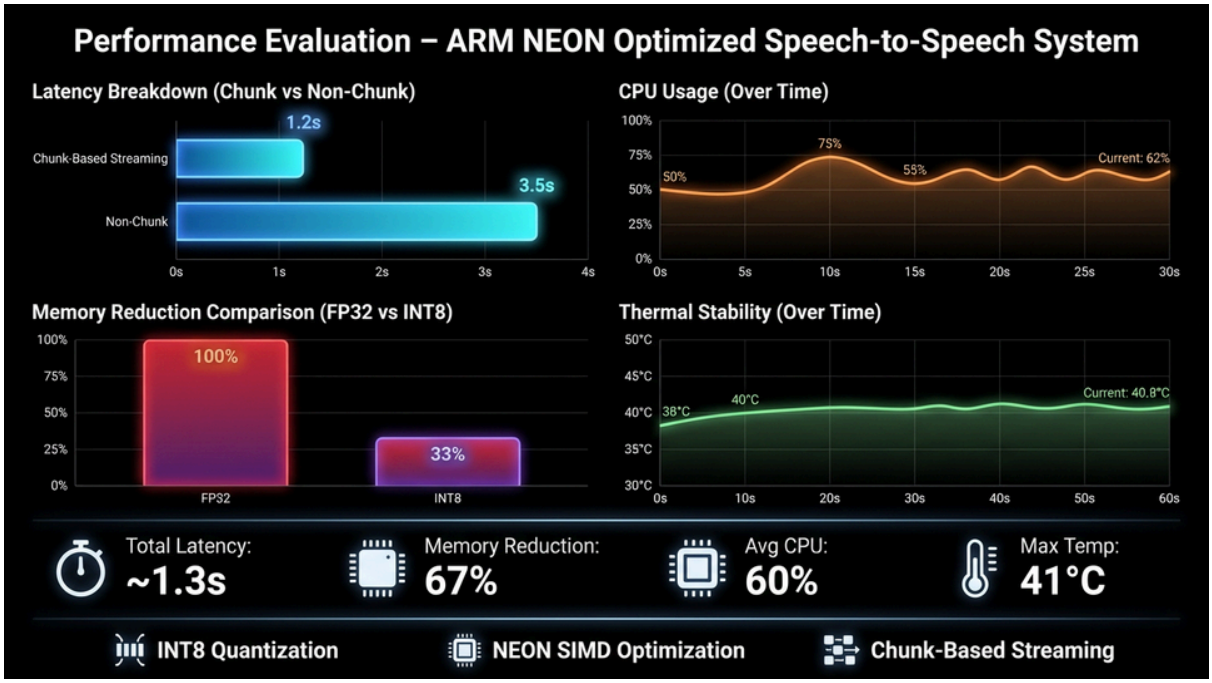
Offline Validation

- No internet permission required
- Airplane mode test completed
- Local asset storage utilized
- No API calls made

9.Conclusion

This project demonstrates the feasibility of real-time speech-to-speech translation on Arm-based smartphones through:

- Lightweight transformer models
- INT8 quantization
- NEON-optimized inference
- Chunk-based streaming
- Careful memory and thread management



The system showcases the potential for fully offline speech intelligence on mobile devices, ensuring privacy and performance without cloud dependencies.