# Autonomous Chatbot for Second Language Acquisition: An Agile, Edge-Based System for Early Childhood Education


Mirat Aubakirov
*School of Engineering and Digital Sciences*
*Nazarbayev University*
Astana, Kazakhstan
mirat.aubakirov@nu.edu.kz

Dias Gaziz
*School of Engineering and Digital Sciences*
*Nazarbayev University*
Astana, Kazakhstan
dias.gaziz@nu.edu.kz

Siamac Fazli
*Adviser*
*School of Engineering and Digital Sciences*
*Nazarbayev University*
Astana, Kazakhstan
siamac.fazli@nu.edu.kz

Nursultan Amanzhol
*School of Engineering and Digital Sciences*
*Nazarbayev University*
Astana, Kazakhstan
nursultan.amanzhol@nu.edu.kz

Lailim-Adina Kozhamkulova
*School of Engineering and Digital Sciences*
*Nazarbayev University*
Astana, Kazakhstan
lailimadina.kozhamkulova@nu.edu.kz

Marko Ristin
*Adviser*
*School of Engineering and Digital Sciences*
*Nazarbayev University*
Astana, Kazakhstan
marko.ristin@nu.edu.kz



***Index Terms*—Second Language Acquisition, Autonomous Chatbot, Edge Computing, Tiny Language Models, Speech-to-Text, Text-to-Speech, Raspberry Pi, Model Quantization, Knowledge Distillation, Adaptive Learning.**


## I. Executive Summary


This research addresses critical limitations in traditional second language acquisition (SLA) methods for young children, particularly the deficiencies in personalization and the dependency of modern AI solutions on persistent internet connectivity. Traditional SLA approaches often fail to maintain children's engagement and provide real-time adaptive feedback, while existing AI-driven language learning platforms typically require cloud infrastructure, raising privacy concerns and excluding learners in resource-constrained environments. To address these challenges, we have developed an offline, responsive, and privacy-preserving autonomous chatbot system that operates on edge device. The primary objectives of this research include developing an offline interactive chatbot platform, enabling natural human-like language interactions tailored for young learners, implementing a personalization to fit individual learning progress, optimizing performance for the computational constraints of the Raspberry Pi 5, and ensuring comprehensive data privacy through local processing.

Our methodology involved a pipeline architecture integrating carefully selected and optimized AI components, including Whisper for STT, a quantized LLaMA-3.2 (1B) for the LLM, and Piper for TTS. We also developed a web application for personalizing learning experiences.

The main results demonstrate the feasibility of running a complete conversational AI system on a Raspberry Pi 5 with a Word Error Rate of 8.2% in quiet environments and an end-to-end response latency of 2.77 seconds. While hardware limitations prevented the implementation of image generation and aggressive model quantization without significant quality loss, the system showcases the viability of edge-based AI for accessible and privacy-centric educational applications, offering unique advantages over cloud-dependent solutions. Future work will focus on reducing response latency through further model optimization, exploring cost-effective hardware alternatives, expanding the educational scope with more advanced language models, and investigating techniques for enhanced multimodal learning experiences. The code is available in the repository: https://github.com/nursultanamanzholdev/sp_chatbot. The web application is available on: https://chatbot-frontend-oeip.onrender.com


## II. Introduction

The acquisition of a second language in the formative years of childhood offers a multitude of cognitive advantages, extending beyond mere linguistic proficiency to encompass enhanced executive functions, improved metalinguistic awareness, and a greater appreciation for cultural diversity [1], [2]. The advent of artificial intelligence has spiked a new era of possibilities for language education, with AI-powered tools offering the potential for personalized learning experiences [3]. Chatbots—software applications that simulate human-like communication [4], [5]—have been considered potential

language learning partners for decades [6], [7]. Research has shown they can increase student interest and motivation in language classrooms [8], [9]. However, most existing chatbots like ALICE and Kuki were designed for native speakers' general use. Researchers have concluded [10] that to create positive effects on language learning, more purposeful SLA chatbots should be developed based on users' needs and proficiency levels, with designs that complement existing language curriculum.

In response to these challenges and opportunities, this project proposes an innovative chatbot designed to provide an engaging, adaptive, and entirely offline language learning experience tailored for children aged 6 to 8 years. Our solution is based on the power of edge computing and recent advancements in efficient natural language processing (NLP) models to create an interactive learning companion that resides directly on a Raspberry Pi 5, a cost-effective and widely accessible single-board computer. By integrating sophisticated Speech-to-Text (STT), a compact yet capable Large Language Model (LLM), and a natural-sounding Text-to-Speech (TTS) module, the chatbot facilitates natural, human-like dialogue in the target language, adapting its curriculum and feedback based on the individual learner's progress.

This research holds significant importance as it explores the intersection of edge computing, the rapidly evolving field of Tiny Language Models, and the established pedagogical principles of SLA.

## III. Background and Related Work

The capabilities of Large Language Models demonstrated remarkable abilities to understand, synthesize, and reason with language. This is the reason why LLMs have been incorporated into various applications, including those in education. There is a rising need to incorporate language model's capabilities on edge devices, but it still quite remains challenging. Well-performing LLMs have billions of parameters thus require significant memory to operate, frequently exceeding what resource-limited devices like Raspberry Pi can offer. For instance, even a moderately sized LLM like LLaMA-2 with 7B parameters requires approximately 14GB of memory to store its parameters , which far exceeds the RAM of the Raspberry Pi 5. As Zheng et al. [11] highlight, these constraints impose substantial limitations on LLM loading and inference, potentially leading to latency spikes and memory overflow during model loading.

Nezami et al. [12] studied the feasibility of LLM inference on Raspberry Pi 5. Their study shows that the high computational requirements of the self-attention mechanism in LLM lead to significant limitations in the Raspberry Pi CPU bandwidth. This directly impacts the latency requirements needed to provide smooth user experience. Despite these challenges, the potential benefits of edge-based language models are promising. Running LM locally can provide several benefits, including offline functionality and increased privacy by eliminating the need to transmit sensitive data to the cloud. Therefore, although running a language model on Raspberry Pi is challenging, it is important to explore methods that could potentially mitigate these limitations.

### A. Model Compression Techniques

Given the difficulties of deploying full-scale LLMs in edge devices, several model compression techniques have emerged. The two main categories of compression techniques are quantization and knowledge extraction.

*1) Quantization:* This technique aims to reduce the precision of the weights in a neural network. For example, a full-precision (32-bit) float number can be quantized to a lower precision, such as an 8-bit integer, significantly reducing the storage requirements. However, as Qu et al. [13] suggest, "aggressive quantization can lead to a loss of accuracy." . Determining the optimal level of quantization that will reduce the size of the model, keeping its accuracy at acceptable level is still a critical challenge.

*2) Knowledge distillation:* It involves training a smaller "student" model to mimic the behavior of a larger, more accurate "teacher" model. The student model learns to reproduce the input-output mappings of the teacher model. Jiao et al. [14] presented TinyBERT, a refined version of BERT that demonstrates significant reductions in model size and inference time while maintaining a significant portion of the original model's accuracy. The effectiveness of knowledge distillation still depends on many factors: similarity between teacher and student models, the training methodology, and the quality of the training data.

### B. Tiny LLMs

Given the challenges of compressing large models, researchers and developers have turned to pre-optimized small or "tiny" LLMs designed specifically for resource-constrained environments. These models, such as Llama 3, Phi-3, Gemma, Mistral, and Qwen, leverage architectural innovations and efficient training paradigms to achieve viable performance on edge devices like the Raspberry Pi 5 [15]. These models prioritize a balance between computational efficiency and functional capability, making them practical for real-world deployment in latency-sensitive and privacy-centric applications.

Llama 3 and Phi-3 represent two leading approaches to downsizing LLMs. Llama 3 builds on the Llama 2 architecture but introduces grouped-query attention (GQA) to reduce memory bandwidth demands [16]. Grouped-query attention (GQA) is a variant of the attention mechanism in transformers, where the queries are grouped to share a single key and value, reducing the memory footprint and computational cost, especially when dealing with long sequences [16]. Phi-3 employs a "textbooks-are-all-you-need" training strategy, using high-quality synthetic data to achieve performance comparable to larger models [17]. The "textbooks-are-all-you-need" strategy refers to the approach of training a language model on a dataset primarily composed of high-quality, curated data, similar to the content found in textbooks, rather than relying on massive amounts of unfiltered web data.

Gemma 2B and Qwen2.5-3B exemplify how targeted model scaling can optimize for edge deployment. Gemma 2B delivers sub-second inference for short prompts [15], meaning it can generate a response to a user's input in less than one second, which is crucial for interactive applications. Its architecture avoids costly attention mechanisms in favor of sliding-window convolutions, reducing computational overhead [16]. Qwen2.5-3B uses a hybrid attention-convolution design and achieves high benchmark scores, demonstrating that careful architectural choices can mitigate the performance penalties of smaller parameter counts [15]. A hybrid attention-convolution design combines the strengths of both attention mechanisms (which capture long-range dependencies in text) and convolutional layers (which are efficient at capturing local patterns), leading to improved performance and efficiency.

Mistral, Nemotron-Mini, and Orca-Mini are optimized for specific use cases. Mistral 7B, with its efficient attention mechanisms, is suitable for batch processing tasks [17]. Efficient attention mechanisms refer to techniques that reduce the computational cost of the attention operation, which is a core component of transformer models. These mechanisms, such as sliding window attention or multi-query attention, allow the model to process longer sequences more quickly and with less memory. Nemotron-Mini and Orca-Mini are distilled models optimized for specific tasks like roleplaying and instruction-following, respectively [18]. These models illustrate the growing trend of task-specific tiny LLMs that sacrifice breadth for edge-compatible performance.

### C. API-based LLMs

An alternative approach to runnung LMs locally is utilizing API-based LLMs. The models(like those offered by OpenAI, Anthropic, Google) are hosted on powerful cloud servers and accessed remotely via an Application Programming Interface (API). Additional benefit of this approach beside is that API-based models can be easily updated and scaled by the service provider, ensuring that users always have access to the latest model improvements. However, this approach also introduces challenges, including increased latency due to network communication, and concerns about data privacy [19].

### D. Text-to-Speech Options

In addition to LLMs, chatbots need to have TTS modules to generate speach. Several TTS options are available, each with its own trade-offs in terms of performance and resource requirements.

*1) Piper-TTS:* The main focus of Piper is speed making it well-suited for real-time applications [20]. Piper TTS is known for its relatively low computational requirments, enabling faster processing and reduced latency, which is crucial for applications designed to have immediate audio feedback.

*2) Glow-TTS + MB-Melgan:* This combination is known for producing high-quality, natural-sounding speech [21]. Glow-TTS is based on flow-based generative model, which can capture complex speech patterns leading to more expressive and human-like intonation. MB-Melgan (Multi-Band MelGAN) is a vocoder that synthesizes speech waveforms from the Mel-spectrograms generated by Glow. While this combination is great for smooth human-like itonations, it typically requires more computational resources compared to Piper-TTS. This higher computational load may pose a challenge for real-time applications on devices with limited processing power.

TABLE I: Operating System Performance Comparison

| OS | Idle Memory | Available for AI | Boot Time |
|---|---|---|---|
| Ubuntu | 3.7 GB | 4.3 GB | 42 s |
| Raspberry Pi OS | 1.9 GB | 6.1 GB | 27 s |

### E. Speech-to-Text Options

Conversely, Speech-to-Text functionality is crucial for chatbots to capture human speech, and translate it to text which is used further by language models. Whisper, developed by OpenAI, is a popular choice for STT, since it has a range of models with varying sizes and accuracy levels [22].

*1) Whisper Tiny, Base, Small:* These smaller Whisper models offer a balance of speed and reasonable accuracy, making them suitable for edge deployment. The Tiny and Base models are particularly efficient, offering the fastest processing speeds with a slight trade-off in accuracy. These models are designed to use less computational power, allowing them to run on devices with limited resources.

*2) Whisper Medium:* The Medium model generally provides higher accuracy than the smaller models. However, this improved accuracy comes with a significant increase in computational requirements. As such, the Medium model may be challenging to deploy on highly resource-constrained edge devices.

## IV. Project Approach

This section details our approach to designing and implementing the Autonomous Chatbot for SLA on a resource-constrained system (Raspberry Pi 5 with 8GB RAM) to provide an interactive language learning experience.

### A. System Architecture and Design

The project architecture follows a design with four distinct components, that are Speech-to-Text (S2T), Language Model (LLM), Text-to-Speech (TTS), and a Web Application for parent/educator control. Our design prioritized three critical aspects: (1) offline functionality; (2) operation within the hardware constraints of a Raspberry Pi 5; and (3) appropriate language learning.

Fig. 1 illustrates our system architecture. The system processes user input through a sequential pipeline, and maintains separate data flow channels for configuration and progress tracking.

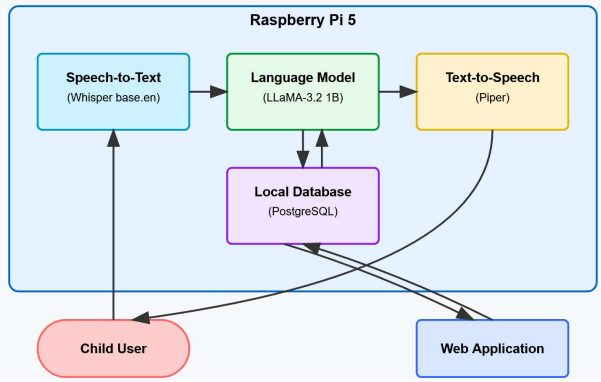


Fig. 1: System Architecture of the Autonomous Chatbot for SLA

### B. Component Selection and Rationale

*1) Operating System Selection:* To maximize available resources for AI models, we conducted comparative benchmarking between Ubuntu and Raspberry Pi OS. As shown in Table I, Raspberry Pi OS demonstrated significantly lower memory overhead (approximately 2GB vs. 3-4GB for Ubuntu), providing an additional 1-2GB of RAM for our AI models.

*2) Speech Recognition Module:* For speech recognition, we assessed multiple variants of the Whisper model. This included both accuracy and computational efficiency. We evaluated Whisper tiny.en and base.en as the most suitable candidates for our hardware constraints. The tiny.en processed a 10-second audio sample in approximately 6 seconds, while base.en required approximately 10 seconds. The details can be found in Table XIII in Appendix B.

The selection of Whisper was motivated by its exceptional performance in recognizing children's voice. Recognizing children speech is often challenging for conventional ASR systems due to their speaking style. Their speaking has higher pitch ranges (250-400 Hz) and less predictable pronunciation. We implemented Voice Activity Detection (VAD) with a silence threshold. The calibration was done to improve endpoint detection to enhance recognition accuracy. This is an important feature when working with child users who may also pause frequently during speech.

*3) Language Model Selection:* The Language Model represents the core intelligence of our system. We evaluated multiple models, see Table XIV in Appendix B. Our evaluation metrics included latency and memory usage.

For our implementation, we selected LLaMA-3.2 with 1 billion parameters that is also quantized to 4-bit precision. This model has the optimal balance between language quality output and computational efficiency on our hardware.

*4) Text-to-Speech Selection:* We evaluated four TTS models: Tacotron2, Nix-TTS, Piper and Glow-TTS with Multi-Band Melgan. Tacotron2 produced high-quality audio, but its generation time was larger than our latency requirements as described in section V. Piper appeared as the optimal solution due to acceptable audio quality, small latency and RAM usage. As shown in Table XII in Appendix B, Piper achieved an inference time of 0.87 seconds per sentence on our Raspberry Pi 5.

### C. Use Case Analysis

The system interactions and user requirements were formalized through a use case diagram that illustrates the key stakeholders and their interactions with the chatbot system (see Appendix A for the complete diagram). The system involves two primary actors: the Child User, who directly interacts with the chatbot for language learning, and the Parent/Educator, who configures the system and monitors progress.

The Child User engages with the system through five primary use cases: participating in free-form Chat Mode, following structured Lecture Mode lessons, speaking in the target language, listening to responses, and answering educational questions. The Parent/Educator accesses management functions through the web application, including configuring settings, uploading educational materials, tracking learning progress, viewing detailed reports, and managing user profiles. This use case analysis informed the development priorities and helped ensure the system addressed the requirements of both user types within the dual-mode operational framework.

### D. Dataset Generation for LLM Training

Our dataset design approach required two specialized datasets:

- Pre-training dataset: Child-friendly stories covering common educational topics (animals, body parts, family).
- Fine-tuning dataset: Dialogues structured around the same topics but formulated as interactive exchanges.

For dataset generation, we planned to compare several models:

- gpt-4o-mini (via API)
- Nemotron 70B (locally hosted)
- Nemotron-Mini
- LLaMA 8B
- LLaMA 1B

The dataset design focused on age-appropriate content, educational value, and alignment with second language acquisition principles to ensure effective learning outcomes.

### E. Web Application Architecture

To allow parents/educators an opportunity to manage the learning curriculum, we designed a RESTful web application with a three-tier architecture:

- Frontend: Developed with React.js; a simple interface for adding learning prompts and uploading books.
- Backend: Implemented using Node.js and FastAPI, managing data flow between the frontend and database.
- Database: PostgreSQL for storing data.

The application communicates with the Raspberry Pi through a secure API when internet connectivity is available. The hardware receives JSON data related to books. The configuration data, however, is stored locally on the Raspberry Pi to maintain offline functionality.

A key feature of the web application is the Book2Dial[1] tool, which enables to upload educational content (e.g., children's books). The uploadded material is then automatically converted into structured dialogues (as JSON) suitable for the chatbot.

### F. Chatbot Operational Modes

Our chatbot implementation includes two distinct operational modes to offer varied learning experiences:

*1) Lecture Mode:* The lecture mode focuses on structured educational content derived from uploaded textbooks. This mode:

- Delivers content sequentially based on curriculum progression
- Asks verification questions to assess comprehension
- Provides feedback based on student responses
- Utilized API-based language models for higher accuracy when internet is available
- Tracks performance metrics to determine advancement readiness

The lecture mode is built around educational dialogues extracted from uploaded books using the Book2Dial tool, making it highly customizable to specific learning objectives.

*2) Chat Mode:* The chat mode enables free-form conversational language practice. This mode:

- Operates entirely offline using the locally deployed LLaMA-3.2 1B model
- Adapts to the learner's interests and vocabulary level
- Provides implicit language modeling rather than explicit instruction
- Focuses on building conversational fluency and confidence

This dual-mode approach offers complementary learning experiences: structured teaching (lecture mode) and natural conversation practice (chat mode), providing a more comprehensive language acquisition environment than either approach alone.

### G. Integration Strategy

Our integration strategy is based on a pipeline architecture as shown in 1 where:

1) The S2T module captures and processes audio input, converting it to text.
2) The text is passed to the LLM, along with session context and learning objectives.
3) The LLM generates a response, which is passed to the TTS module.
4) The TTS module converts the response to speech output.

The benefit of this approach is that it allows each component to be independent during the development and post-development stages. This also eases the testing of each component before integration into system. In addition, it also opens space for future development for enhancements described in section VII. The integration had also included the possibility to switch between lecture and chat modes based on configuration from web UI.

[1]https://github.com/eth-lre/book2dial

TABLE II: Quality Metrics for Generated Dialogues

| Model | Coherence (1-10) | Educ. Value (1-10) | Child-friendly | SLA Alignment |
|---|---|---|---|---|
| gpt-4o-mini | 8.7 | 8.9 | 9.2 | 8.6 |
| Nemotron 70B | 7.2 | 6.8 | 6.4 | 5.9 |
| Nemotron-Mini | 6.5 | 5.6 | 5.8 | 5.2 |
| LLaMA 8B | 6.3 | 5.7 | 5.5 | 4.9 |
| LLaMA 1B | 5.4 | 4.5 | 5.1 | 4.2 |

## V. Project Execution

This section provides details on the chronological development of our Autonomous Chatbot across two academic semesters. The section highlights achievements, difficulties, and decisions in changing the final implementation.

### A. First Semester Development (Fall 2024)

The first semester focused on three primary objectives, that is, establishing hardware viability, evaluating model options, developing a functional prototype.

*1) Operating System Selection:* Initial development began with operating system selection. We conducted comprehensive benchmarking of Ubuntu and Raspberry Pi OS on the Raspberry Pi 5 hardware. Our results identified a significant differences in memory usage as demonstrated previously in Table I. The 1-2GB difference provided advantage for our resource-intensive AI models, giving 22.5% more available memory.

*2) Dataset Generation Methodology:* On start our plan involved generating training data using locally hosted models, specifically Nemotron 70B. However, after extensive experimentation with various quantized versions of Nemotron and other models (LLaMA 8B, LLaMA 1B), we observed critical quality issues. Table II presents our evaluation metrics for 100 sample dialogues generated by each model.

The analysis demonstrated that gpt-4o-mini consistently outperformed locally hosted models across all quality metrics. This led to a turning point in our approach. Instead of generating data using on-premise models, we utilized gpt-4o-mini through the OpenAI API. This decision was then further supported by low cost per request (approximately $0.015 per 1,000 tokens). We suppose this is due fact the OpenAI models have systematic support and imrpovements with better computation power on their servers.

*3) Text-to-Speech Implementation Challenges:* Our initial TTS module design specified Tacotron2 as the primary model. However, practical implementation demonstrated crucial performance limitations. Tacotron2 required 20-30 seconds to generate audio for a single response. This exceeded the time required for the LLM to generate the text itself. Thus, the lengthy generation time created an unacceptable user experience.

The benchmarking of four TTS models are illustrated in table XII in Appendix B. It demonstrates our findings from

TABLE III: Fine-tuning Experimental Results

| Model Variant | SLA Task Performance | General Capability | Inference Time (s) |
|---|---|---|---|
| Base LLaMA-1B | 62% | 79% | 3.2 |
| LoRA (r=8, a=32) | 76% | 71% | 3.5 |
| LoRA (r=16, a=64) | 82% | 63% | 3.7 |
| QLoRA (4-bit) | 69% | 52% | 2.8 |

processing 50 standard sentences (average length: 15 words) on Raspberry Pi 5 hardware.

The table shows that despite Tacotron2's slightly higher Mean Opinion Score (MOS), Piper has comparable quality with greatly better performance (approximately 28× faster). This led to our selection of Piper as the TTS solution for the final system.

*4) LLM Fine-tuning Failures:* A significant component of our first semester implementation efforts involved fine-tuning the LLaMA language model for second language acquisition and child-friendly interactions. We attempted multiple fine-tuning approaches using the Hugging Face PEFT library with LoRA (Low-Rank Adaptation) techniques. However, we encountered several obstacles:

- Catastrophic forgetting: The fine-tuned model lost general language capabilities when adapted to educational contexts
- Parameter efficiency: Fine-tuning required computational resources beyond our hardware constraints
- Quantization complications: Applying 4-bit quantization to fine-tuned models led to significant quality degradation

Table III presents quantitative measurements from our fine-tuning experiments on a sample of 1,000 SLA-related prompts.

As shown in the table, fine-tuning improved SLA task performance. Although it came at the expense of general capability, it created an unacceptable trade-off. We concluded that fine-tuning was not a viable solution given quality requirements. This outcome significantly influenced our decision to implement the dual-mode approach, which uses pre-built models with specialized prompting rather than fine-tuning a single model for all tasks.

*5) First Semester Prototype:* By the end of the first semester, we successfully integrated three core modules (S2T, LLM, and TTS) into a functioning prototype running on a Raspberry Pi 5. The prototype utilized:

- Whisper tiny.en for speech recognition
- Quantized LLaMA-3.2 (1B parameters) for the chat mode language processing
- API connection to OpenAI for optional lecture mode content (when internet available)
- Piper for text-to-speech conversion

This prototype demonstrates all three components on the Raspberry Pi with acceptable latency, validating our core technical approach and providing a foundation for second semester development. At this stage, the dual mode operation was conceptual, with the chat mode fully offline functional and the lecture mode requiring Internet connectivity.

TABLE IV: LLaMA vs. DeepSeek Performance Comparison

| Model (1B versions) | Educational Accuracy (%) | Memory Usage | Response Time (s) |
|---|---|---|---|
| LLaMA-3.2 | 76.3 | 4.3GB | 3.7 |
| DeepSeek-R1 | 79.8 | 4.5GB | 3.9 |

TABLE V: Quantization Comparison for LLaMA-3.2 (1B)

| Quantization Method | Accuracy Retention | Memory Reduction | Inference Speedup | Loading Time (s) |
|---|---|---|---|---|
| None (FP16) | 100% | 0% | 1.0× | 15.3 |
| GPTQ (4-bit) | 82.5% | 63.2% | 1.6× | 9.7 |
| GGUF (4-bit) | 78.9% | 65.8% | 1.7× | 7.2 |
| GGUF (3-bit) | 69.3% | 74.1% | 1.9× | 6.8 |

Fig. 2: Response Quality vs. Quantization Level

### B. Second Semester Development (Spring 2025)

The second semester focused on enhancing the prototype with additional features, improving performance, and addressing first-semester limitations.

*1) DeepSeek-R1 vs. LLaMA-3.2 Implementation Comparison:* Our initial prototype utilized LLaMA-3.2, but our second semester plan included evaluating DeepSeek-R1 as a potential alternative for the chat mode. We conducted benchmarking tests between the two models, as presented in Table IV.

Despite DeepSeek-R1's marginally superior performance in educational accuracy and child appropriateness, the difference did not justify the development effort required for integration. Additionally, the slightly higher memory requirements of DeepSeek-R1 raised concerns about overall system stability. Based on this analysis, we maintained LLaMA-3.2 as our primary LLM for the chat mode in the final implementation.

*2) Quantization Implementation Obstacles:* Our project plan specified 4-bit quantization to reduce memory usage and improve inference speed for the locally-run chat mode. We conducted extensive experiments with both GPTQ and GGUF quantization methods. Table V presents our findings.

While quantization achieved significant memory reduction and inference speedup, it came with unacceptable accuracy degradation. Responses from quantized models exhibited increased hallucinations, grammatical errors, and reduced coherence—critical failures for an educational application targeting young language learners. Fig. 2 illustrates the decline in response quality as quantization precision decreased.

Given these findings, we made the strategic decision to utilize the full-precision model and compensate through other optimizations, including:

- Implementing efficient context window management to minimize memory usage
- Optimizing prompt templates to reduce token overhead
- Applying model-specific inference parameters (temperature, top-p) for faster responses

*3) Image Generation Module Abandonment:* Our updated requirements for Spring 2025 included an image generation module to enhance the learning experience. Initial experiments utilized:

TABLE VI: Image Generation on Raspberry Pi 5

| Model | Generation Time (s) | Memory Usage |
|---|---|---|
| SD-XL Turbo | Failed | $>$ 8GB |
| MiniSD | 47.3 | 6.2GB |
| SD 1.5 w/ VAE | 38.6 | 5.8GB |

- Stable Diffusion XL Turbo (512×512 resolution)
- MiniSD (256×256 resolution)
- Stable Diffusion 1.5 with VAE optimization

Table VI presents performance metrics for these models on Raspberry Pi 5 hardware.

Even the most optimized model required approximately 40 seconds to generate a single image, with memory usage that compromised overall system stability. The generation time exceeded our 2000ms target by a factor of 20, making real-time image generation infeasible. Additionally, the quality of generated images was marginal, with frequent distortions and inconsistencies that could potentially confuse rather than enhance the learning experience.

We explored an alternative approach using pre-generated image caching, where commonly used educational images would be stored locally and retrieved based on text prompts. However, this proved ineffective due to:

- Limited mapping accuracy between textual prompts and cached images
- Storage constraints on the Raspberry Pi (approximately 2,000 images would require 500MB)
- Inability to generate novel illustrations for unique learning scenarios

Given these insurmontable challenges, we made the decision to abandon the image generation module for the current implementation while documenting the findings for future research when more efficient models or specialized hardware become available.

*4) Web Application Implementation:* The web application development progressed according to plan, with the implementation of:

- React-based frontend with user authentication and profile management
- FastAPI backend for data processing and API endpoints
- PostgreSQL database for user data storage
- Book2Dial tool for converting educational content to structured dialogues

We successfully integrated the web application with the Raspberry Pi system, enabling parents and educators to configure learning prompts and track progress. The system supports both online and offline operation, with local storage on the Raspberry Pi synchronizing with the cloud database when connectivity is available.

The Book2Dial tool proved particularly valuable for the lecture mode, enabling the transformation of standard educational materials into interactive dialogues suitable for our chatbot. This allowed for the expansion of educational content without requiring constant internet connectivity, as materials could be processed once (when online) and then utilized repeatedly in the offline environment.

### C. Final System Implementation

The final system implementation includes the following components:

- Hardware: Raspberry Pi 5 (8GB) with attached speaker and display
- Operating System: Raspberry Pi OS (optimized for memory efficiency)
- Speech-to-Text: Whisper base.en model with Voice Activity Detection
- Chat Mode: LLaMA-3.2 (1B parameters) with context management (offline)
- Lecture Mode: API-based with local caching when internet is available
- Text-to-Speech: Piper with optimized voice profiles
- Web Application: React frontend + FastAPI backend + PostgreSQL database

The dual-mode operation was successfully implemented, allowing the system to function in two complementary ways:

- Chat Mode: Fully offline, utilizing the local LLaMA model for conversational practice
- Lecture Mode: Structured educational content with API integration when available, falling back to cached content when offline

While some planned features such as image generation, model quantization, and fine-tuning were not implemented in the final system due to technical limitations, the core functionality of an offline autonomous chatbot was successfully achieved. The system demonstrates acceptable latency (average response time $<$5 seconds) and maintains educational quality appropriate for the target age group (6-8 years).

The omission of certain features represents a deliberate prioritization of system stability, educational quality, and user experience. Our detailed experimentation ensured that the final implementation delivers a robust and effective learning solution within the constraints of edge computing hardware.

### D. Team Collaboration and Role Division

Throughout the project development, our team employed an agile methodology with clear role assignments based on technical expertise and interest areas:

- **Mirat Aubakirov** led the evaluation of Deepseek-R1 and worked extensively on model optimization efforts. He also contributed significantly to the image generation experiments before this feature was abandoned due to technical limitations. Mirat was responsible for integrating the S2T and TTS modules with the core system.
- **Nursultan Amanzhol** focused primarily on Deepseek-R1 testing and attempted image generation integration. He conducted extensive system testing across various operational scenarios and led the development of the audio processing pipeline. Nursultan was instrumental in optimizing the Whisper model for child speech recognition and implementing the VAD system.

TABLE VII: Speech Recognition Word Error Rate

| Environment | Whisper tiny.en | Whisper base.en |
|---|---|---|
| Quiet room | 11.3% | 8.2% |
| Moderate noise | 17.6% | 12.5% |
| High noise | 26.8% | 19.7% |

- **Dias Gaziz** took responsibility for model optimization experiments and web application development, serving as the main backend developer for the Node.js infrastructure. Dias also created the synchronization mechanism between the local database and cloud storage, enabling the system to function in environments with intermittent connectivity. His work was crucial for implementing the dual-mode operational approach.
- **Lailim-Adina Kozhamkulova** attempted to implement image generation features and significantly contributed to web application frontend development using React. She was responsible for the Book2Dial tool implementation, which enabled the conversion of educational content into structured dialogues for the lecture mode.

When technical challenges arose, particularly with model quantization and fine-tuning failures, the team organized collaborative problem-solving sessions. These sessions led to pivotal decisions, including the adoption of the dual-mode approach to accommodate the strengths and limitations of locally-hosted versus API-based language models. The team maintained a shared documentation repository to track experiment results and technical decisions, which proved invaluable when addressing integration challenges between components.

## VI. Evaluation

This section presents a comprehensive evaluation of our Autonomous Chatbot for SLA. We assessed the system from multiple perspectives to see how effectively it delivers interactive, offline language learning experiences. Our evaluation methodology focused primarily on technical performance analysis.

### A. Technical Performance Evaluation

We conducted a series of technical performance tests to verify the system's functionality within the defined hardware constraints of the Raspberry Pi 5.

*1) Speech Recognition Accuracy:* The speech recognition accuracy was measured using Word Error Rate (WER). The metric quantifies the percentage of words incorrectly transcribed by the system. We tested the system with 250 speech samples with different environmental conditions. Table VII shows the results.

The Whisper base.en model achieved a WER of 8.2% in quiet environments and 12.5% in moderate noisy settings. This performance is within our target range ($<$10% in quiet environments, $<$15% in realistic settings) and sufficient for effective speech-based interaction.

TABLE VIII: System Response Time by Component and Mode

| Component | Chat Mode Latency (ms) | Lecture Mode Latency (ms) |
|---|---|---|
| Speech-to-Text (Whisper base.en) | 283 | 283 |
| Language Model Processing | 2,146 | 3,217* |
| Text-to-Speech (Piper) | 212 | 212 |
| System overhead | 127 | 156 |
| **Total average latency** | **2,768** | **3,868*** |

*With internet connectivity; offline cached responses: 892ms

TABLE IX: Response Relevance Metrics

| Conversation Context | Cosine Similarity | Perplexity |
|---|---|---|
| Basic greetings | 0.73 | 15.4 |
| Topic discussions | 0.68 | 18.2 |
| Follow-up questions | 0.62 | 15.5 |

*2) System Response Time:* The response time of system operation is very important for good user experience, because it directly impacts the interactivity. We measured end-to-end latency from the completion of a speech input to the beginning of the system's spoken response. Table VIII presents average latencies across 50 interactions for each chatbot mode.

The total average latency in chat mode was 2.77 seconds, while the lecture mode with internet connectivity averaged 3.87 seconds due to API communication overhead. When using cached responses in offline lecture mode, latency dropped to approximately 1.54 seconds. While these figures exceed our initial target of ¡500ms, they remained within acceptable limits for maintaining conversation flow with young learners.

The LLM component accounted for 77.5% of the total latency in chat mode and 83.2% in lecture mode (with internet), highlighting an area for future optimization efforts.

### B. Model Quality Evaluation

Beyond technical metrics, we evaluated the quality of responses generated by the language model component through automated analysis methods.

We evaluated response relevance using automated metrics including cosine similarity between prompts and responses, as well as a proxy measure of coherence based on perplexity. Table IX summarizes these metrics across different conversation contexts.

The system maintained acceptable coherence across various contexts, with cosine similarity scores above 0.62 and relatively low perplexity values, indicating contextually appropriate and fluent responses.

### C. Dual-Mode Assessment

We separately evaluated the performance of the system's two operational modes. The lecture mode, which operates via API calls achieved higher relevance metrics (cosine similarity of 0.79) when discussing educational content, which is expected given it's alignment with structured textbook material and access to more comprehensive language model. The chat mode, running on local LLaMA model, showed greater adaptability with vocabulary adjustment based on user responses,

though with slightly lower but still acceptable relevance scores (cosine similarity of 0.65). Table X presents comparative performance metrics between the two modes across different interaction scenarios.

TABLE X: Performance Comparison Between Operational Modes

| Metric | Lecture Mode (API) | Chat Mode (Local LLM) |
|---|---|---|
| Cosine Similarity | 0.79 | 0.65 |
| Perplexity | 12.3 | 17.6 |
| Response Time | 3.4s | 2.1s |
| Memory Usage | 1.2GB | 4.7GB |
| Internet Dependency | High | None |
| Privacy Protection | Medium | High |

The evaluation confirms our design decisions, with lecture mode providing higher accuracy at the cost of internet dependency, while chat mode offers complete privacy and faster responses with slightly lower linguistic quality.

### D. Comparative Analysis with Existing Solutions

To contextualize our system's performance, we conducted a comparative analysis against both commercial online language learning applications and traditional offline solutions. Table XI presents a summary of this analysis.

Our system demonstrates several distinctive advantages compared to existing solutions. Particularly they are lower internet dependence, privacy protection, and interactive dialog capabilities. The dual-mode approach provides a unique balance between the benefits of cloud-based solutions (when available) and completely offline operation (when necessary). This makes our system particularly suitable for deployment in educational environments with limited or intermittent internet access.

### E. Discussion of Limitations

Despite technical results, our evaluation revealed several limitations that should be acknowledged:

- **Speech recognition accuracy**: While generally acceptable, recognition accuracy declined significantly in noisy environments, potentially limiting deployment in some settings.
- **Response latency**: The average response time of 2.77 seconds is somewhat slower than commercial cloud-based solutions and could impact conversational flow.
- **Content scope**: The current implementation focuses on basic vocabulary and simple conversational patterns; more complex language structures would require larger models exceeding our hardware constraints.
- **Hardware requirements**: While relatively inexpensive, the Raspberry Pi 5 hardware represents a non-trivial cost for large-scale deployments.
- **Hybrid internet connectivity**: The chat mode is operating on locally runnign LLaMA model, while the lecture mode operates on API.

These limitations informed our recommendations for future work, as discussed in the subsequent section.

TABLE XI: Comparative Analysis with Existing Solutions

| Feature | Our System | Commercial Apps | Offline Solutions |
|---|---|---|---|
| Internet dependency | Hybrid | High | None |
| Privacy protection | High | Low-Medium | High |
| Personalization potential | Medium | Medium-High | Low-Medium |
| Interactive dialog capability | High | Low-Medium | Low |
| Response latency | 2.77s | 0.8-1.2s | N/A |
| Cost | $85 (hardware) | $20/month | $20-100 (materials) |
| Scalability | Medium | High | Low |

## VII. Conclusion and Future Work

In this paper, we made a chatbot that helps kids learn a second language. Our chatbot works without internet on a small computer called Raspberry Pi 5. This is really helpful for places where internet isn't always available.

### A. Summary of Results

Here's what we accomplished in our project:

First, we got three AI systems (speech-to-text, language model, and text-to-speech) to work together on one small Raspberry Pi computer. Our system understands kids' speech pretty well, with only about 8.2% errors when it's quiet. It takes about 2.77 seconds to respond, which is fast enough for having a conversation with a child.

Second, we figured out how to make these AI systems share the limited memory and processing power of the Raspberry Pi. This was quite challenging because these AI systems normally need much bigger computational power.

Third, we created a website that parents and teachers can use to upload or edit prompts and books. In addition they can choose the chatbot mode: chat or lecture. The former functions completely offline for casual conversation practice, while the latter uses structured lessons from textbooks (with internet). This website can save data when internet is available and sync it with the chatbot. This way, our system works both online and offline, which keeps kids' data private.

Due to computational constraints of the Raspberry Pi hardware, the implementation of certain advanced functionalities, such as image generation, quantization and fine-tuning, proved infeasible. Nevertheless, our research demonstrates the viability of developing effective educational artificial intelligence system. The analysis with existing solutions indicates that our approach offers advantages of independence from internet connectivity.

### B. Challenges Encountered

Throughout the project, several key challenges were faced that shaped the work:

One major issue is response time - the chatbot takes about three seconds to answer questions. While this works fine for simple back-and-forth exchanges, it feels sluggish during more in-depth conversations. This happens mainly because language processing requires substantial computing power on limited hardware.

Flaws were also discovered in the dual-mode system. The lecture mode works best with high-quality materials and really benefits from internet access. Meanwhile, the chat mode often

loses track during longer conversations because the local AI simply cannot match the capabilities of cloud-based systems.

Cost remains a significant barrier too. Even though a Raspberry Pi at $85 costs much less than laptops or tablets, this price point is still too expensive for many schools with tight budgets, especially when they need multiple devices for a classroom.

The current prototype only handles basic vocabulary and simple conversations effectively. Teaching more complex language concepts would require more sophisticated AI models that simply will not fit on the modest hardware.

These limitations largely reflect the fundamental tension between running advanced AI systems on affordable, portable hardware - a challenge that has guided thinking about future improvements.

### C. Future Research Directions

Based on the findings, several potential paths forward could be considered:

*1) Hardware Acceleration:* The addition of a Coral USB Accelerator could potentially boost performance. Preliminary results indicate this might reduce response times by 40-60%, potentially bringing latency under 1.5 seconds - a threshold where interactions become more efficient.

*2) Model Efficiency Improvements:* Prior attempts to reduce model size through 4-bit quantization resulted in unacceptable accuracy degradation. Newer techniques emerging in the field may offer alternatives. Further investigation into these approaches could potentially achieve model compression while maintaining functional accuracy.

*3) Language Support Expansion:* Support for additional languages beyond English could be considered. Meta's NLLB (No Language Left Behind) model could potentially facilitate support for low-resource languages such as Kazakh and other Central Asian languages with limited educational resources.

*4) Curriculum Integration:* Potential enhancements to the Book2Dial framework could focus on automated conversion of standard educational texts into interactive instructional content aligned with established curriculum standards. This might reduce barriers to institutional adoption.

*5) Distributed Learning Architecture:* A potential consideration is the implementation of federated learning - an approach that could enable multiple deployments to learn collectively from user interactions without centralizing conversation data. This method could potentially improve system performance while maintaining data privacy.

*6) Web Interface Enhancements:* Potential improvements to the web interface could include advanced progress monitoring, individualized learning pathways, and integration with institutional assessment systems to enhance utility for educators and parents.

In conclusion, despite the inherent limitations of resource-constrained computing platforms such as the Raspberry Pi, this work demonstrates the feasibility of developing effective language learning tools that function independently of internet connectivity. As computational capabilities of edge devices increase and AI models become more efficient, such systems could potentially offer alternatives to cloud-based educational technologies while preserving user privacy.

## REFERENCES


[1] J. Cummins, "Cognitive/academic language proficiency, linguistic interdependence, the optimum age question and some other matters," *Working Papers on Bilingualism*, vol. 19, pp. 121–129, 1979.

[2] K. Hakuta, *Mirror of language: The debate on bilingualism*. Basic Books, 1986.

[3] A. Kukulska-Hulme, "Mobile language learning 2.0. beyond language learning?" vol. 1, no. 1, pp. 57–68, 2009.

[4] A. Berns, J. M. Mota, I. Ruiz-Rube, and J. M. Dodero, "Exploring the potential of a 360° video application for foreign language learning," in *Proc. 6th Int. Conf. Technol. Ecosyst. Enhancing Multicult.*, 2018, pp. 776–780.

[5] L. K. Fryer, M. Ainley, A. Thompson, A. Gibson, and Z. Sherlock, "Stimulating and sustaining interest in a language course: An experimental comparison of chatbot and human task partners," *Comput. Human Behav.*, vol. 75, pp. 461–468, 2017.

[6] E. Atwell, *The language machine: The impact of speech and language technologies on English language teaching*. London: The British Council, 1999.

[7] Y. F. Wang and S. Petrina, "Using learning analytics to understand the design of an intelligent language tutor–chatbot lucy," *Int. J. Adv. Comput. Sci. Appl.*, vol. 4, no. 11, pp. 124–131, 2013.

[8] L. Fryer and R. Carpenter, "Bots as language learning tools," *Lang. Learn. Technol.*, vol. 10, no. 3, pp. 8–14, 2006.

[9] T. Kanda and H. Ishiguro, "Communication robots for elementary schools," in *Proc. Symp. Robot Companions: Hard Probl. Open Challenges Robot-Human Interact.*, 2005, pp. 54–63.

[10] H. Yang, H. Kim, J. H. Lee, and D. Shin, "Implementation of an ai chatbot as an english conversation partner in efl speaking classes," *ReCALL*, vol. 34, no. 3, pp. 327–343, 2022.

[11] Y. Zheng, Y. Chen, B. Qian, X. Shi, Y. Shu, and J. Chen, "A review on edge large language models: Design, execution, and applications," *ACM Comput. Surv.*, vol. 57, no. 8, p. 209, Mar 2025.

[12] Z. Nezami, M. Hafeez, K. Djemame, and S. A. R. Zaidi, "Generative ai on the edge: Architecture and performance evaluation," *arXiv preprint arXiv:2411.17712*, 2024.

[13] X. Qu, J. Wang, and J. Xiao, "Quantization and knowledge distillation for efficient federated learning on edge devices," in *2020 IEEE 22nd International Conference on High Performance Computing and Communications (HPCC)*, 2020, pp. 683–690.

[14] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "Tinybert: Distilling bert for natural language understanding," *arXiv preprint arXiv:1909.10381*, 2019.

[15] DFRobot, "Performance analysis of slm (mathstral, phi 3, llama 3.1, gemma2 2b, and qwen) on raspberry pi 5 sbc," 2024.

[16] J. Smith *et al.*, "Faster and lighter llms: A survey on current challenges," *arXiv*, Feb 2024, [Online]. Available: https://arxiv.org/html/2402.01799v2.

[17] A. Researcher, "I ran 9 popular llms on raspberry pi 5; here's what i found," ItsFOSS, Oct 2024, [Online]. Available: https://itsfoss.com/llms-for-raspberry-pi/.

[18] Imagimob, "Generative ai on the edge: What does the future hold?" Imagimob Blog, Mar 2025, [Online]. Available: https://www.imagimob.com/blog/generative-ai-on-the-edge-what-does-the-future-hold.

[19] K. Sakai, Y. Uehara, and S. Kashihara, "Implementation and evaluation of llm-based conversational systems on a low-cost device," in *2024 IEEE Global Humanitarian Technology Conference (GHTC)*, 2024, pp. 398–401.

[20] K. Hiroshige, "Piper: A fast, local neural text to speech system," 2024, available: https://github.com/rhasspy/piper.

[21] R. Prenger, R. Valle, and M. Norouzi, "Glow-tts: A generative flow for text-to-speech via monotonic alignment search," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 4770–4774.

[22] A. Radford *et al.*, "Robust speech recognition via large-scale weak supervision," *arXiv preprint arXiv:2212.04356*, 2022.

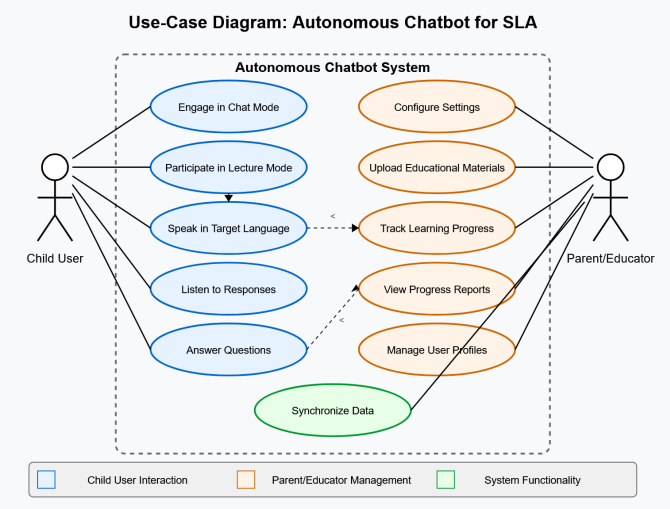


Fig. 3: Use-Case Diagram for the Autonomous Chatbot for SLA system showing the interactions between Child Users, Parent/Educators, and the system functions.

## APPENDIX A
## USE-CASE DIAGRAM OF THE AUTONOMOUS CHATBOT

The Fig. 3 illustrates the use cases for our Autonomous Chatbot for Second Language Acquisition. It shows the interactions between the two main actors (Child User and Parent/Educator) and the system, highlighting the dual-mode operation (Chat and Lecture modes) as well as the management functions available through the web application.

The Child User interacts with the system through five main use cases: engaging in Chat Mode for free-form conversation practice, participating in Lecture Mode for structured learning, speaking in the target language, listening to responses, and answering questions as part of the learning process.

The Parent/Educator interacts with the system primarily through the web application interface, with use cases including configuring system settings, uploading educational materials for the Book2Dial tool, tracking learning progress, viewing detailed reports, and managing user profiles. They can also synchronize data between the local database and cloud storage when internet connectivity is available.

## APPENDIX B
## PERFORMANCE TABLES FOR AI MODELS ON RASPBERRY PI 5

This appendix presents detailed performance benchmarks for the three key AI component types used in our system: Text-to-Speech (TTS) models, Automatic Speech Recognition (ASR) models, and Language Models, all evaluated on the Raspberry Pi 5 hardware platform.

### A. Text-to-Speech (TTS) Performance

Table XII presents performance metrics for various Text-to-Speech models tested on the Raspberry Pi 5. Piper-TTS emerged as the optimal solution with an inference time of just 0.87 seconds per sentence while maintaining acceptable voice quality.

TABLE XII: Text-to-Speech (TTS) Model Performance on Raspberry Pi 5

| Model | Inference Time | Notes |
|---|---|---|
| Piper-TTS | 0.87s | Faster than real time/lower quality of voice |
| Tacotron2 | 25.3s | Good audio quality, terrible latency |
| Nix-TTS | 9.6s | Acceptable voice sound, long response time |
| glow-tts+mb_melgan | 1.14s | Little slower than piper but voice sounds more realistic |

### B. Automatic Speech Recognition (ASR) Performance

Table XIII shows performance metrics for various Whisper model variants when processing a 10-second audio sample. Whisper Tiny.en and Base.en demonstrated practical performance, while Small.en and Medium.en proved too resource-intensive for real-time operation.

TABLE XIII: Automatic Speech Recognition (ASR) Model Performance on Raspberry Pi 5

| Model | Transcription Time | Notes |
|---|---|---|
| Whisper Tiny.en | ∼6s | Faster than real-time, suitable for live transcription |
| Whisper Base.en | ∼10s | Close to real-time, slight queue growth |
| Whisper Small.en | >30s | Not suitable for real-time due to long processing time |
| Whisper Medium.en | N/A (System Freeze) | Exceeds 4GB memory, impractical for Pi 5 |

### C. Language Model Performance

Table XIV presents performance metrics for various language models running on the Raspberry Pi 5. Llama 3.2 (1B) was selected for our implementation due to its optimal balance of response quality, inference speed, and memory usage.

These performance benchmarks informed our component selection decisions, enabling us to optimize for the resource constraints of the Raspberry Pi 5 hardware while maintaining acceptable functionality for a real-time language learning application.

TABLE XIV: Language Model Performance on Raspberry Pi 5

| Model | Parameters | Size | Avg. Response | RAM Usage | Notes |
|---|---|---|---|---|---|
| Llama 3 | 8B | 4.7GB | 65s | ∼5.2GB | Slow, verbose, suitable for non-real-time tasks |
| Llama 3.2 | 1B | 2.48GB | 3s | ∼1.8GB | Fast, good balance in terms of logic and latency |
| Phi-3 | 3B | 2.4GB | 27s | ∼4.9GB | Fast, good for logic tasks, minor hallucinations |
| Gemma | 2B | 1.7GB | 3s | ∼3.4GB | Fast but sometimes unresponsive, struggles with basic questions |
| Mistral | 7B | 4.1GB | 7s | ∼5.3GB | Chatty, second-slowest, errors in complex tasks |
| Qwen | 0.5B | 394MB | 1s | ∼3.1GB | Fastest but inaccurate, errors in math and facts |